Multiresolution coding of 3D meshes

Marc Antonini

Directeur de Recherche CNRS

am@i3s.unice.fr

http ://www.i3s.unice.fr/~am

Laboratoire I3S

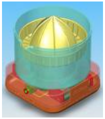
MULTIMEDIA IMAGE CODING AND PROCESSING GROUP

Université de Nice-Sophia Antipolis - CNRS

The applications

[Online] Gaming On PC
Performance for HighRes 3D models,
Smaller size of datasets
Network and buses transfer rate

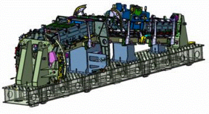
[Online] Embedded Gaming
Performance for quality 3D models
on restricted hardware
Network and buses transfer rate



CAD/CAM and Design
For storage and transfer
of datasets

DCC and Production
Brings new capabilities on modelers

Medical and Seismic
Lets build huge datasets
from volume images

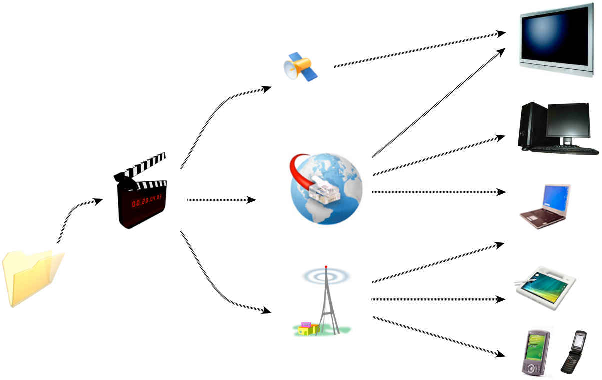


Virtual Mockup
Lets build huge datasets
from Huge CAD/CAM assemblies

Tele conferencing
Transport realistic clones
on traditional networks

Required functionality : the scalability

Different clients, different channels, **ONE mesh or animation file**



Scalability

Different kinds of scalability

- Resolution (spatial or temporal)
- Rate
- Quality
- Complexity
- Region of interest (ROI)
- etc.

Support of scalability

- Usually causes
  - → Complexity increase
  - → Performance drop
- Alternative : multiresolution and **wavelet-based coders**

Outline

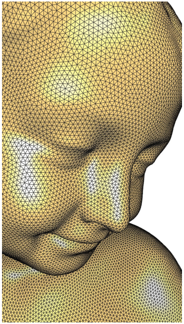
1. 3D mesh coding : Problem statement
2. The semi-regular remeshing
3. Coding the meshes
4. Demo : coding/decoding multiresolution meshes
5. Discussion and challenges

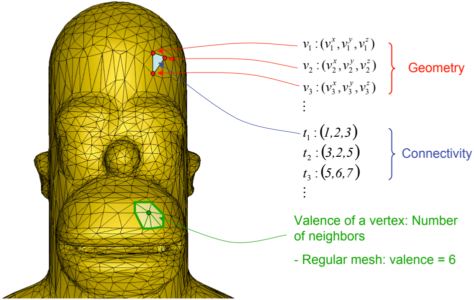
Outline

What is a surface mesh ?

A surface triangle mesh is composed by

- A geometry : the position of vertices in $\mathbb{R}^3$ (irregular sampling)
- A connectivity : the connections between the vertices

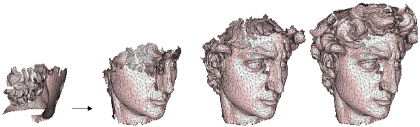




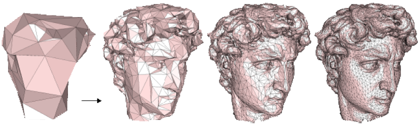
State of the art

Single rate compression (Lossless)

- No asumption on the mesh
- Specialized for massive datasets which cannot fit entirely into memory
- Encoding of connectivity (*e.g.Touma-Gotsman, topological surgery, Edgebreaker*) or based on remeshing (*e.g. geometry images*)



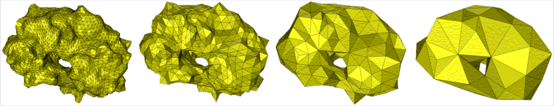
Progressive compression (Lossy to lossless)



State of the art : Progressive compression

Two kinds of approaches

- Based on simplification/refinement (decimation, edge collapse, vertex split)
- Based on **multiresolution analysis** (wavelets)



Objective

Rate-distortion optimization between data size and approximation accuracy

Multiresolution for irregular meshes ?

