PROJECTIVE DISPLACEMENT MAPPING FOR RAYTRACED EDITABLE SURFACES

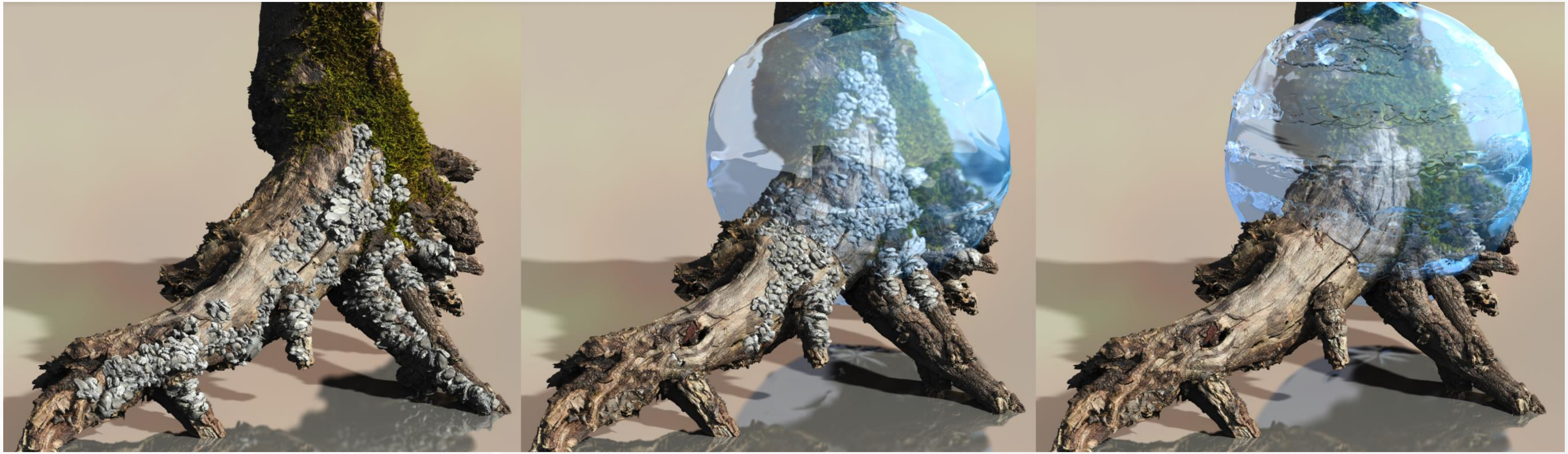
Rama Carl Hoetzlein

Professor of Interaction Design | California College of the Arts
Founder | Quanta Sciences

Artists often seek **rapid iteration** combined *with* **high quality feedback.**

GOAL

2025 TAIPEI



Authoring of detailed surfaces with ray traced quality, at low compute & memory cost.

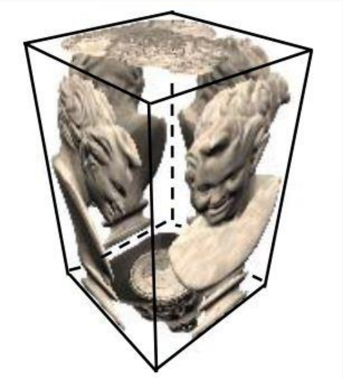
- Edit Detailed surfaces
- Real-time Feedback
- Ray traced quality
- Low memory & disk usage

BACKGROUND

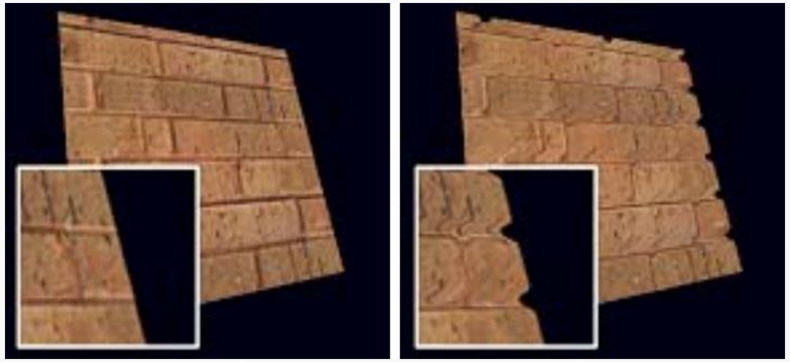
BACKGROUND

RASTER-BASED SAMPLING

TEXTURE-SAMPLING METHODS



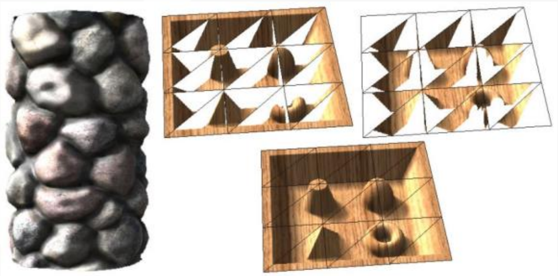
2000, Oliveira et al.,
Relief texture mapping



2001, Kaneko, Detailed shape representation with parallax mapping

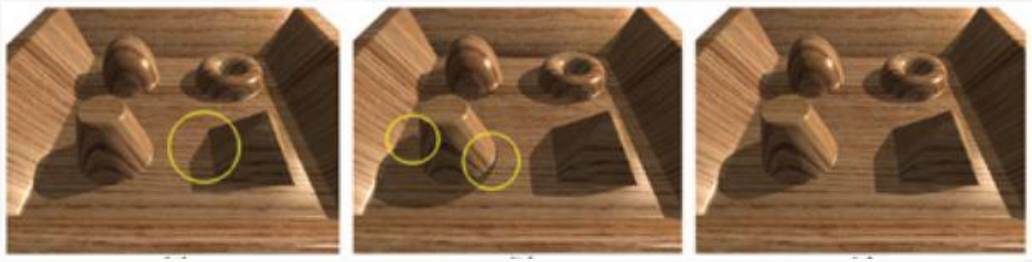
2005, Policarpo et al., Real-Time Relief Mapping on Arbitrary Polygonal Surfaces

