Spherical Parameterization and Remeshing

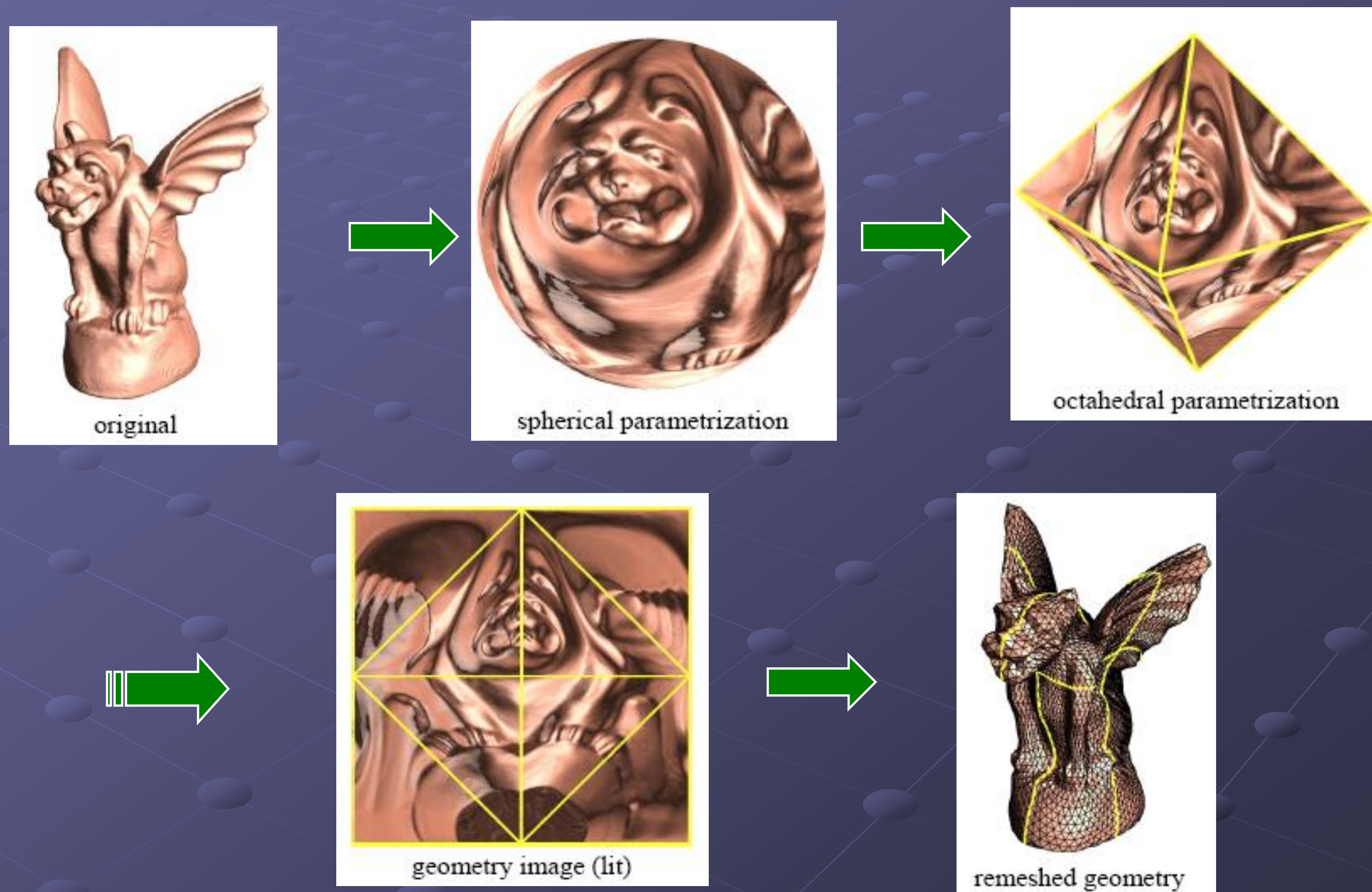
Emil Praun

University of Utah

Hugues Hoppe

Microsoft Research

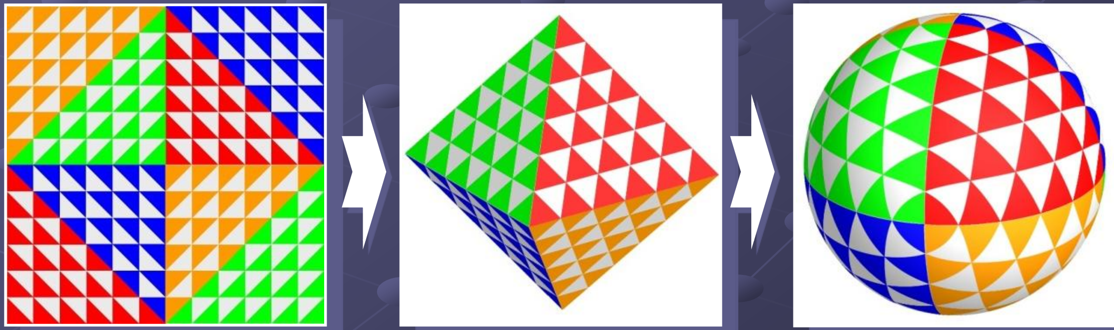
Introduction



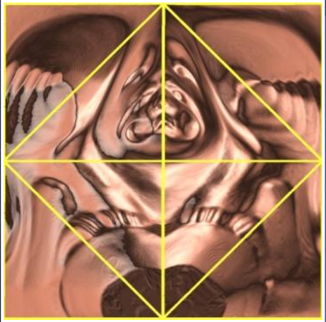
Outline

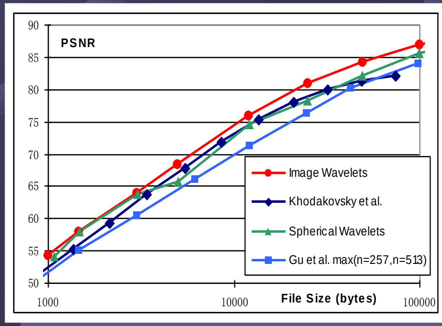
1. Spherical parameterization
   S→M

2. Spherical remeshing
   D→S
   I→D



3. Results & applications





Spherical parameterization (S→M)

- Previous work:
  - Kent et al.[1992] : simulate a balloon inflation process
  - Alexa[2002] uses spring-like relaxation process
  - Grimm[2002] : surfaces->6 charts->cube->sphere
  - Haker et al.[2000] find conformal approximation of meshes over sphere
  - Sheffer et al.[2003] find the angles of a spherical embedding as a constrained nonlinear system
  - Gostman et al.[2003] embed simple meshes on the sphere by solving a quadratic system
  - Quicken et al.[2000] parameterize the surface of a voxel volume onto a sphere.

These prior schemes cannot parameterize a complex mesh robustly and with low scale-distortion necessary for good remeshing.

Spherical parameterization (S→M)

Planar-domain Stretch Metric:

$$\phi : D \rightarrow M$$

$$\Gamma, \gamma : \text{singular value of Jacobian matrix } \mathrm{J}_{\phi}$$

Represent the largest and smallest local stretch.

rms:

$$L^2(s,t) = \sqrt{\tfrac{1}{2}\left(\Gamma^2 + \gamma^2\right)} \quad \text{and} \quad L^\infty(s,t) = \Gamma$$

L2-stretch norm :

$$L^2(M) = \sqrt{\frac{1}{A_M} \iint_{(s,t)\in D} \left(L^2(s,t)\right)^2 \, dA_M(s,t)}$$

$dA_M(s,t) = \gamma \Gamma \, ds\, dt$ is the differential surface area.

L2 stretch efficiency:

$$(A_M / A_D)\,(1/L^2(M)^2)$$

From 0 to 1

Spherical parameterization (S→M)

- Spherical-domain stretch metric:

$\phi : S \to M$

Γ, γ : singular value of Jacobian matrix $J_{\phi^{-1}}$

T : a triangle of mesh M

$$L^2(T) = \sqrt{\frac{1}{A_{M_T}} \iint_{(s,t)\in T} \left(\frac{1}{\gamma^2} + \frac{1}{\Gamma^2} \right) dA_{M_T}(s,t)} \ ,$$

$dA_{M_T}(s,t) = ds\,dt$ is the differential mesh triangle area.