Two options for computing the transform

- Without connectivity modification

  e.g. wavelet transform for irregular meshes (Valette, Prost 2004)

- A mesh is considered as one instance of the surface geometry

  → **REMESHING** operation

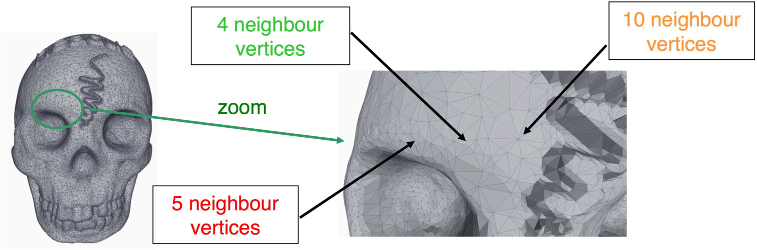
  → Create regular and uniform geometry sampling

  → **Wavelet transform (DWT) for semi-regular meshes**

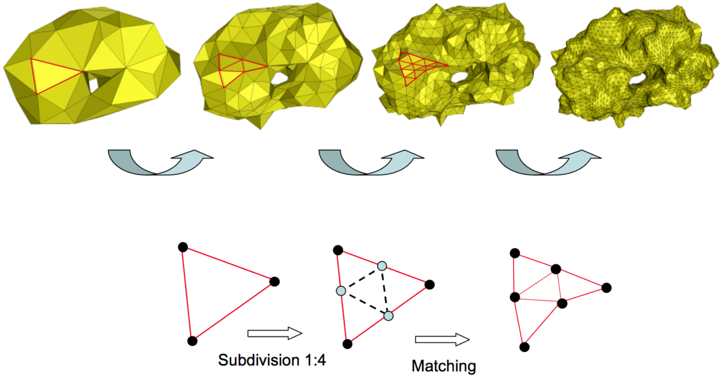
Outline

Irregular meshes

- Irregular sampling $\rightarrow$ valency $\neq 6$



The semi-regular mesh : a multiscale data



Advantages of semi-regularity

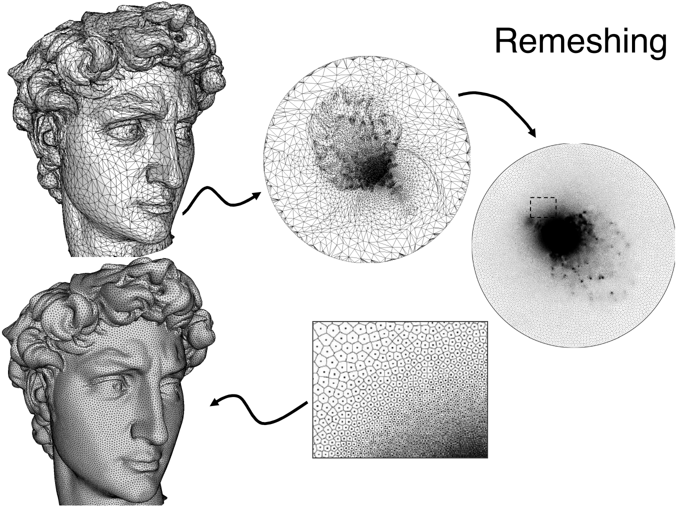
- Multiresolution structure
- Quasi-implicit connectivity (only base mesh connectivity)
- Efficient compression
- Progressive transmission
- Scalability properties

The most famous semi-regular remeshers

- **MAPS** [Lee et al (1998)]
  - → A coarse mesh containing geometry and connectivity
  - → N_1 sets of 3D details (ONLY geometry) (3 floating numbers)
- **Normal meshes** [Guskov et al 2000]
  - → A coarse mesh containing geometry and connectivity
  - → N_2 sets of 3D details (ONLY geometry) (1 floating number, i.e., the normal to the surface)
  - → MORE COMPACT semi-regular representation
- **Globally smooth parametrization** (GSP) [Khodakovsky et al 2003]
- **Variational normal meshes** (VNM) [Khodakovsky et al 2004]
- **TriReme** [Guskov et al 2007]