RAY-STEPPING METHODS



2005, Tatarchuk, Practical dynamic parallax occlusion mapping

2007, Dachsbacher et al. Prism Parallax Occlusion Mapping with Accurate Silhouette Generation



2007, Policarpo et al.,
Relaxed Cone Stepping for Relief Mapping

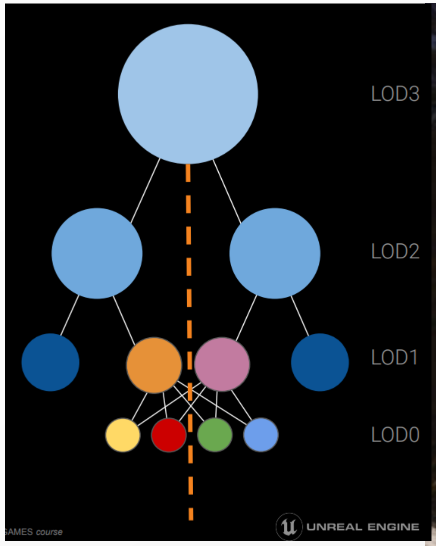
Benefits: ray stepping gives self-shadows

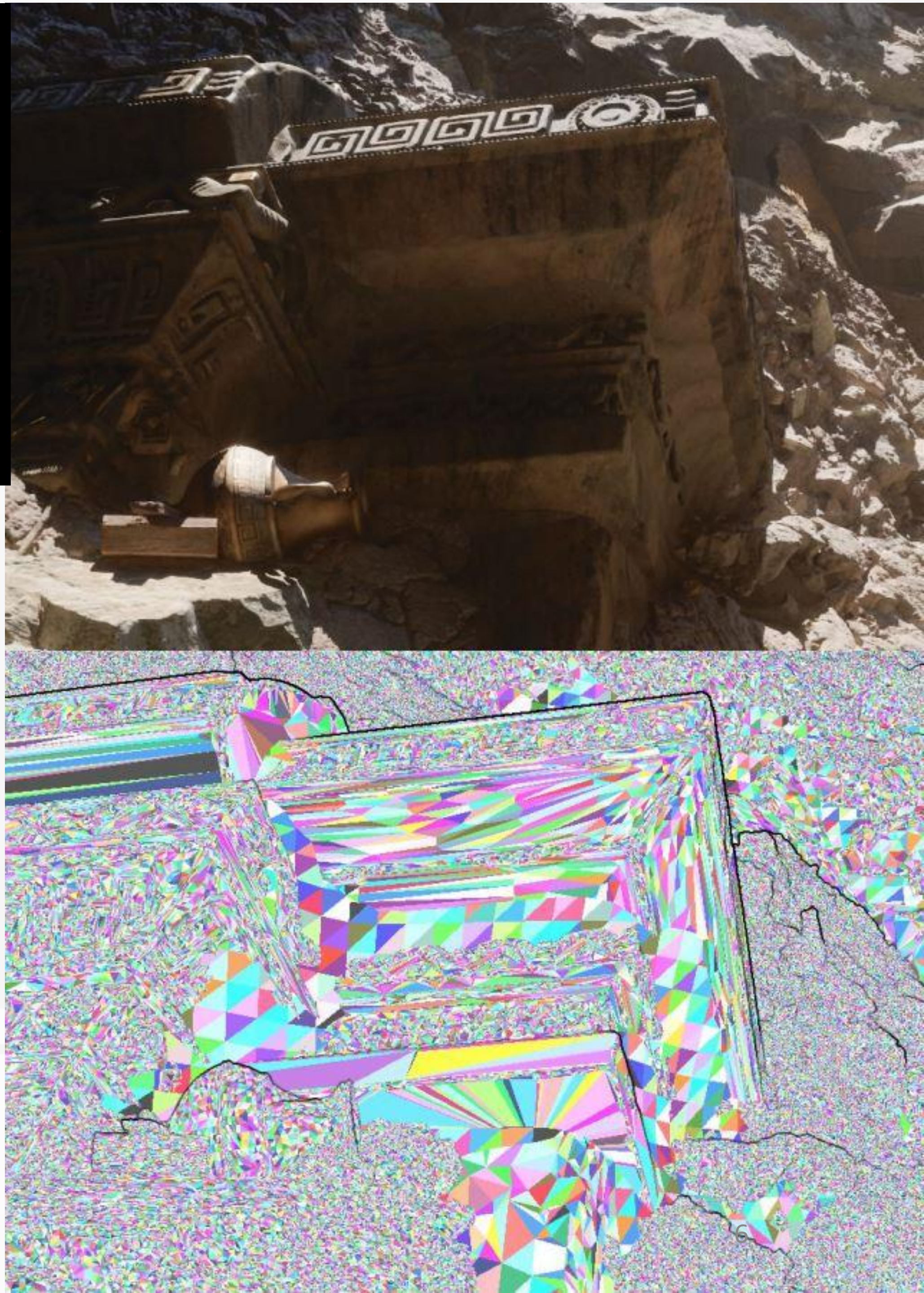
Limitations: only PPOM gives correct silhouettes, all are view dependent

BACKGROUND

VIRTUALIZED GEOMETRY

Unreal Engine™ Nanite





Hierarchical LOD splitting and rendering by partitioning large meshes into streamable clusters.