Spherical parameterization (S→M)

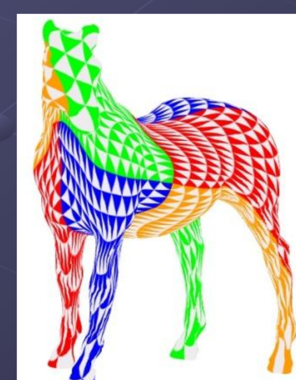
- To prevent an over-sampling problem from stretch-metric, add a penalty for **inverse-stretch**

$\Gamma \rangle\rangle \gamma$

$$\varepsilon(\mathrm{A}_{\mathrm{M}}/4\pi)^{p/2+1}(\Gamma)^{p}\ ,$$

$\varepsilon = 0.0001, p = 6$ work well for all of tested models.


without



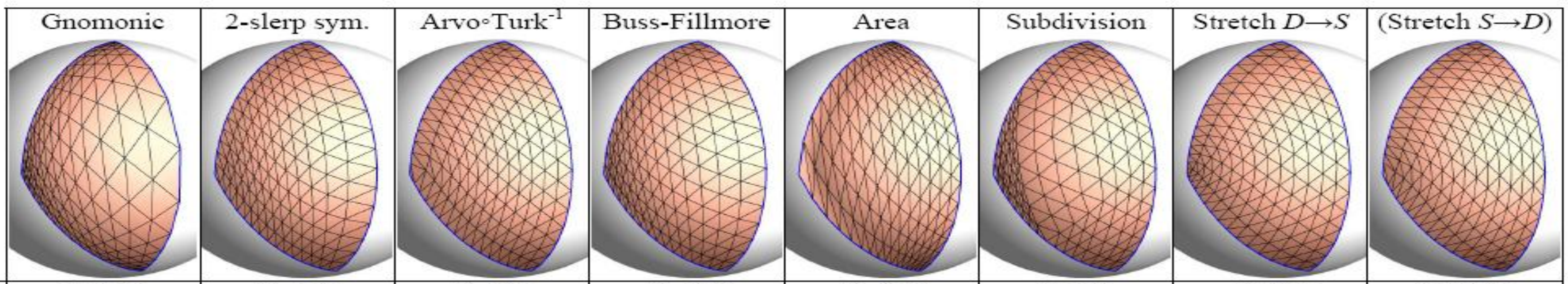

with

Spherical parameterization (S→M)

- Spherical triangle map:

  For performance, they have chosen to use the gnomonic map for the coarse-to-fine optimization of S->M

Gnomonic: It is simply spherical projection about the sphere center O. That is, $P = (\alpha A+\beta B+\gamma C) / \|\alpha A+\beta B+\gamma C\|$. The inverse is easily computable as spherical projection back onto the triangle.



	Gnomonic	2-slerp sym.	Arvo∘Turk^{-1}	Buss-Fillmore	Area	Subdivision	Stretch $D\to S$	(Stretch $S\to D$)
$D\to S$	L^2=0.31	L^2=0.67	L^2=0.75	L^2=0.70	L^2=0.40	L^2=0.71	L^2=0.85	(L^2=0.80)
$S\to D$	L^2=0.23	L^2=0.55	L^2=0.73	L^2=0.45	L^2=0.42	L^2=0.32	L^2<0.01	(L^2=0.75)

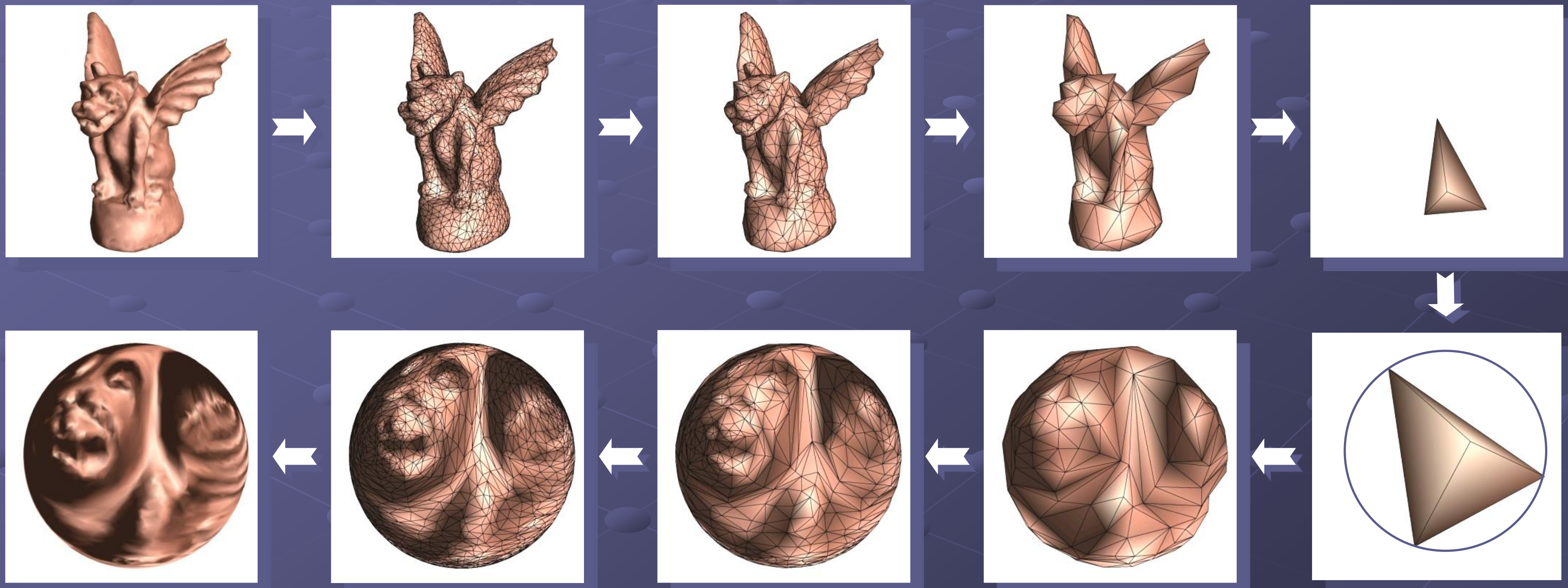
Figure 3: Comparison of spherical triangle maps for a large spherical triangle, and computed stretch efficiencies in both map directions. (The black curves show a uniform tessellation of the planar triangle in domain D mapped onto the sphere S.)

Spherical parameterization (S→M)

- Algorithm:
  1. **Coarse-to-fine strategy**:
  - simplify mesh M to a tetrahedron-base domain with progressive mesh (PM),
  - map the base domain to the sphere,
  - traverse the PM sequence backward and insert vertices on the surface

Coarse-to-Fine Strategy

Convert to progressive mesh



Parameterize coarse-to-fine

Maintain embedding & minimize stretch

Spherical parameterization (S→M)

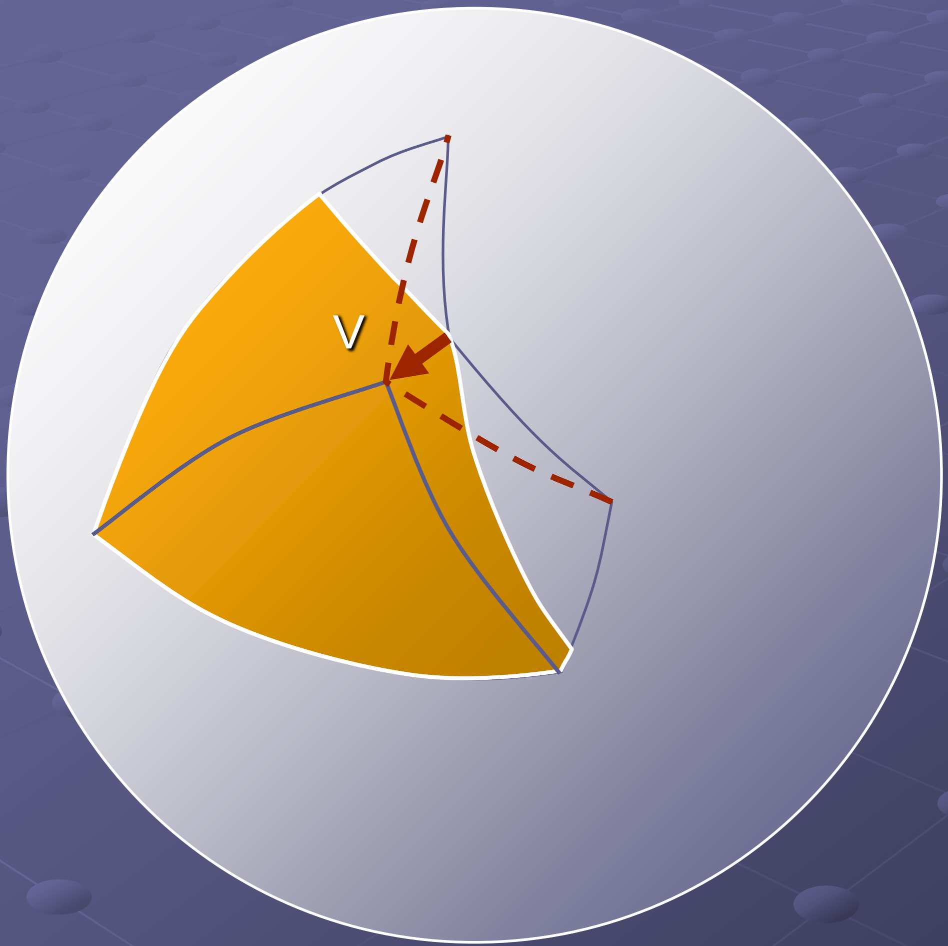
- Algorithm:

  2. **Vertex insertion:**
     - new vertex has the 1-ring neighbors form a spherical polygon
     - the kernel of this spherical polygon is defined as the intersection of hemispheres, each of which defined by each of the polygon edges
     - new vertex can only be placed in this kernel