→ **Methods based on 2D PARAMETERIZATION**

Remeshing using a 2D parameterization

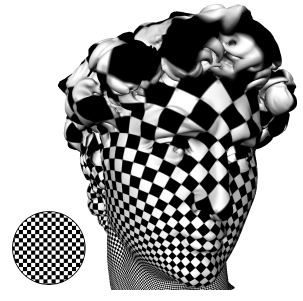


images from P. Alliez

The 2D parameterization of 3D meshes

Requirements (slide from P. Alliez and G. Gotsman)

- Bijective
- Minimal distortion
  - Preserve 3D angles
  - Preserve 3D distances
  - Preserve 3D areas
  - No 'strech'



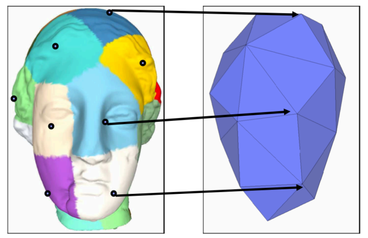
A remeshing solution without parameterization

I3S solution based on Lloyd relaxation

Main idea : Construct progressively a Voronoi partition of the irregular mesh geometry

Basic principle :

- Simplification step : Create a Voronoi tesselation of the irregular mesh with few regions



- Refinement step : Add semi-regular Voronoi seeds to refine the tesselation

Construction of a Voronoi tesselation

Two optimal conditions :

- Nearest neighbor condition

  → The Voronoi tesselation of $\mathbb{R}^n$ in L clusters R_k is given by

$$R_k = \{v \in \mathbb{R}^n / d(v, s_k) \leq d(s, s_j) \quad \forall j \in \{1, 2, ..., L\}\}$$

  where $d(u, v)$ stands for the geodesic distance [a]

- The centroid (or mass center) condition

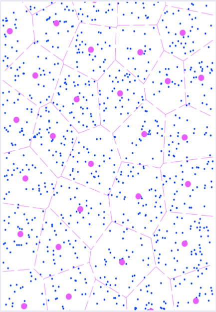
$$s_k = \frac{\int_{R_k} v\rho(v)dv}{\int_{R_k} \rho(v)dv}$$

  where $\rho(v)$ corresponds to the mass of v

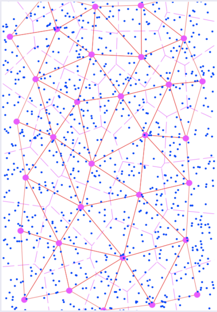
a. Can be computed by Dijkstra algorithm

The Lloyd's relaxation

Example of tessellation of $\mathbb{R}^2$



Voronoi

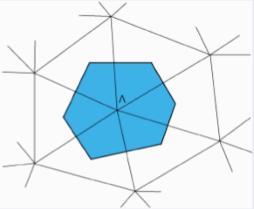


Dual : Delaunay triangulation

Tesselation of a surface mesh

Use the geometry of the original mesh as input data

- Let $S = \{v_i, i = \{0, 1, ..., N-1\}\}$ be the set of vertices in $\mathbb{R}^3$ of a irregular surface mesh.
  - $\rightarrow$ S is considered as the input data to be meshed
- The mass $\rho(v)$ is considered as the area of the dual cell of v



The mesh simplification

Idea

- Construct a Voronoi Tesselation with a small number of clusters
- Use the Lloyd's relaxation on the input data set S

Principle of the algorithm

→ Initial conditions :

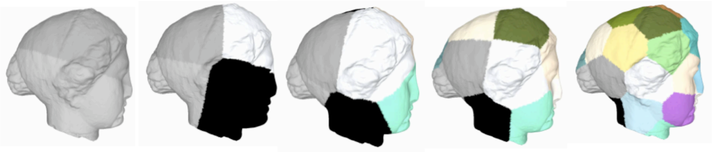
- Let V the desired number of vertices in the simplified mesh
- Select V seeds (high curvature or dart throwing...)

→ Apply the Lloyd's relaxation until convergence

→ Project the final centroid onto the original mesh

The mesh simplification

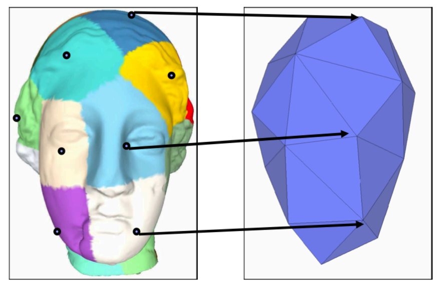
Example of a surface tesselation



The mesh simplification

How to obtain the base mesh ?