Benefits: real-time, high res meshes, pixel-level detail

Limitations: not easily editable. high mem & disk usage, even when compressed. e.g. 433M tri -> 4.6 GB

2021, Brian Karis, Rune Stubbe, Graham Wihlidal, **Nanite: A Deep Dive**. Siggraph 2021, Unreal Engine

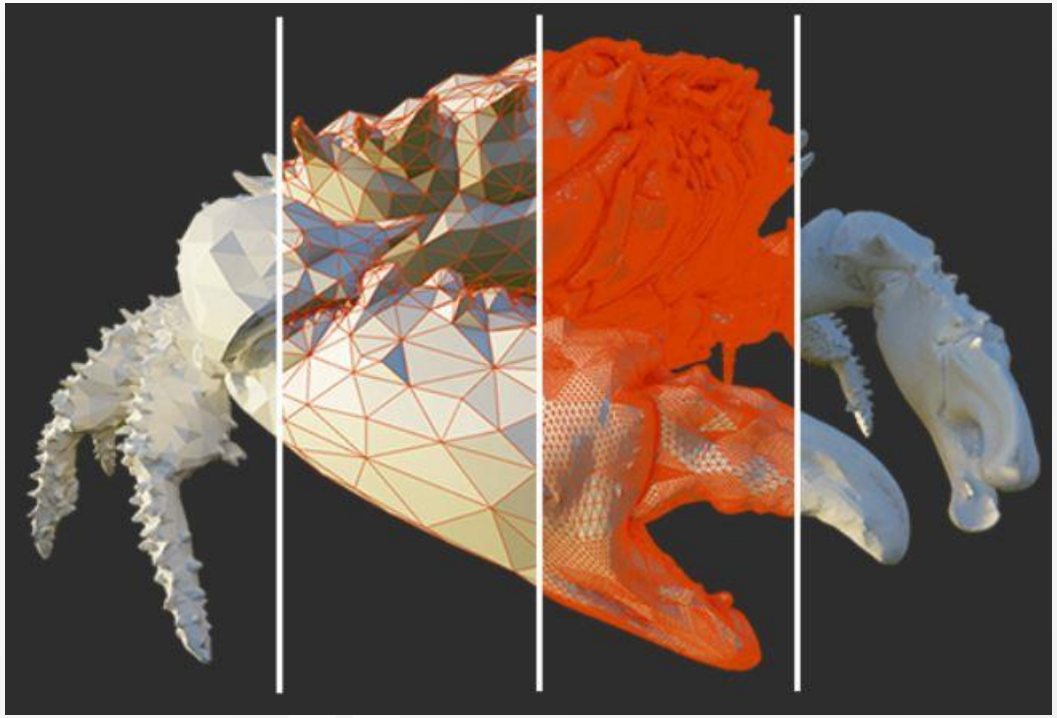
BACKGROUND

TESSELLATION

Hardware Adaptive Tessellation (Real-Time)

NVIDIA Micro-Meshes

Adaptive tessellation of detailed meshes by converting to compressed barycentric coordinate maps.



2023, Nvidia Micro-Mesh API

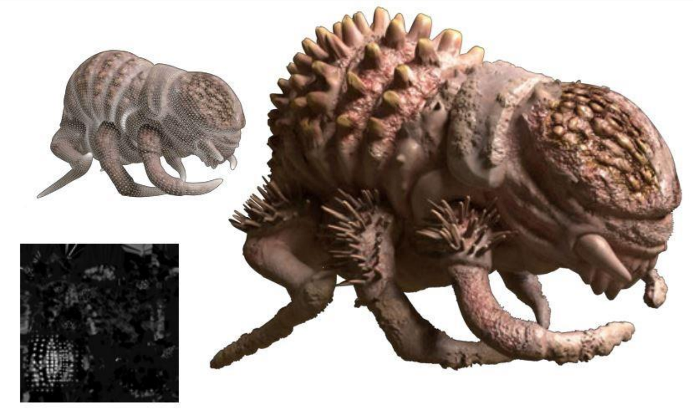
Authors: Andrea Maggiordomo, Henry Moreton, Marco Tarini, Christoph Kubisch, Pascal Gautron, Tristan Lorach, Pyarelal Knowles, Neil Bickford, Mathias Heyer, Martin-Karl Lefrancois

Benefits: real-time, adaptive, dynamic detail

Limitations: view dependent, barycentric map compression

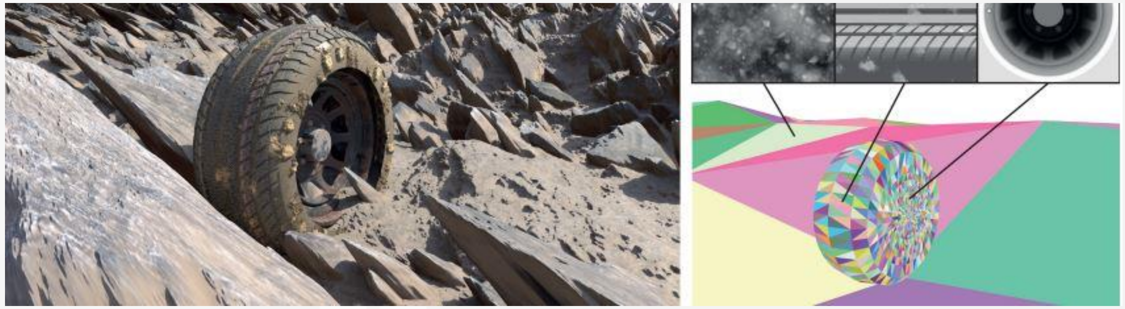
BACKGROUND

ADAPTIVE SAMPLING



D-BVH: Displacement BVH

Tessellation-Free Displacement Mapping for Ray Tracing
2021, Thonat, Beaune, Sun, Carr, Boubekeur (Adobe)



RMIP: Rectangular minmax image pyramid

RMIP: Displacement ray-tracing via inversion and oblong bounds
2023, Thonat, Georgiev, Beaune, Boubekeur (Adobe)

Introduce hierarchical, minmax data structures to accelerate direct sampling of displaced surfaces.