Spherical parameterization (S→M)

- Algorithm:
  3. **Vertex optimization:**
     - After insert a new vertex, optimize all vertices in their neighborhood one at a time
     - Each optimization using the stretch metric summed from adjacent triangles, perturbing the vertex only in the kernel of its 1-ring
     - All vertices traversed by a priority queue ordered by the amount of change in their neighborhood, and stop when the largest change is below a threshold

Vertex Insertion

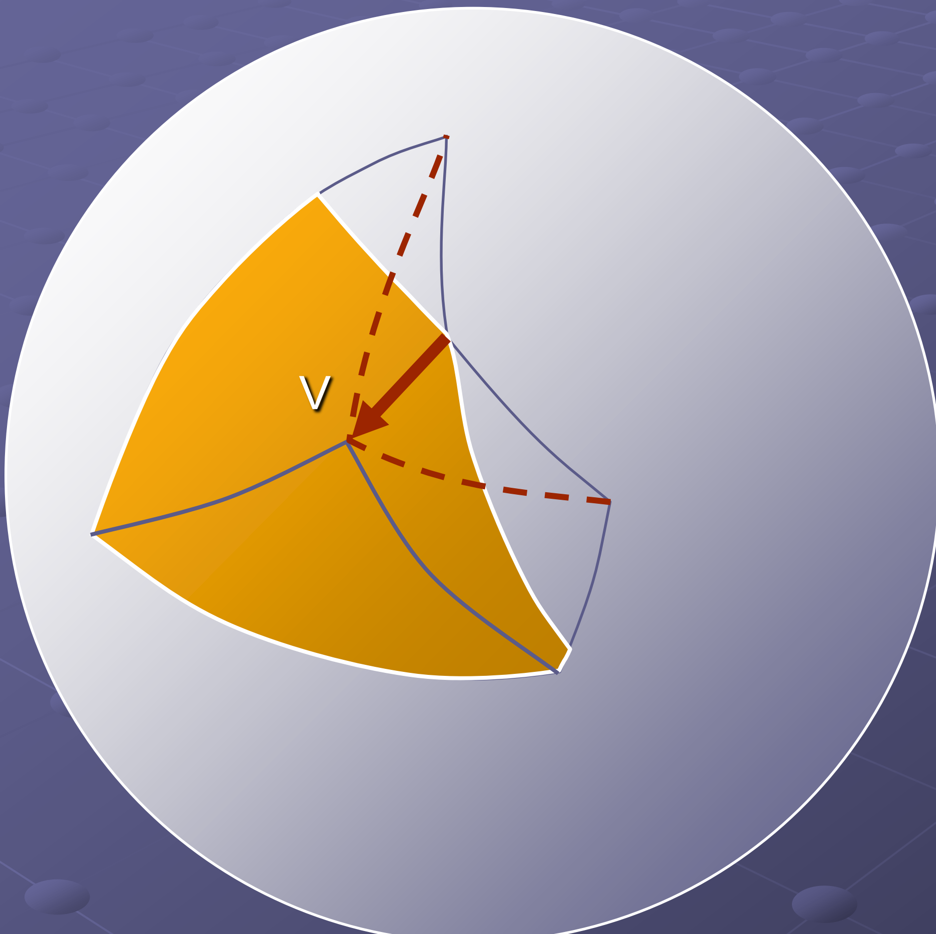


- Before Vsplit:
  - No degenerate/flipped Δ

  ⇒ 1-ring kernel ≠∅
- Apply Vsplit:

  No flips if V inside kernel

Vertex Optimization



- Before Vsplit:
  - No degenerate/flipped Δ

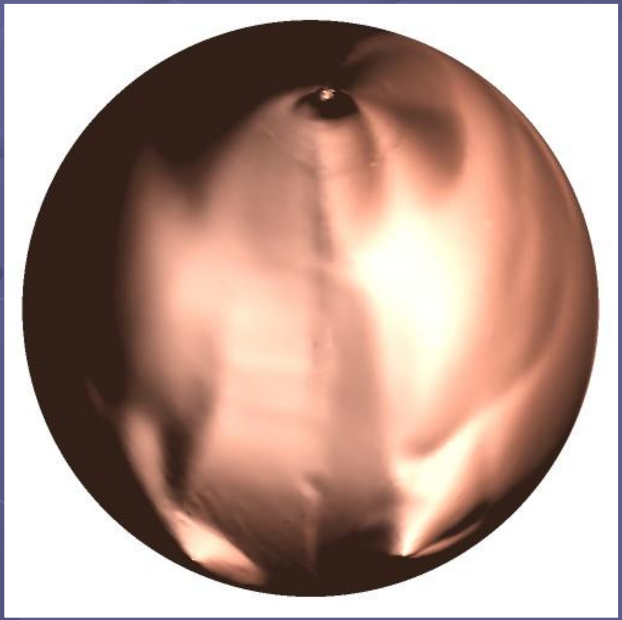
  $\Rightarrow$ 1-ring kernel $\neq \emptyset$
- Apply Vsplit:

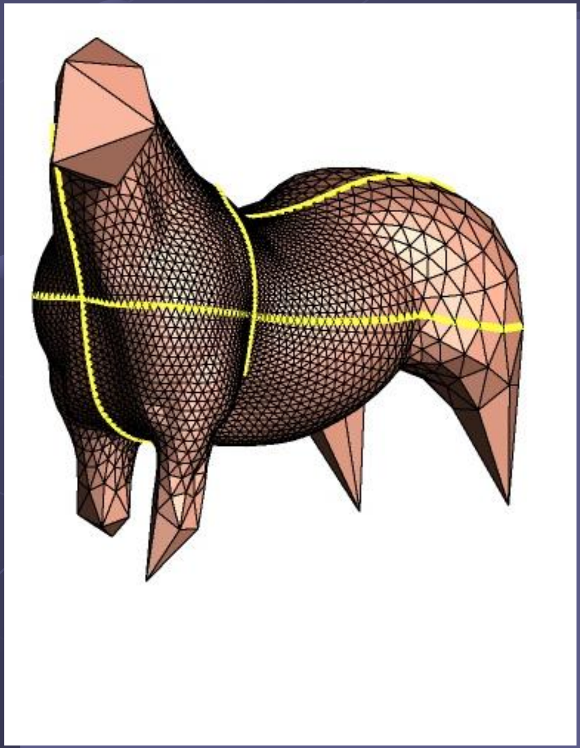
  No flips if V inside kernel
- Optimize stretch:

  No degenerate Δ
  (they have ∞ stretch)

Traditional Conformal Metric

- Preserve angles but “area compression”
- Bad for sampling using regular grids

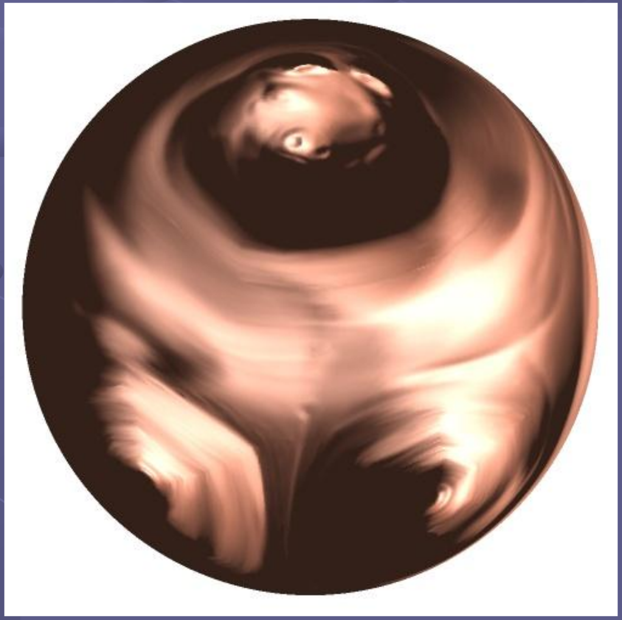


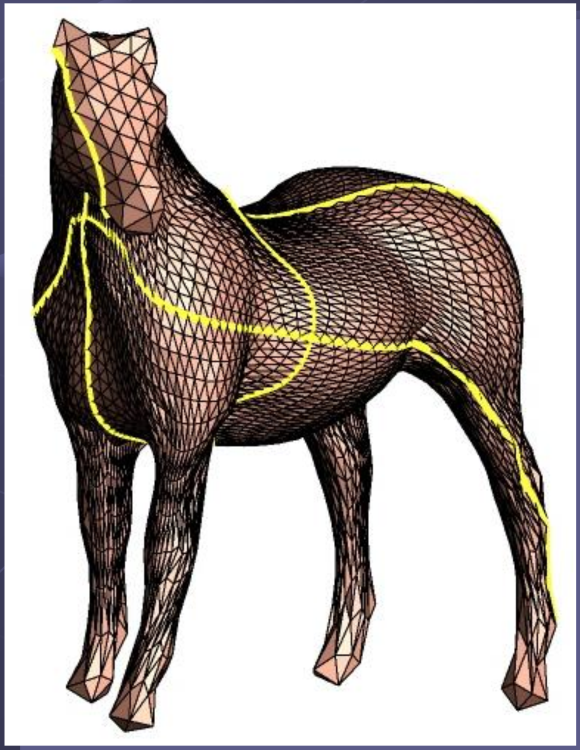


Stretch Metric

[Sander et al. 2001]
[Sander et al. 2002]