- Keep the mass centers created by the Lloyd's relaxation
- Construct the Delaunay triangulation

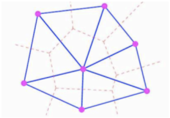


Voronoi tesselation (left) and the corresponding mesh (right)

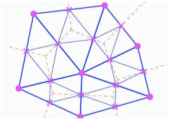
Refinement by subdivisions of the base mesh

At each subdivision level (resolution)

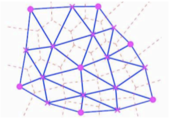
- Subdivise the triangles (1 : 4 subdivision)
- Consider the added vertices as Voronoi seeds
- Update the tesselation using Lloyd's relaxation



first resolution



Add Voronoi seeds

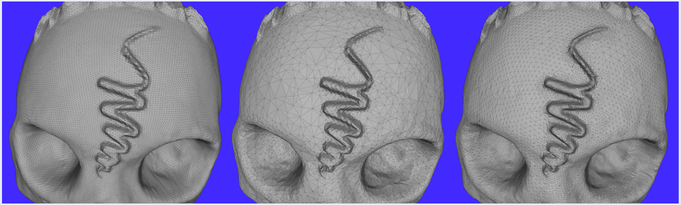


Update tesselation

Example of remeshing

Adaptive vs non adaptive refinements

Remeshing of skull

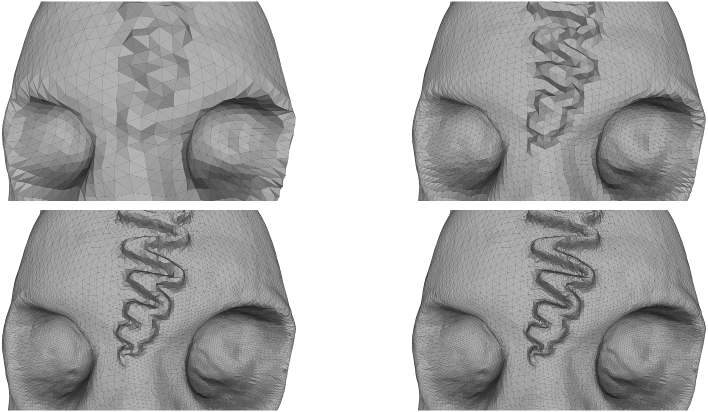


LEFT Uniform subdivision [Guskov et al. '01]
(262 144 triangles - RMSE = 2.09×10^{-2})

MIDDLE Original

RIGHT Remeshed with adaptive subdivision (in-house solution)
(140 544 triangles - RMSE = 1.05×10^{-2})

Multiresolution properties of the adaptive mesh



How to measure the remeshing distortion ?

The surface-surface distance

- The point-surface distance

$$d(p, S') = min_{p' \in S'} \|p - p'\|_2$$

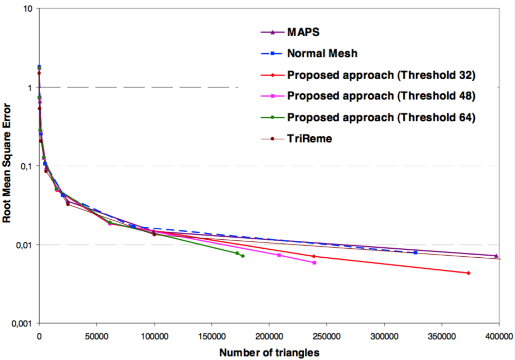
- The unilateral distance between 2 surfaces S and S'
  - **RMSE** $\rightarrow \bar{d}(S, S') = \left(\frac{1}{|S|} \int_{p \in S} d(p, S')^2 \mathrm{d}s \right)^{\frac{1}{2}}$
  - **Hausdorff distance** $\rightarrow \bar{d}(S, S') = max_{p \in S} d(p, S')$