Benefits: tessellation-free, fast rebuild

Limitations: two types of acceleration structs, hierarchical sampling is costly

ALGORITHM DESIGN

PROJECTIVE DISPLACEMENT MAPPING (PDM)

ALGORITHM DESIGN

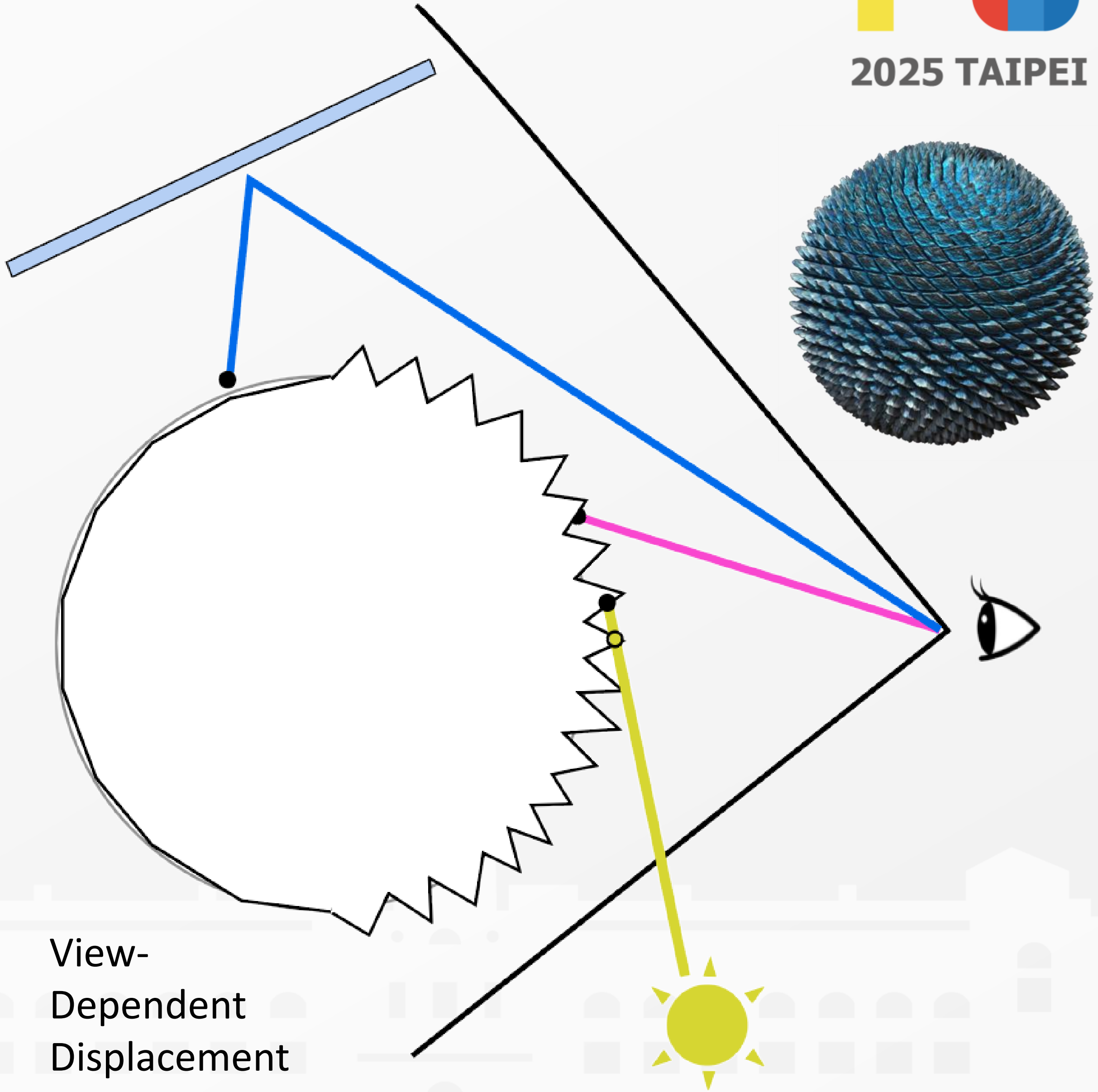
GOAL: TRUE DISPLACEMENT

View-Dependent Displacement

- Visible surfaces are tessellated / shaded based on camera.
- Only primary rays “see” detailed surface.
- Secondary rays see non-displaced detail.

True Displacement

- All rays exhibit detailed surfaces: primary rays, shadows, reflections, refractions, and path tracing