- Penalizes undersampling
- Better samples the surface





Domain Spherical parameterization (D→S)

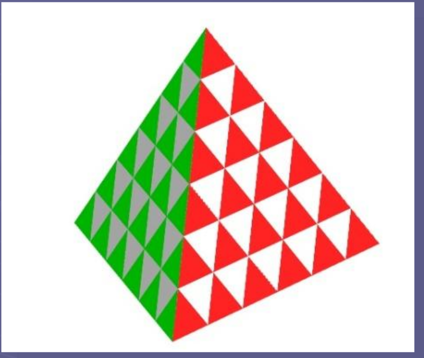
- Use stretch-optimized spherical triangle map on the n-tessellated domain D
- Compare for other maps:

	Stretch-optimiz.	Procedural Maps					
		Gno-monic	2-slerp	Arvo∘ Turk^{-1}	Buss-Fillmore	Area	Subdiv.
tetra	**0.910**	0.628	0.871	0.846	**0.889**	0.849	0.645
octa	**0.969**	0.893	0.954	0.943	0.958	**0.958**	0.902
cube2	**0.983**	0.859	0.945	0.967	0.953	0.932	0.924
cube4		0.956	0.965	0.966	0.966	**0.978**	0.959
cube8		0.961	0.966	0.966	0.966	0.965	0.964
flat-octa	**0.896**	-	-	-	-	-	-

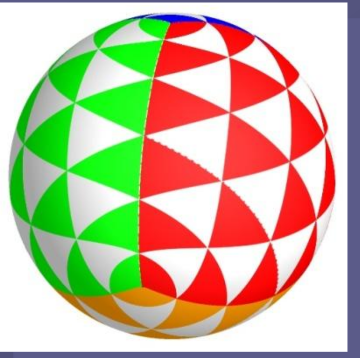
Table 2: L^2 stretch efficiencies for $D \rightarrow S$ using optimization and procedural maps. (Cube faces are split into 2, 4, or 8 triangles.)

Domains And Their Sphere Maps

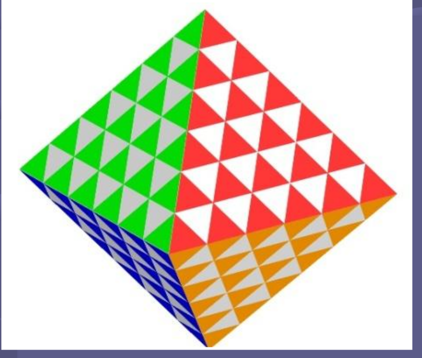
tetrahedron



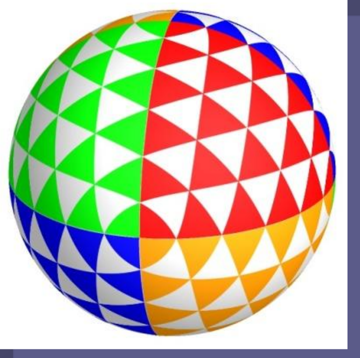
→



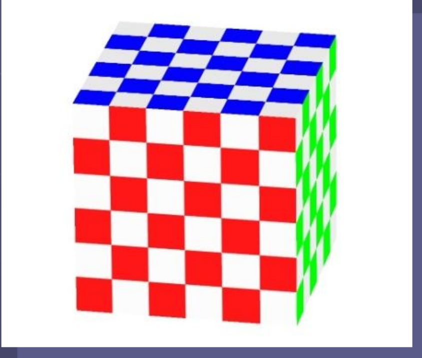
octahedron



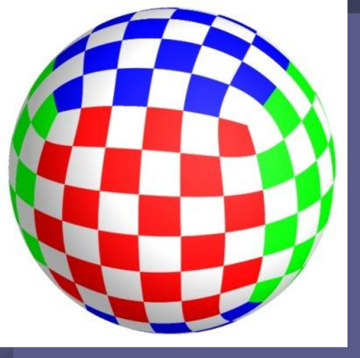
→



cube

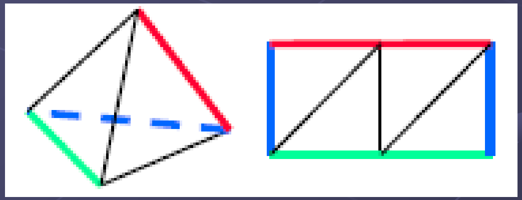


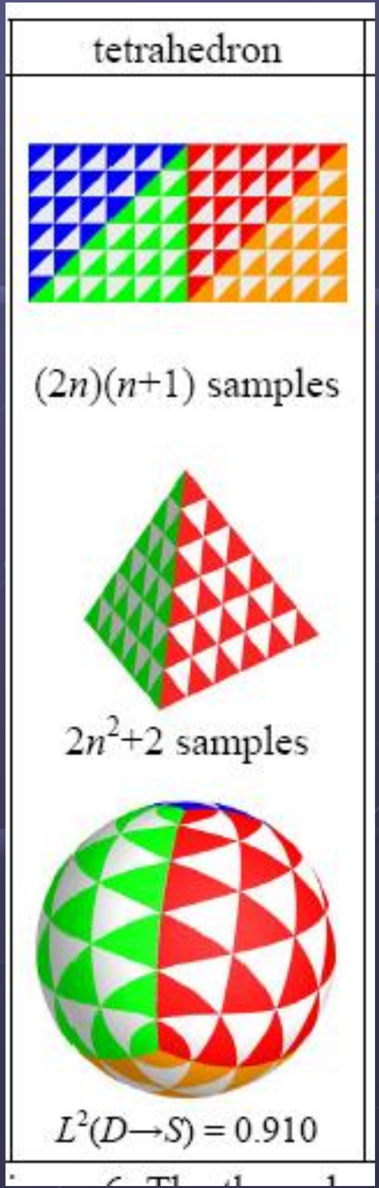
→



Domain unfolding (I→D)

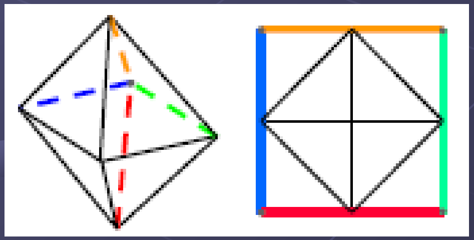
- Tetrahedron:
  - Non-isometric
  - Rectangular image
  - Linear interpolation

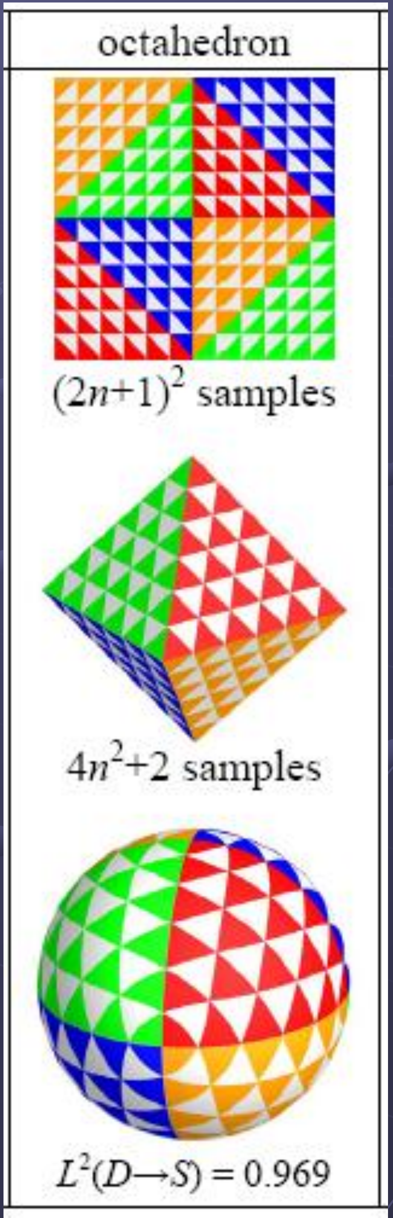




Domain unfolding (I→D)

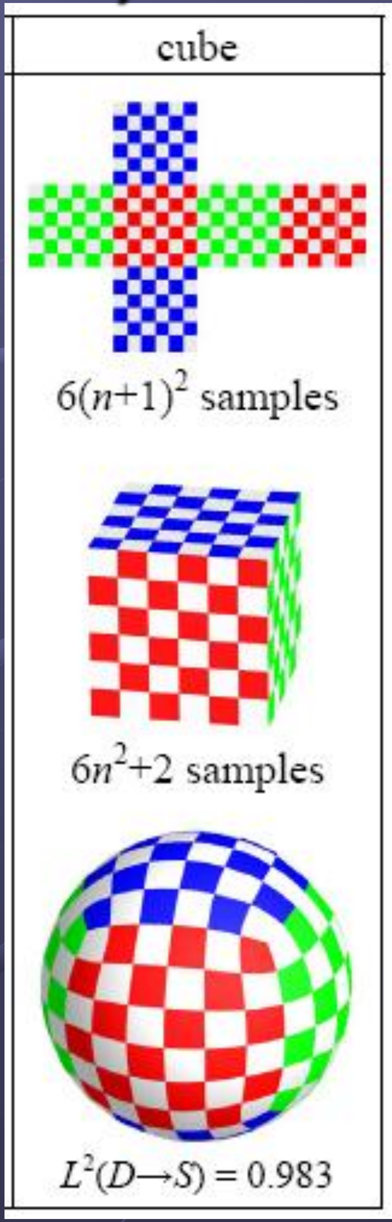
- Octahedron:
  - Non-isometric
  - Square image
  - Linear interpolation