$\rightarrow$ The symmetrical surface-surface distance

$$d_{sym}(S, S') = max[\bar{d}(S, S'), \bar{d}(S', S)]$$

The remeshing error

RSME in function of the number of triangles for Venus

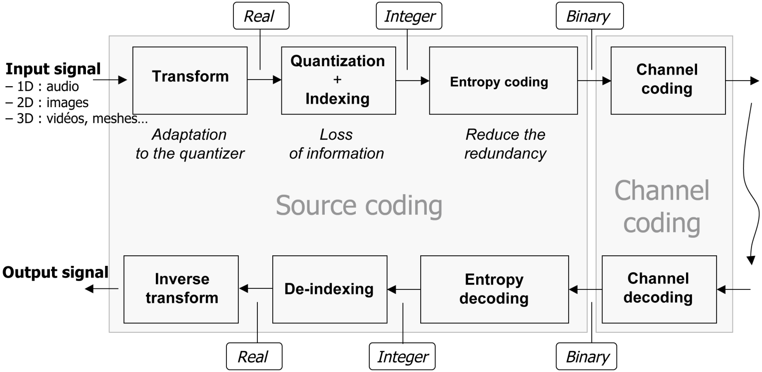


RMSE given by MESH software

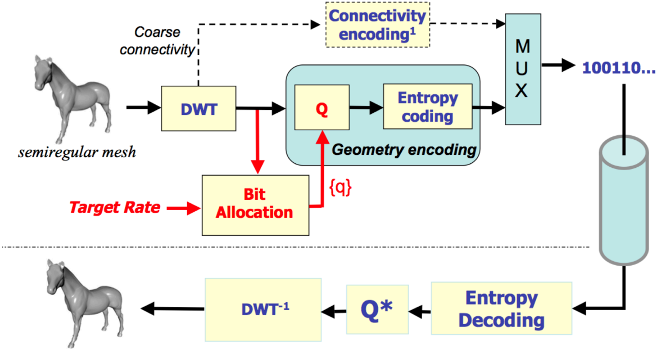
$$\bar{d}(S, S') = \sqrt{\frac{1}{|S|} \int_{p \in S} d(p, S') \mathrm{d}s}$$

Outline

General coding scheme



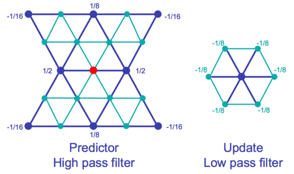
Typical coding scheme for semi-regular meshes



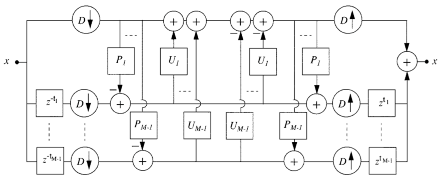
[1] Connectivity encoding: Touma-Gotsman coder

Butterfly-based wavelet transform (1996)

→ A lifting scheme implementation - Interpolating filter

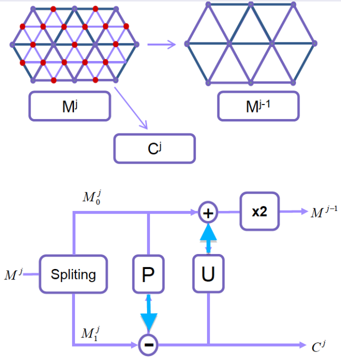


→ The 4-Channels lifting scheme



The lifting scheme

Principle



- PREDICTION :

$$C^j = M_1^j - P * M_0^j$$

- UPDATE :

$$M^{j-1} = 2 \times (M_0^j + U^j * C^j)$$

DWT for remeshed surfaces

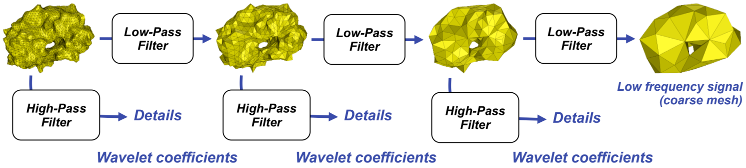
Properties for compression

→ The connectivity is implicit except for the coarse mesh

→ Only the geometry (wavelet coefficients) must be coded

Optimize the rate-distortion trade-off !

→ **Bit allocation**



Optimal bit allocation

Objective

- Given a rate constraint $\sum_{i=1}^{M} a_i R_i \leq R_{\mathrm{MAX}}$,
- Determine the optimal set of bit-rates $\mathbf{R} = \{R_i\}_{i=1}^{M}$
- Which minimizes global distortion $D(\mathbf{R})$,
- Knowing that $D(\mathbf{R}) = \sum_{i=1}^{M} w_i D_i(R_i)$