ALGORITHM DESIGN

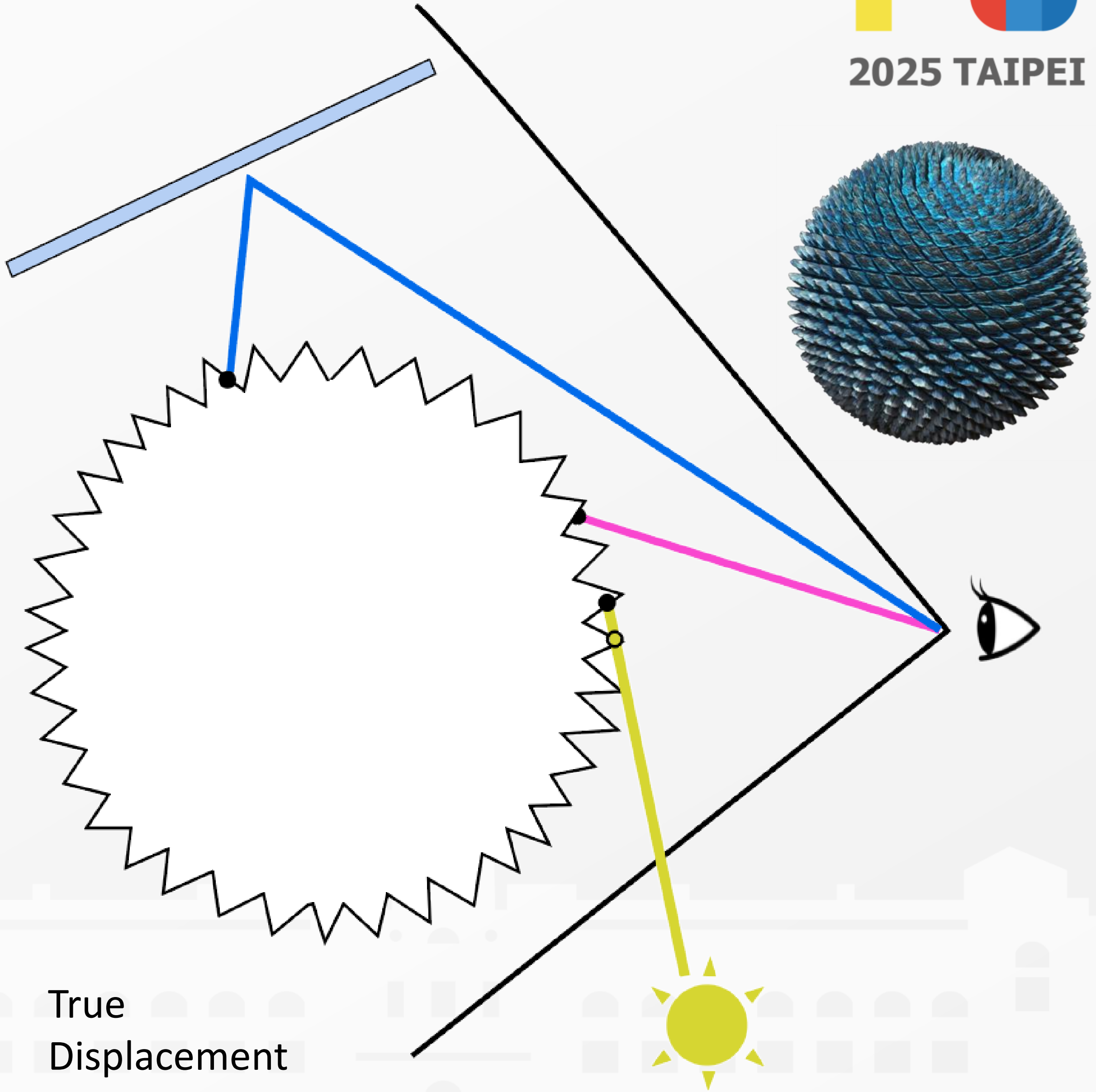
GOAL: TRUE DISPLACEMENT

View-Dependent Displacement

- Visible surfaces are tessellated / shaded based on camera.
- Only primary rays “see” detailed surface.
- Secondary rays see non-displaced detail.

True Displacement

- All rays exhibit detailed surfaces: primary rays, shadows, reflections, refractions, and path tracing



ALGORITHM DESIGN

OBSERVATIONS

1. View dependent methods prevent “true” all-ray details.
   Must use BVHs to get true displacement ray tracing.

2. Data structures usually introduce *either* sampling cost or rebuild cost.
   Can we avoid secondary data structures?

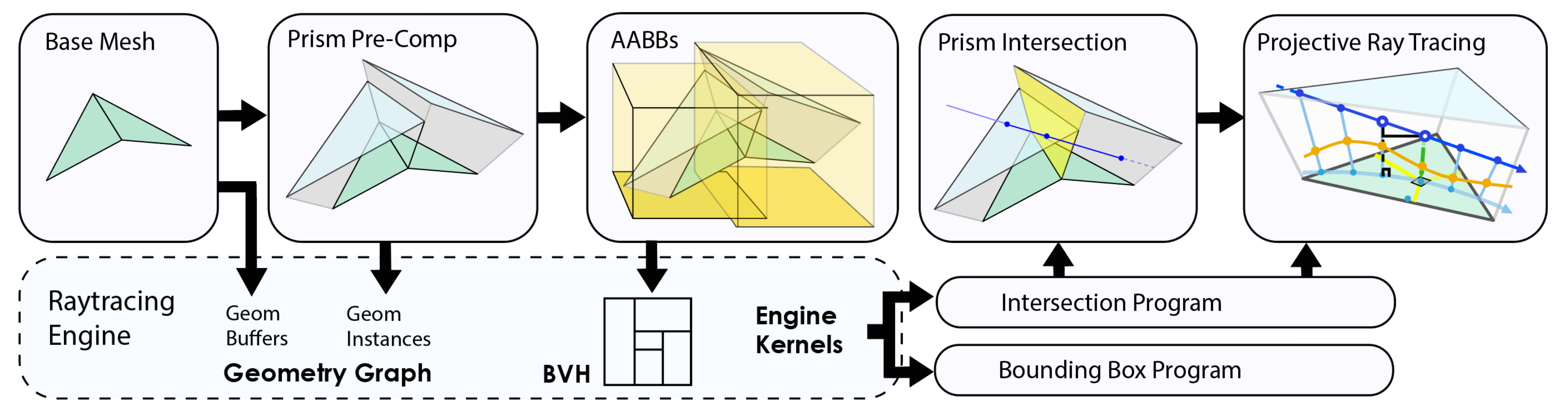
3. Performance is essential!
   Both editing and ray tracing must be <30 milliseconds/frame.