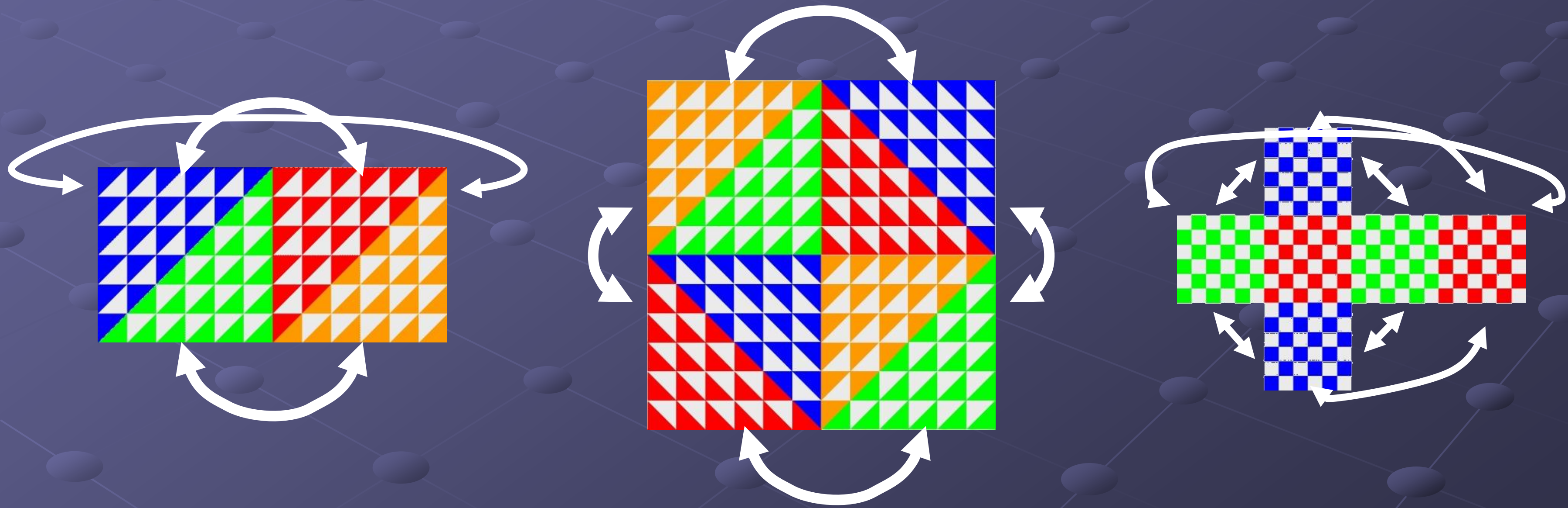
Domain unfolding (I→D)

- Cube:
  - Isometric
  - Bi-linear interpolation



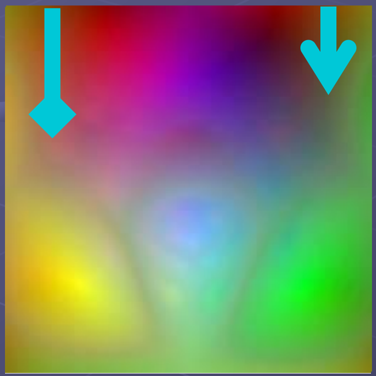
Domain Unfolding (I→D)

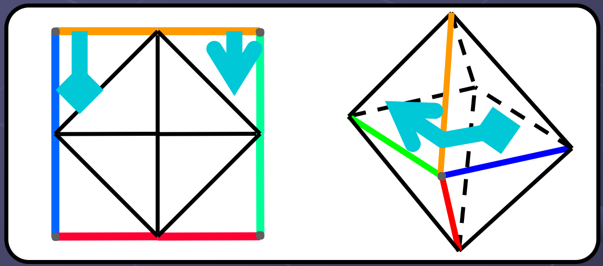
Boundary Constraints



Domain Unfolding (I→D)

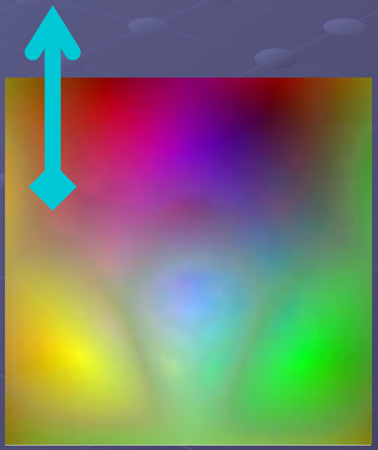
Boundary extension rules

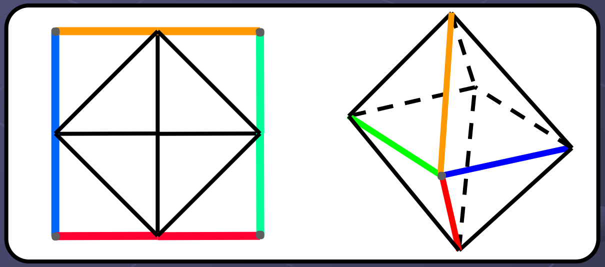




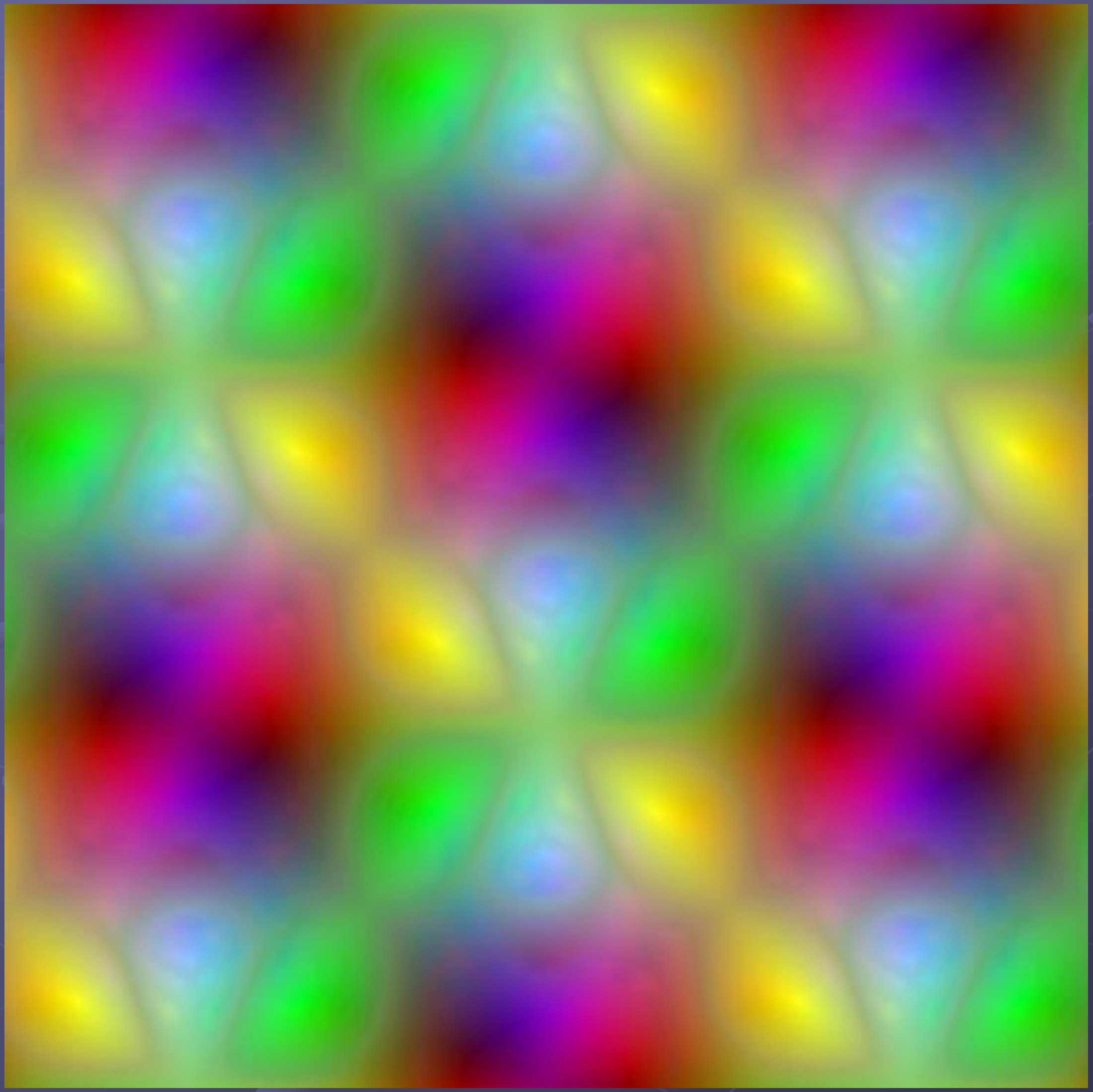
Domain Unfolding (I→D)