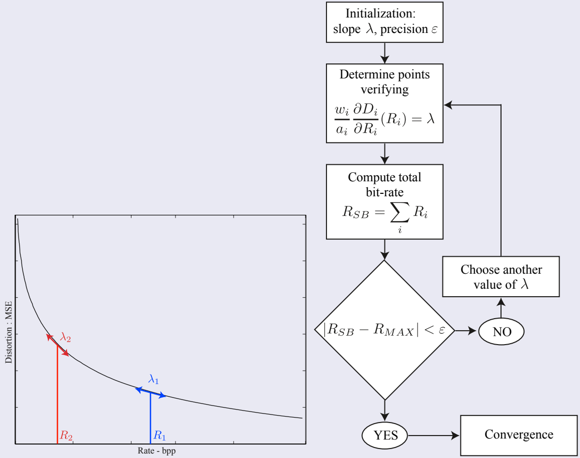
Lagrangian optimization : minimize

$$J(\mathbf{R}, \lambda) = \sum_{i=1}^{M} w_i D_i(R_i) - \lambda\left(\sum_{i=1}^{M} a_i R_i - R_{\mathrm{MAX}}\right)$$

λ : common slope to curves $D_i(R_i)$
hypothesis : $D_i(R_i)$ are convex and monotonic

Optimal bit allocation : algorithm

Determine the rate corresponding to the slope λ



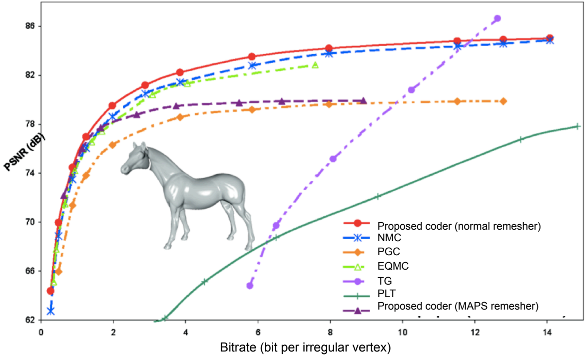
for each curve $D_i(R_i)$ corresponding to subband i

Calculus of $\frac{\Delta D_k}{\Delta R_k}$

Two kinds of approaches

- **Signal based :** BRUTE FORCE coding
  - Estimation of the real bitrate and distortion (EZW and SPIHT family, JPEG 2000)
- **Model based :** theoretical models
  - Asymptotical modeling of the rate-distortion function (Shannon, Bennett, Zador,...)
  - Exact modeling (in some cases)
    1- exact modeling in the case of scalar quantization
    2- approximation of the rate-distortion function using "smoothing splines"

Coding/decoding results [1]



$$PSNR_{dB} = 20log_{10}\frac{BoundingBox}{d_{sym}(S,S')}$$

1. results from F. Payan phD thesis

Some visual coding/decoding results [2]

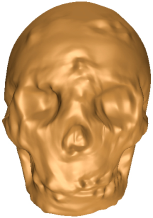
Input mesh (irregular)

CR = 45
4.07 bits/iv

CR = 83
2.2 bits/iv

CR = 226
0.82 bits/iv

CR = 900
0.2 bits/iv



Object Skull:
20 002 irregular vertices (40 000 triangles)
131 074 semi-regular vertices

2. results from F. Payan phD thesis

Outline

Visualization and manipulation application

Motivation

- Visualize massive meshes (> millions of triangles)
- "Real time" rendering
- Scalability (resolution, rate, ROI...)
- Parallel processing $\rightarrow$ use of Vector Quantization

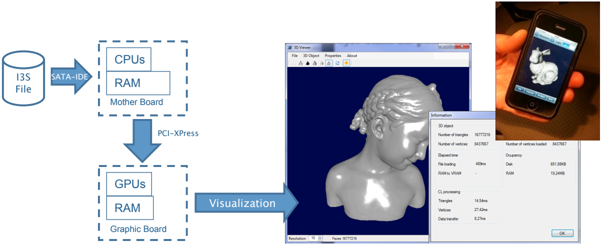
Bottleneck

- The DATA BUS between HDD, RAM and VRAM ! !
  - $\rightarrow$ Slow data transmission compared to Tera flops computation capacity of today Graphic Cards
  - $\rightarrow$ DATA BUS seen as a **low bandwidth transmission channel**

Visualization and manipulation application

Solution

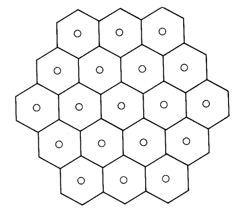
- Push COMPRESSED GEOMETRY to the VRAM
- Decoding INSIDE the GPU (GPGPU implemented)