4. Can we accelerate direct sampling of displacement maps?

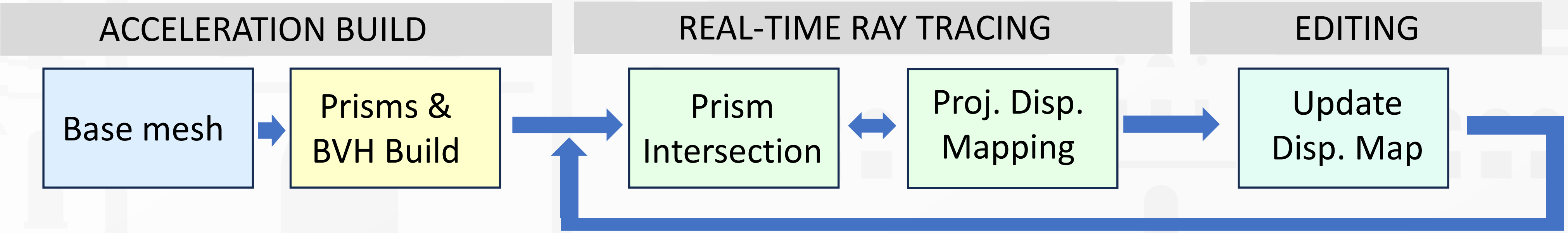
PROJECTIVE DISPLACEMENT MAPPING

2025 TAIPEI

OVERVIEW



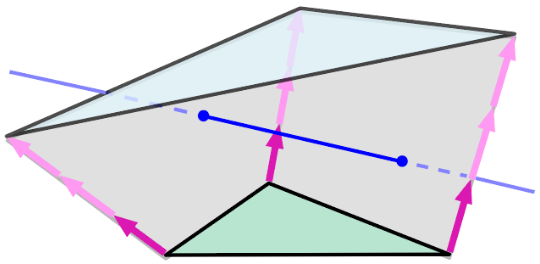
WORKFLOW



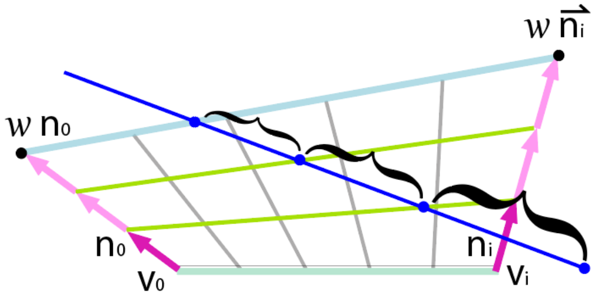
PROJECTIVE DISPLACEMENT MAPPING

DESIGN

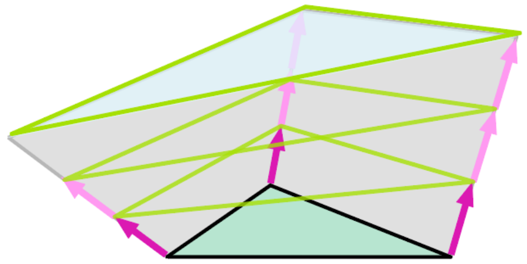
Question: Why is displacement mapping costly to ray trace?



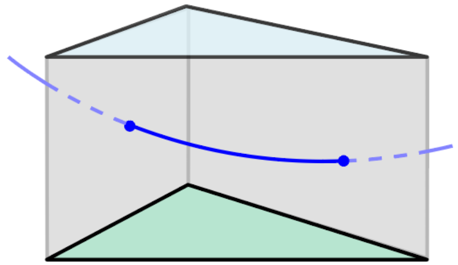
Standard Offset Prism



Triangle Construction



Non-parallel Extruded Triangles



Canonical Space (non-linear rays)

2023, Ogaki S., Nonlinear ray tracing for displacement and shell mapping

Interpolated Surface Point: $$\mathbf{P}(u,v) = uV_0 + vV_1 + (1-u-v)V_2,$$

Interpolated Surface Normal: $$\mathbf{N}'(u,v) = uN_0 + vN_1 + (1-u-v)N_2,$$

Def. Displaced Surface: $$\mathbf{S}(u,v) = \mathbf{P}(u,v) + D(u,v)\mathbf{N}'(u,v)$$

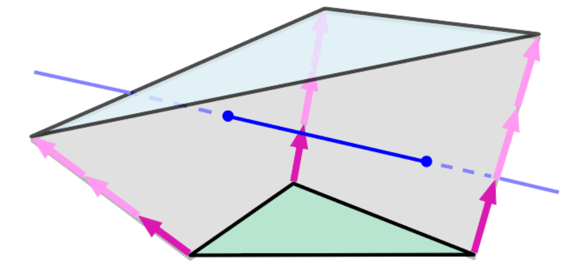
Def. Classical Prisms: $$\mathbf{R}_{std}(u,v,w) = \mathbf{P}(u,v) + w\mathbf{N}'(u,v)$$

Since the normals N_i can point in arbitrary directions, the volume defined by R_{std} is *non-linear* with respect to the triangle's UV coordinates.

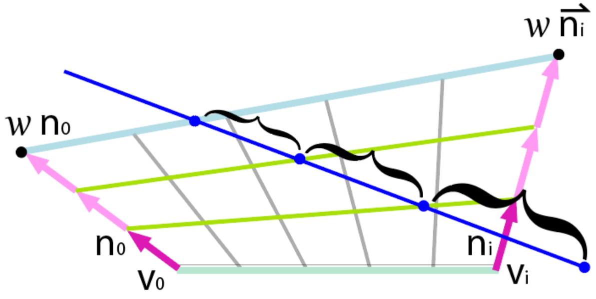
PROJECTIVE DISPLACEMENT MAPPING

DESIGN

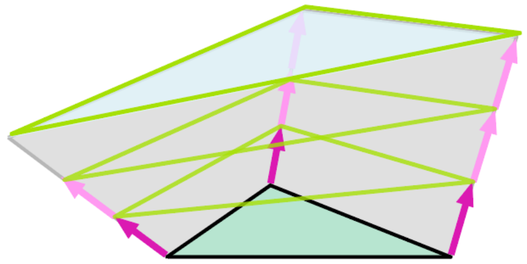
Make the volume orthogonal. **Offsets are perpendicular to base normal.**



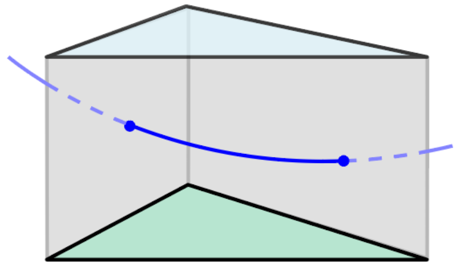
Standard Offset Prism



Triangle Construction

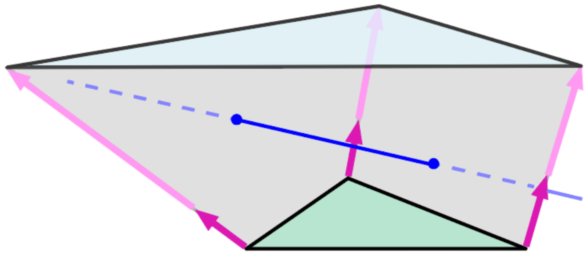


Non-parallel Extruded Triangles

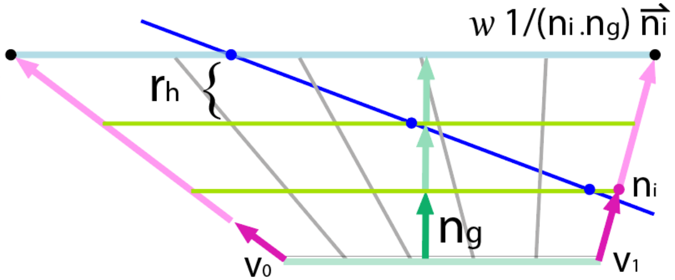


Canonical Space (non-linear rays)

