A General Method for Preserving Attributes Values on Simplified Meshes

P. Cignoni, C. Montani, C. Rocchini, R. Scopigno

Istituto Elaborazione dell’Informazione
Italian National Research Council (C.N.R.)
Pisa, Italy
Overview

- Mesh simplification
- Current approaches to preserve attribute detail in mesh simplification
- Our texture-based approach:
  - retrieval from the original mesh of the attributes detail
  - coding attributes detail into textures
- Evaluation of results
- Extension to multi-resolution representations
- Conclusions and further work
Mesh simplification and LOD encoding:

- Objective: produce the **simplest** mesh that satisfies the accuracy required by the application

- Many good solutions proposed for *shape-oriented* simplification

- What if the mesh holds also crucial **attributes**? (e.g. color)
An example

A real object

Mesh simplification

130K faces!

20K faces!

2M faces!

Range scan

AND COLOR

??

IEEE Vis'98
Preserving detail on simplified meshes

- **Problem Statement:**

  how can we preserve on a *simplified* surface most of the **detail** (or **attribute values**) defined on the *original* surface?

- **What one would preserve:**

  - **color** (per-vertex or texture-encoded)
  - **high frequency shape detail** (bumps)
  - **scalar/vector fields** (e.g. Sci.Viz. applic.)
  - **procedural textures** mapped on the mesh
Preserving detail : State of the art

Approaches proposed in literature:

- **integrated** in the simplification process
  (i.e. ad hoc solutions **embedded** in the simplification code)
  - use an enhanced **approximation evaluation metrics**
    [Hoppe96, Frank etal98, Garland etal98, Cohen etal98]
  - store removed detail in **textures**
    [Krish.etal96, Maruka95, Soucy etal96]
  - preserve **topology** of the attribute field

Our approach:

- **independent** from the simplification process
  (post-processing phase to restore attributes detail)
Our approach

Simplification-Independent detail preservation:

- **general w.r.t. simplification**: attribute/detail preservation is not part of the simplification process
  - de-couples shape simplification and attribute preservation
  - performed as a **post-processing** phase (after simplification)
  - any simplifier can be adopted

- **general w.r.t. detail**: any attribute can be preserved, by constructing ad-hoc **texture maps**
  - preserving multiple attributes does not increase code complexity or processing overhead

- **efficient in time**
...Our approach...

original mesh + detail

Mesh Simplifier
- Jade v.2
- Quadric Metrics

Detail Preservation

simplified mesh

Renderer
(with texture mapping)
texture-encoded detail

textured image
A simple idea:

Phase 1
- for each simplified face:
  - detect the original detail
  - store it into a **triangular texture patch**

Phase 2
- pack all textures patches into a std. rectangular texture
Phase 1: Recovering Detail

Given an original mesh $M$ and a simplified mesh $S$:

for each triangular face of $S$ produce a texture patch, which encodes the “detail” of $M$ lost in $S$

- Scan-convert each face of $S$
  - for each sample point $p$
    - find the corresponding point $p'$ on the original $M$
    - compute the attribute value in $M$ on $p'$
    - store this value into the corresponding texel of a triangular texture patch
Phase 2: Pack the Texture Patches

- Store **texture patches** in an efficient manner into a single, std. **rectangular texture**
  - use std. textures to be compatible with std. texture mapping sw/hw
    - **rgba** textures rendering interactive on most graphics system
    - hw-assisted management of bump maps forthcoming

- Texture patches can be packed in two different manners:
  - restrict to **regular** texture patches [Maruka95, Soucy et al96]
  - support **not regular** patches shapes **<- our choice**
Surface sampling

- **Sampling step** determines:
  - texture **size** and **quality**, running **time**

- **Sampling:**
  - scan-convert face $f$ of $S$
  - for each sampling point $p$
    - find nearest face $f'$ in $M$ (kernel action, efficient via the use of a **bucketing data structure**);
    - compute corresponding point $p'$ on $f'$

- Why looking for **nearest points** and not for points on the **normal direction**?
Sampling: a possible problem

- when mesh section is very thin, incorrect “nearest points” can be produced

Heuristic adopted:

- return the nearest point $p'$ such that the corresponding face has orientation compatible with that on point $p$
Multisampling

- improves texture quality, without increasing its size
- for each texel:
  - evaluate multiple samples
  - texel := samples average
- particularly useful on meshes with highly discontinuous detail, reduces aliasing
- sampling times (R4400 200MHz):

<table>
<thead>
<tr>
<th>Multisampl.</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>14.71 sec</td>
</tr>
<tr>
<td>2x2</td>
<td>58.43 sec</td>
</tr>
<tr>
<td>3x3</td>
<td>129.44 sec</td>
</tr>
</tbody>
</table>
We can sample any field/quantity defined on the surface:

- RGB color, given on a per-vertex base
- RGB color, given via texture-s
- High frequency shape detail (interpolation of normals or distances \( d( M(p') - S(p) ) \))
- Scalar / vector field

Output:
- rgb texture  
- bump/displacement text.  
- field value text.
Texture patches

Patch size:
- discrete set of possible heights ($2^i$ texels), to allow easy packing

Packing algorithm depends on patch shape:
- **regular** shape (rectilinear):
  - very easy to pack
  - different sampling rate in the two axes
- **not-regular** shape:
  - slightly more complex to pack,
  - more compact in shape
  - lower aliasing (identical sampling step in the two directions)
Packing Texture Patches

- **Regular shape**: straightforward
  [Soucy et al. 96]

- **Irregular shape**:
  - use heuristic rules or an optimization process
    (optimal packing NP hard)
  - **our choice**: simple heuristic
Packing heuristic