Boundary extension rules

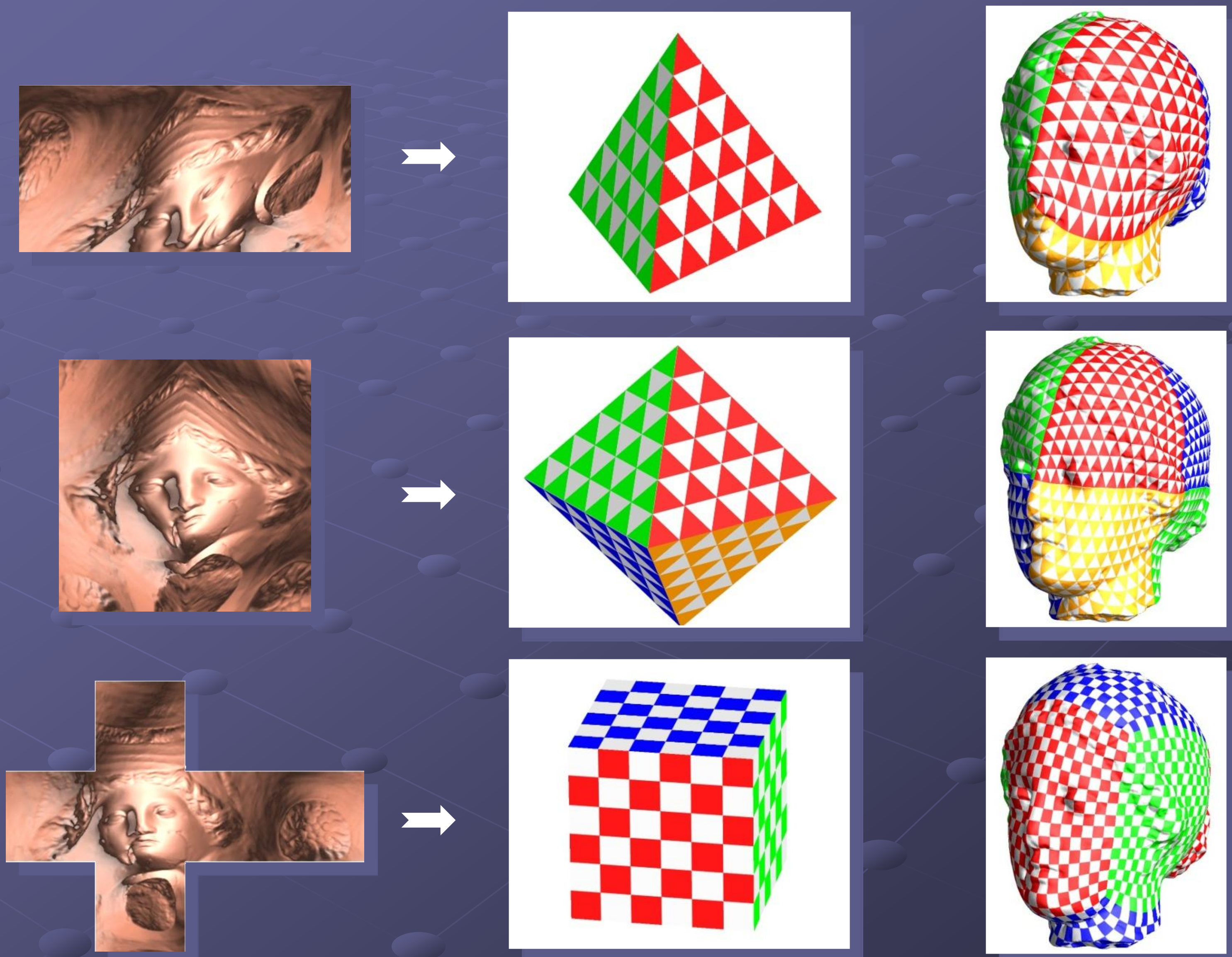




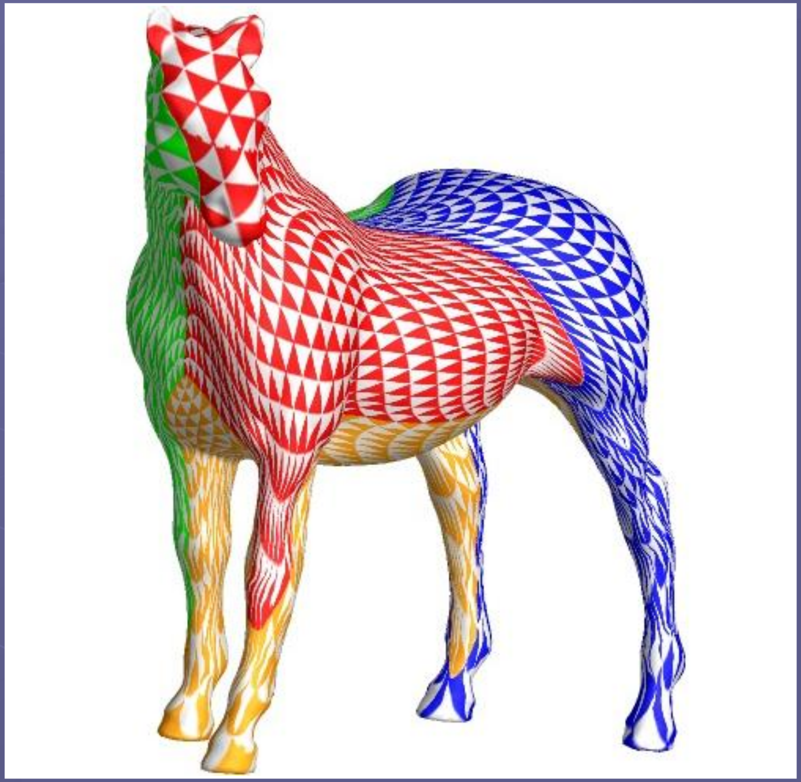
Boundary extension rules

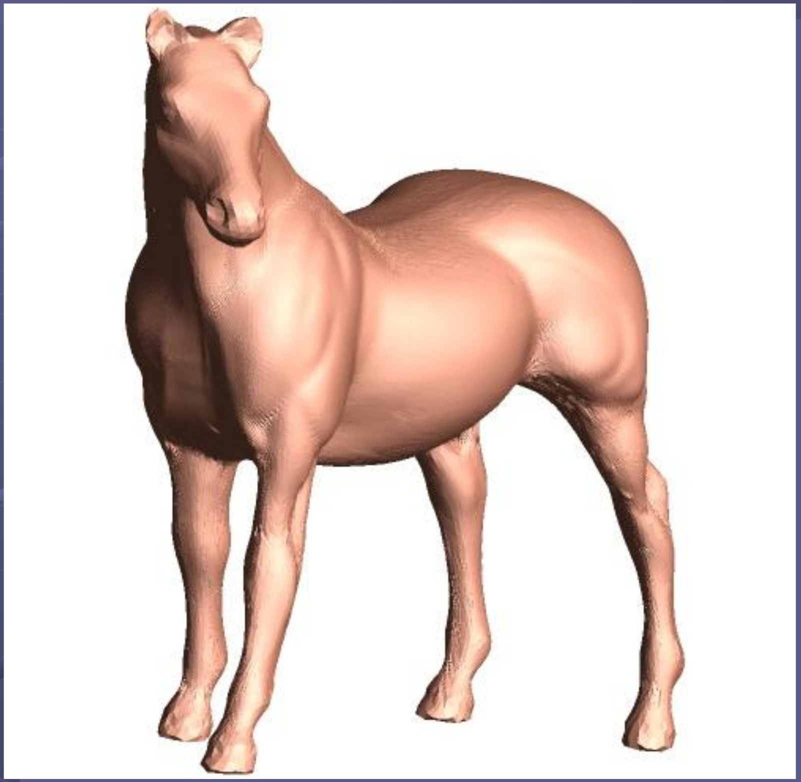


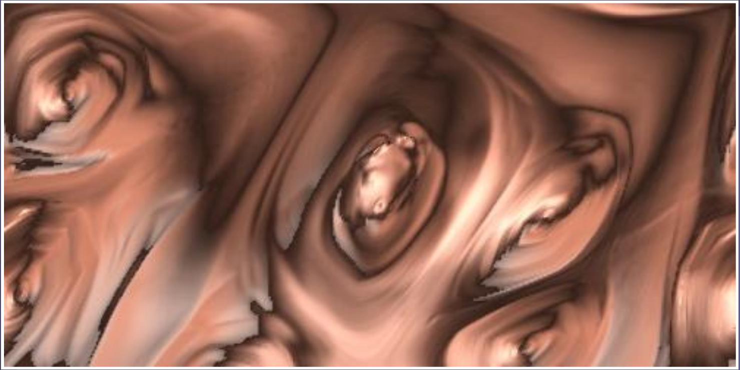
Results



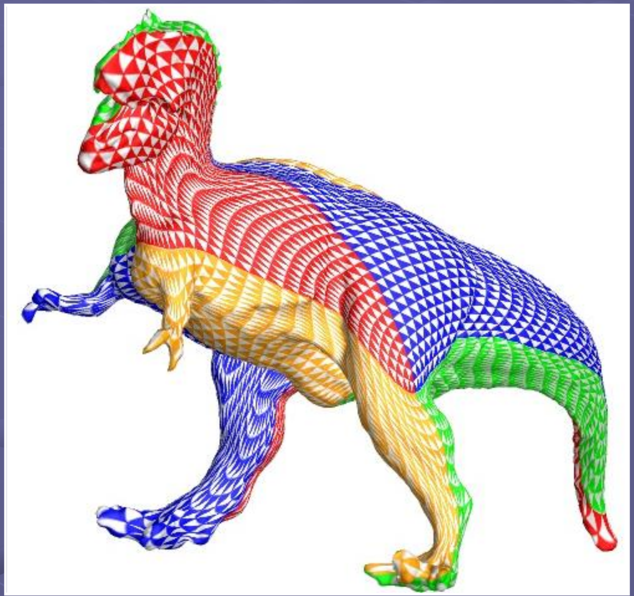
Results

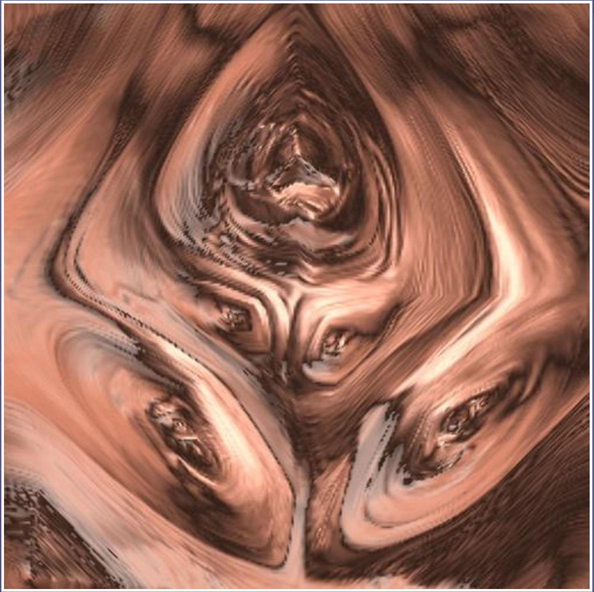


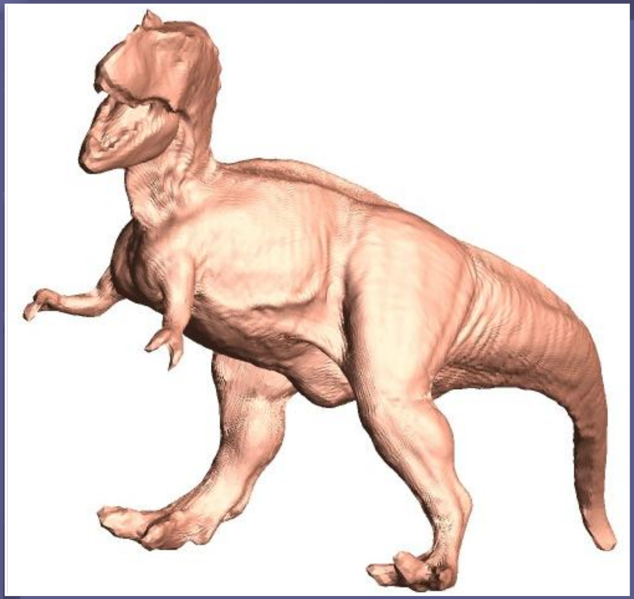




Results

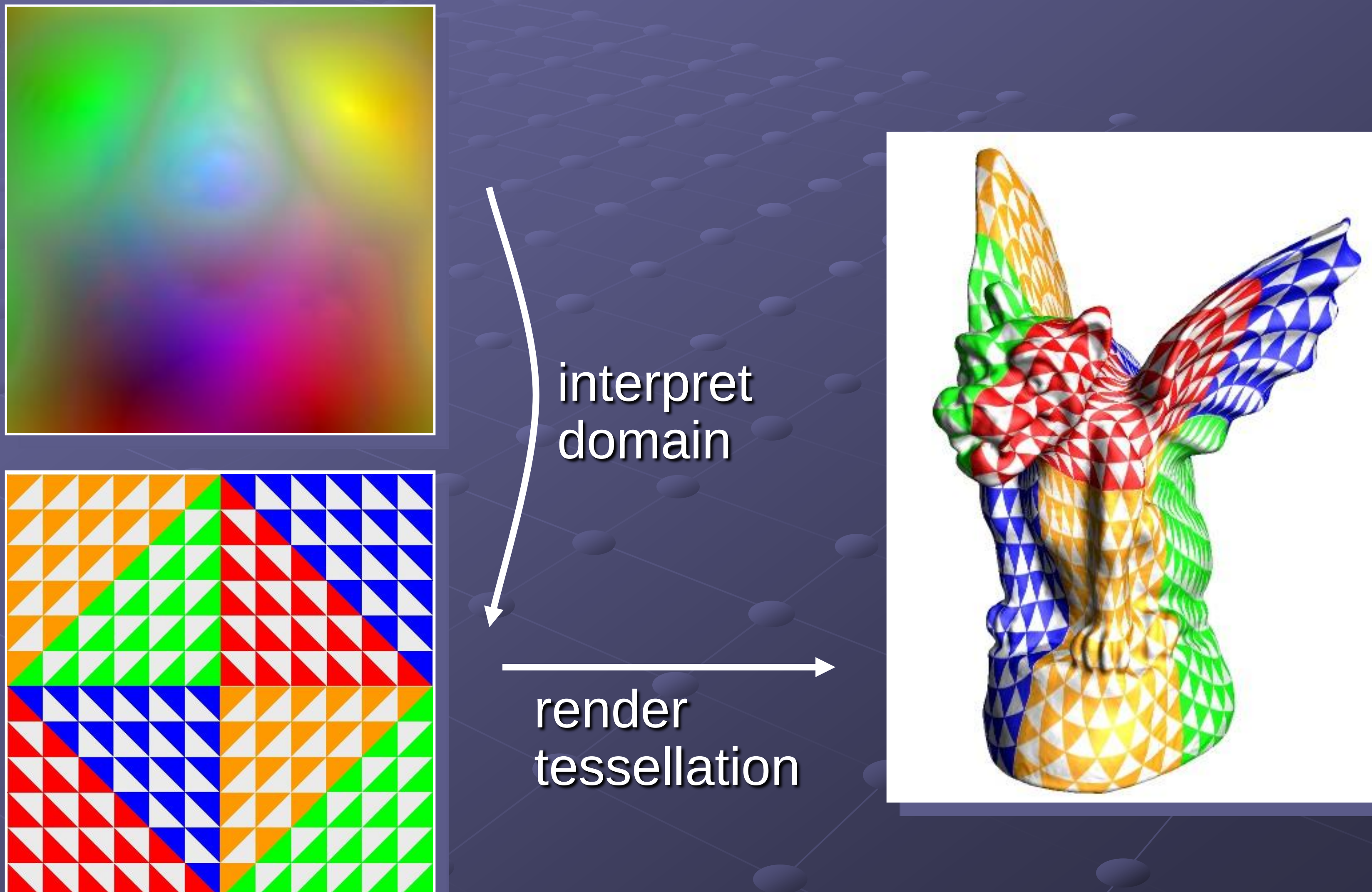






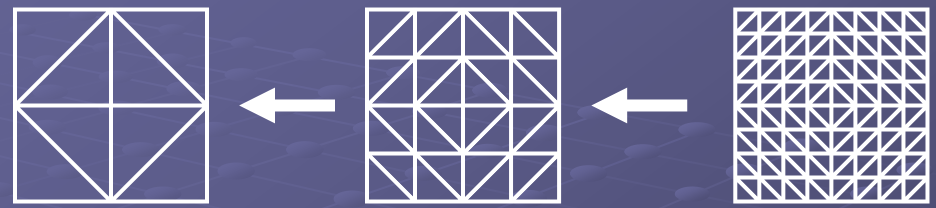
Applications

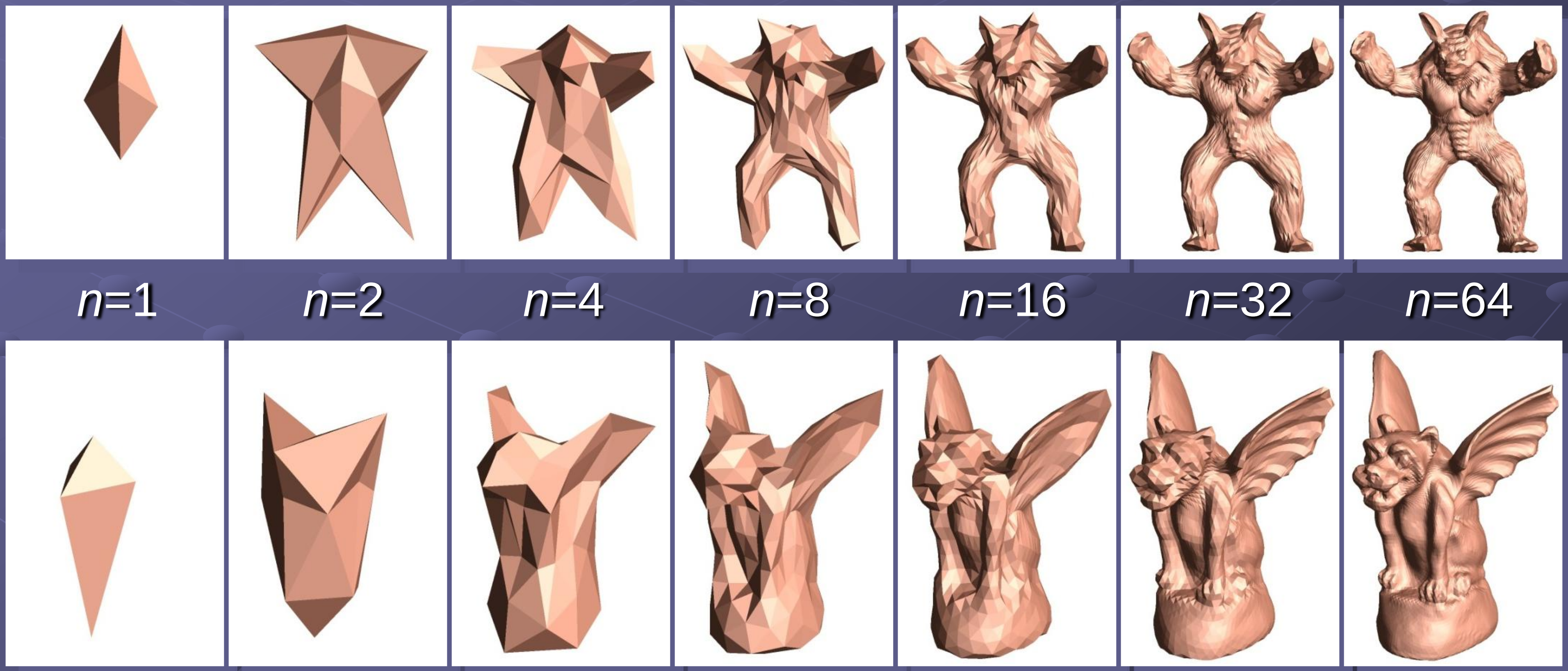
- Rendering



Applications

- Level-of-Detail