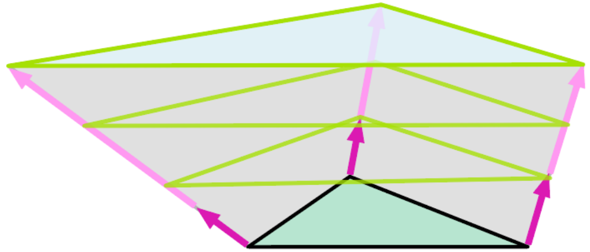
Def. Classical Prisms: $$\mathbf{R}_{std}(u,v,w) = \mathbf{P}(u,v) + w\mathbf{N}'(u,v)$$



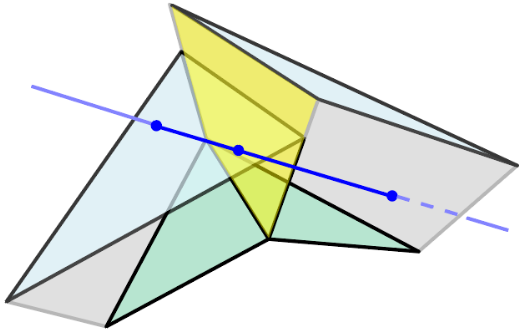
Parallel Offset Prism



Parallel Triangle Construction



Parallel Extruded Triangles



Prism Intersection
(Ray/Bilinear Patch)

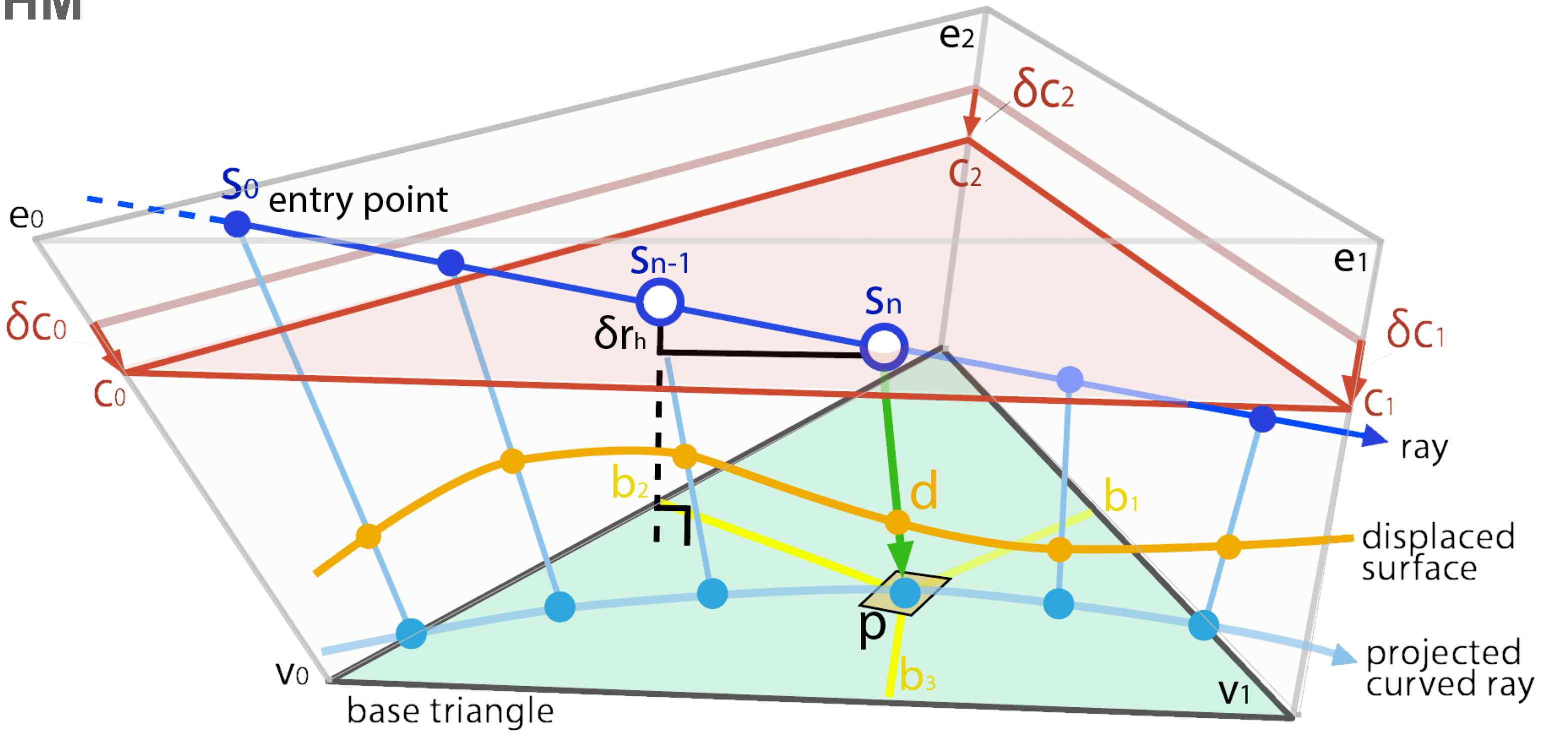
$$n_f(u,v) = u\frac{1}{N_0 \cdot N_g} + v\frac{1}{N_1 \cdot N_g} + (1-u-v)\frac{1}{N_2 \cdot N_g}$$