- divide patches in buckets, ordered by **height** (discrete set);
- process buckets in order of decreasing height (tallest first):
  - **loop**
    - find the face which adapts better to the *open edge slope*
    - copy it in the texture
  - **until** no more faces in the bucket

- each face has 4 possible slopes (flip in the two direction)
- gaps in the textures if slopes do not match precisely
...Packing Texture Patches

- In average, texture sizes: \textit{regular} \approx 2 \times \textit{non-regular}

- Overhead of \textit{non-regular packing} on the \textit{bunny} mesh:
  - patch\_expansion (2^l height) 12%
  - patch\_borders 28%
  - text\_gaps 4%
  - text\_tails 12% Total 67%

- An example on the \textit{fandisk} mesh (98 faces):
  \begin{itemize}
  \item regular patches
  \item non-regular patches
  \end{itemize}
  \begin{itemize}
  \item Size: 1024 x 640 texels
  \item Size: 978 x 256 texels
  \end{itemize}
Results: preserving color detail

- Mesh simplification
- Preserving Detail
- Textured rendering

original mesh

rgb texture

simplified mesh

Total time: 12.3 sec
Fandisk mesh: 12K --> 98 faces
Texture size: 978 x 256 = 240K texels
Results: preserving textured-color detail

- **Total time:** 6.1 sec. (9.8 w. I/O)
- **Face mesh:** 10K --> 1,745 faces
- **Texture size:** 702 x 128 = 87K texels
Results: preserving shape detail

Original mesh

Mesh simplification

Preserving detail

Bump Texture

Texture size: 719 x 128 = 83K texels

Total time: 27.2 sec. (38.7 w I/O)

Bunny mesh: 69K --> 251 faces

Textured rendering

Simplified mesh w. bump map
Results: preserving field detail

- Original mesh
- Mesh simplification
- Preserving detail
- Simplified mesh
- Textured rendering

Field value texture (after Trasf. Funct. Mapping)

- Total time: 26.2 sec. (88.6 w. I/O)
- Plane mesh: 87K --> 500 faces
- Texture size: 656 x 128 = 82K texels
Integrating different detail maps

- **Color only**
  - (RGB texture, HW gouraud shading)

- **Shape only**
  - (pre-shaded bump map)

- **Color + shape**
  - (pre-shaded RGB texture using bump map, SW)

Geometry used in all 3 images:
- simplif. fandisk,
  - 98 faces
Procedural Textures

- **Procedural textures:**
  - widely used to synthesize complex materials
  - require software rendering

- Use the same approach also to produce a *std. 2D texture* which stores all the detail that the *procedural texture* paints on the object surface:
  - sample the surface,
  - for each sampling point evaluates the procedural texture

- Use procedural textures also on HW-assisted systems!

geometry (3250 faces) + 2D “procedural” texture
If we have a simplified mesh, procedural textures may be applied (or, in our case, sampled) on:

- **texture**: sampled on simplified mesh
  - **rendering**: simplified mesh

- **texture**: sampled on original mesh
  - **rendering**: simplified mesh

Equivalent to applying proc.text to the simplified mesh.
A “wooden” bunny
- procedural text.
  - shape text.

geometry:
- 251 faces
Results

- Show meshes...
Multiresolution Management

Extend this approach to multiresolution repr.:

- **Linear sequences**
  - list $F = \{ f_i \}$ of all the faces produced during simplification
  - for each face $f$ we store its accuracy interval $(\min_f, \max_f)$, such that $f$ is part of a simplified mesh at accuracy $\varepsilon$ IFF:
    \[
    \min_f \leq \varepsilon \leq \max_f
    \]

- **Preserving detail on linear sequences:**
  - **pre-processing:** build the detail texture associated to the list $F$
  - **run-time:** given an accuracy $\varepsilon_1$
    - extract from $F$ all faces $F' = \{ f \}$ such that: $\min_f \leq \varepsilon_1 \leq \max_f$
    - extract from the multires texture all texture patches associated to $F'$ and pack them in a new texture map
When we sample texture patches for a multiresolution mesh:

- sampling step may be **constant**

- sampling step may **depend on** current face **accuracy**
  (lower is accuracy, coarser is sampling)

Choice depends on applications:

- similar quality of preserved detail on meshes with different size

- detail quality proportional to geometric size (e.g. construction of LOD models)
Multiresolution Detail Textures:

- number of faces in a **linear sequence** is ~2.5 no. faces in original mesh
- **size of multires. detail texture** depends on the total surface area of the faces in the linear sequence
- on a number of experiments: 3x .. 10x

To **reduce** multires. texture size:

- do not represent faces in the **head** and the **tail** of the linear sequence
  (i.e. with accuracy $< \varepsilon_{\text{min}}$ and $> \varepsilon_{\text{max}}$)
Conclusions

Detail preserved via **patched textures**

- general solution, simplification-independent
- allows to recover multiple attributes
- highly efficient (<1min)
- accuracy depends on sampling resolution (user-selectable)

Extensions

- detect faces whose texture patch is “linear” with the values on the vertices, use an **hybrid mesh encoding**
  - non-linear faces have texture coordinates to a text. patch;
  - linear faces are defined with per-vertex coded detail (color, normal)

- improve **multiresolution** management
Eurographics ‘99 Conference

“Bringing to new life our Cultural Heritage”

Milano (Italy), Sept. 7-11, 1999

Deadlines

- Tutorials
  Nov. 15th, 1998
- State of the Art Reports
  Nov. 15th, 1998
- Papers
  Jan. 15th 1999