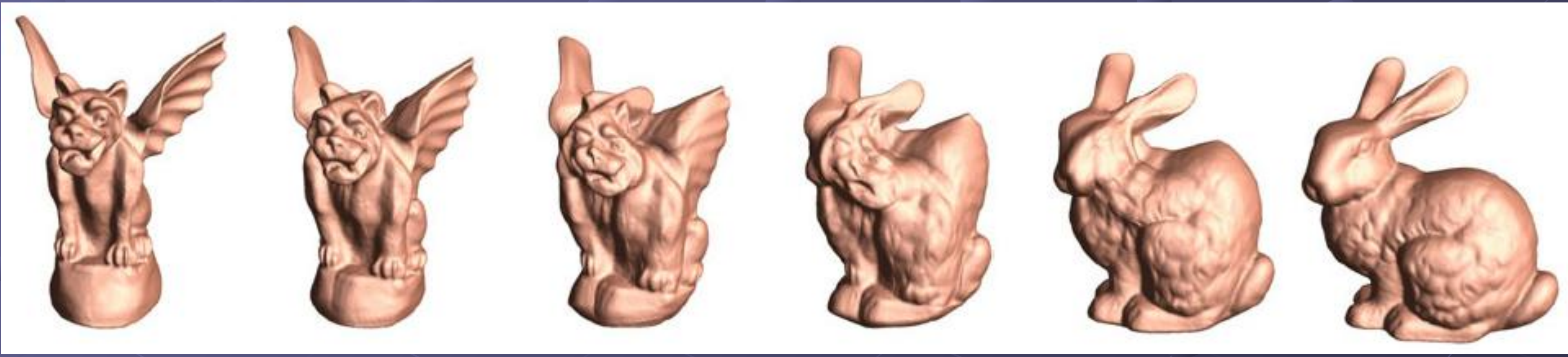




Applications

- Morphing

  Interpolating 2 geometry images



Applications

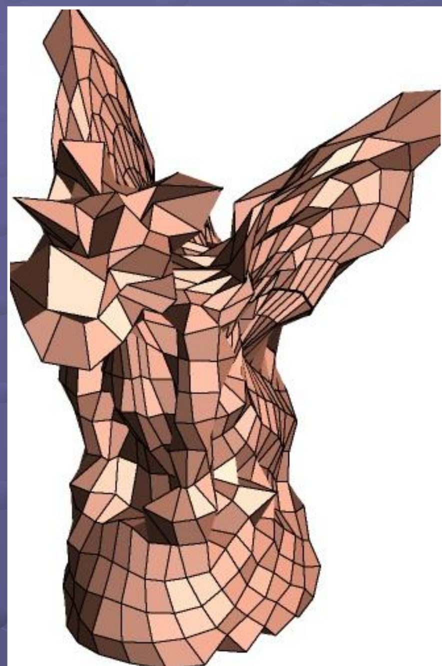
- Geometry Compression

12 KB 3 KB 1.5 KB

Applications

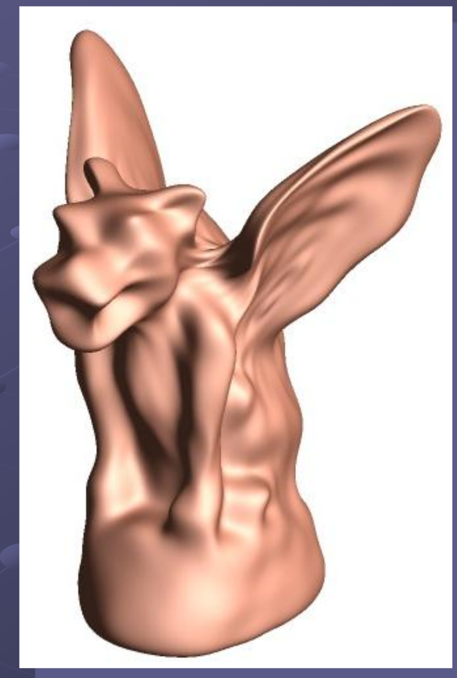
- Smooth Subdivision

[Losasso et al. 2003]



GPU

3.17 ms



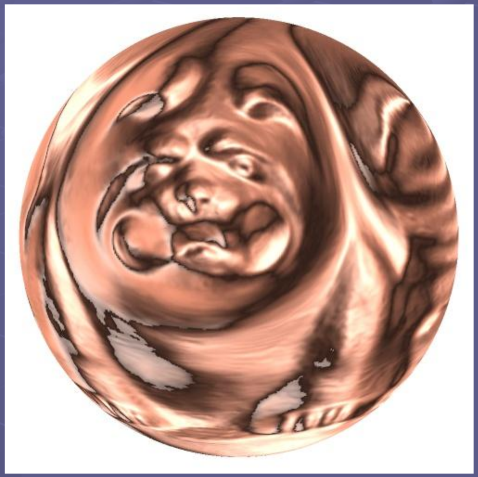
33x33 geometry image

C^1 surface

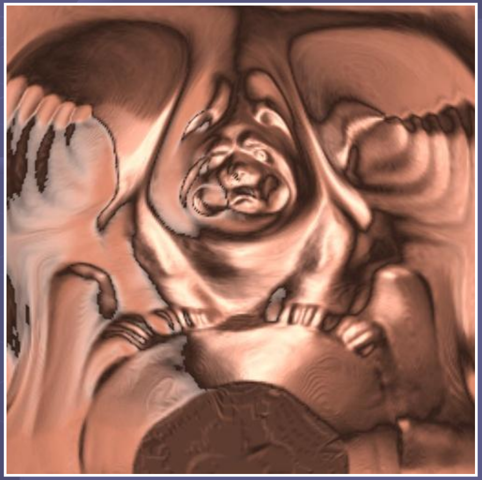
ordinary uniform bicubic B-spline

Summary

original



spherical parametrization



geometry image

remesh

Conclusions

- Spherical parametrization
  - Guaranteed one-to-one
- New construction for geometry images
  - Specialized to genus-0
  - No *a priori* cuts $\Rightarrow$ better performance
  - New boundary extension rules
    - Effective compression, DSP, GPU splines, …

Future Work

- Explore DSP on unfolded octahedron
  - 4 singular points at image edge midpoints
- Fine-to-coarse integrated metric tensors
  - Faster parametrization; signal-specialized map
- Direct $D \leftrightarrow S \leftrightarrow M$ optimization
- Consistent inter-model parametrization