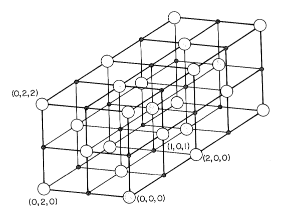


Quantizing the DWT of the geometry

The lattice vector quantization solution (LVQ)

- Why LVQ ?
  - $\rightarrow$ VQ is well suited for geometry coding
    - a vertex v is a vector $\in \mathbb{R}^3$
    - a triangle T is a vector $\in \mathbb{R}^9$
    - ...
  - $\rightarrow$ LVQ is a structured vector quantizer defined as $\Lambda \in \mathbb{R}^n$
    $\Lambda = \{\mathbf{x} | \mathbf{x} = u_1\mathbf{a}_1 + u_2\mathbf{a}_2 + ...u_n\mathbf{a}_n\}$ where $a_i \in \mathbb{R}^m$ $(m \geq n)$
- Examples

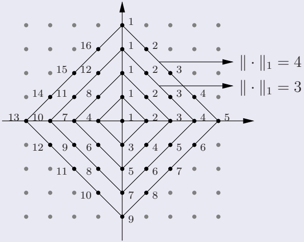




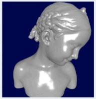
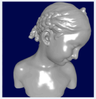
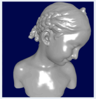
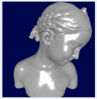
Using a LVQ for quantization

The product code

- A group of triangles of the mesh is quantized by a lattice point
- Each lattice point is indexed using a product code composed by
  - $\rightarrow$ the lattice point norm
  - $\rightarrow$ its position on a surface with constant norm



Loading and rendering time consumption

	DAT Format State of the art No compression	I3S Format No compression (No GPGPU)	I3S Format Compressed 1:50 (GPGPU Implemented)	I3S Format Compressed 1:100 (GPGPU Implemented)
6 750 K Triangles	 194 MB Disk 225 MB RAM **49 s Load** Display 166 ms	 36 MB Disk 225 MB RAM **7 s Load** Display 166 ms	 732 KB Disk 8.4 MB RAM **280 ms Load** Display 30 ms	 365 KB Disk 8 MB RAM **210 ms Load** Display 21 ms
16 750 K Triangles	 480 MB Disk 567MB RAM **127 s Load** Display 420 ms	 100 MB Disk 567 MB RAM **19 s Load** Display 420 ms	 1 960 KB Disk 21 MB RAM **590 ms Load** Display 67 ms	 1 000 KB Disk 20 MB RAM **515 ms Load** Display 55ms

GPU: 7x48 OpenCL cores @ 1430MHz, VRAM: 1GB, Bandwidth: 115.2GB/s

Demonstration : visualizing huge meshes

Characteristics of the material

- NVIDIA GeForce 320M (GPU)
- Memory (RAM) = 4 Go
- Memory (VRAM) = 256 Mo

Characteristics of the mesh (BIMBA)

- Number of triangles = 16 750 000
- Number of vertices = 8 437 666
- Total cost on HDD = 290 Mo
- Total cost including the normals = 386 Mo

Outline

Discussion

Marc ANTONINI

Semi-regularity allows

→ Implicit connectivity

→ DWT multiresolution analysis

→ Good scalability properties

Wavelets and vector quantization allow

→ highly parallel coding/decoding

→ last moment GPU decoding
solving the data transfer bottleneck on data buses

→ multiresolution technology
minimizing drastically the GPU resources needed

→ to visualize or manipulate multi-millions triangles objects on Workstations (multi-thousands on Smartphones)

Challenges

- Construct or scan directly semi-regular meshes

$\rightarrow$ NO REMESHING operation

- Deal with more complex objects and/or more detailed

$\rightarrow$ OUT-OF-CORE data

- 3D ANIMATIONS

- Take into account human visual perception

$\rightarrow$ What EFFICIENT PERCEPTUAL DISTORTION MEASURE ?

Is there anything left to do in image coding ?

Marc ANTONINI

From massive to out-of-core data... With billions of faces !

[Rendering of the 940 millions faces done by I3S lab]

(MICHELANGELO PROJECT - DAVID 940 MILLIONS FACES, FILE SIZE > 20 GBYTES)

Thank you ! !

Acknowledgement to
F. Payan, L.H. Fonteles, A. Meftah