Normalizing factor

Def. Parallel Offset Prims: $$\mathbf{R}_{pop}(u,v,w) = \mathbf{P}(u,v) + w\underline{n_f(u,v)}\mathbf{N}'(u,v)$$

PROJECTIVE DISPLACEMENT MAPPING

ALGORITHM

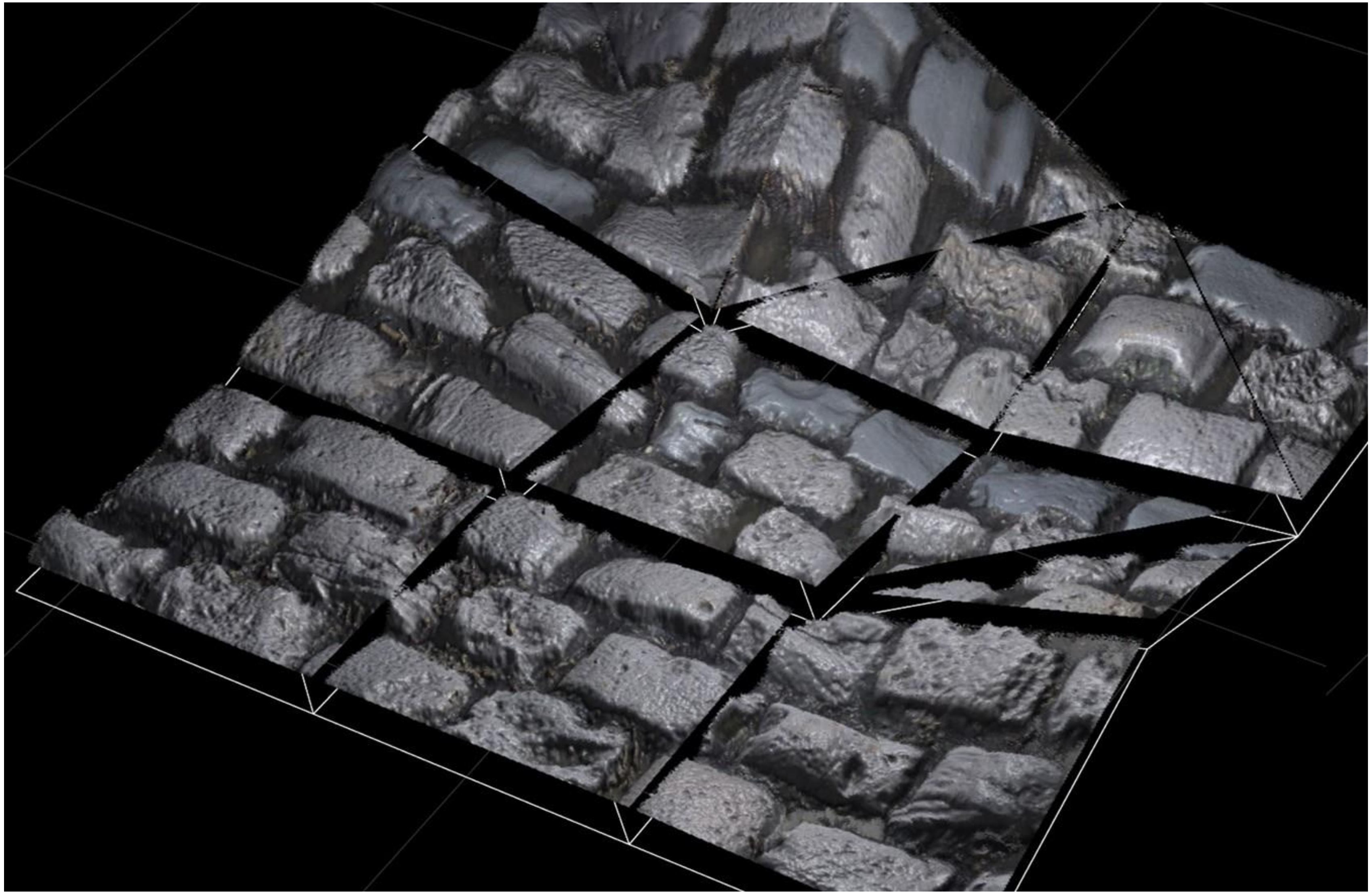


Contribution:
An efficient, direct sampling method for evaluating displacement maps by *projecting* the ray samples, S, along their interpolated normals.
See paper for detailed steps.

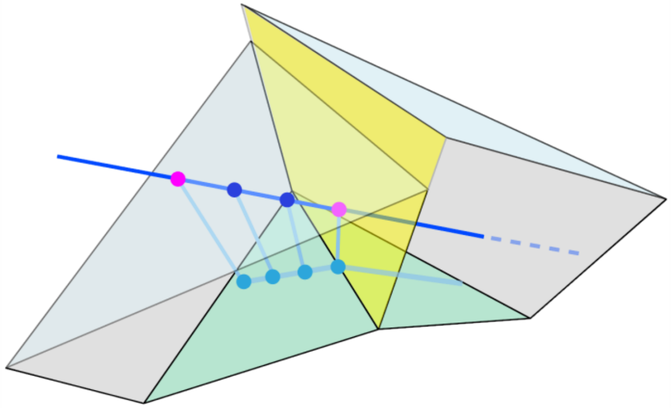
IMPROVEMENTS & FEATURES

WATERTIGHT PRISMS

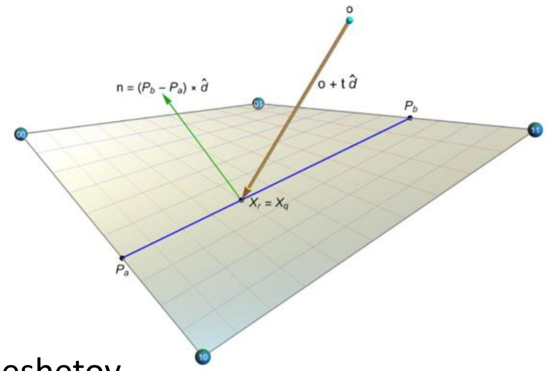
Early results on CPU
(February 2023)



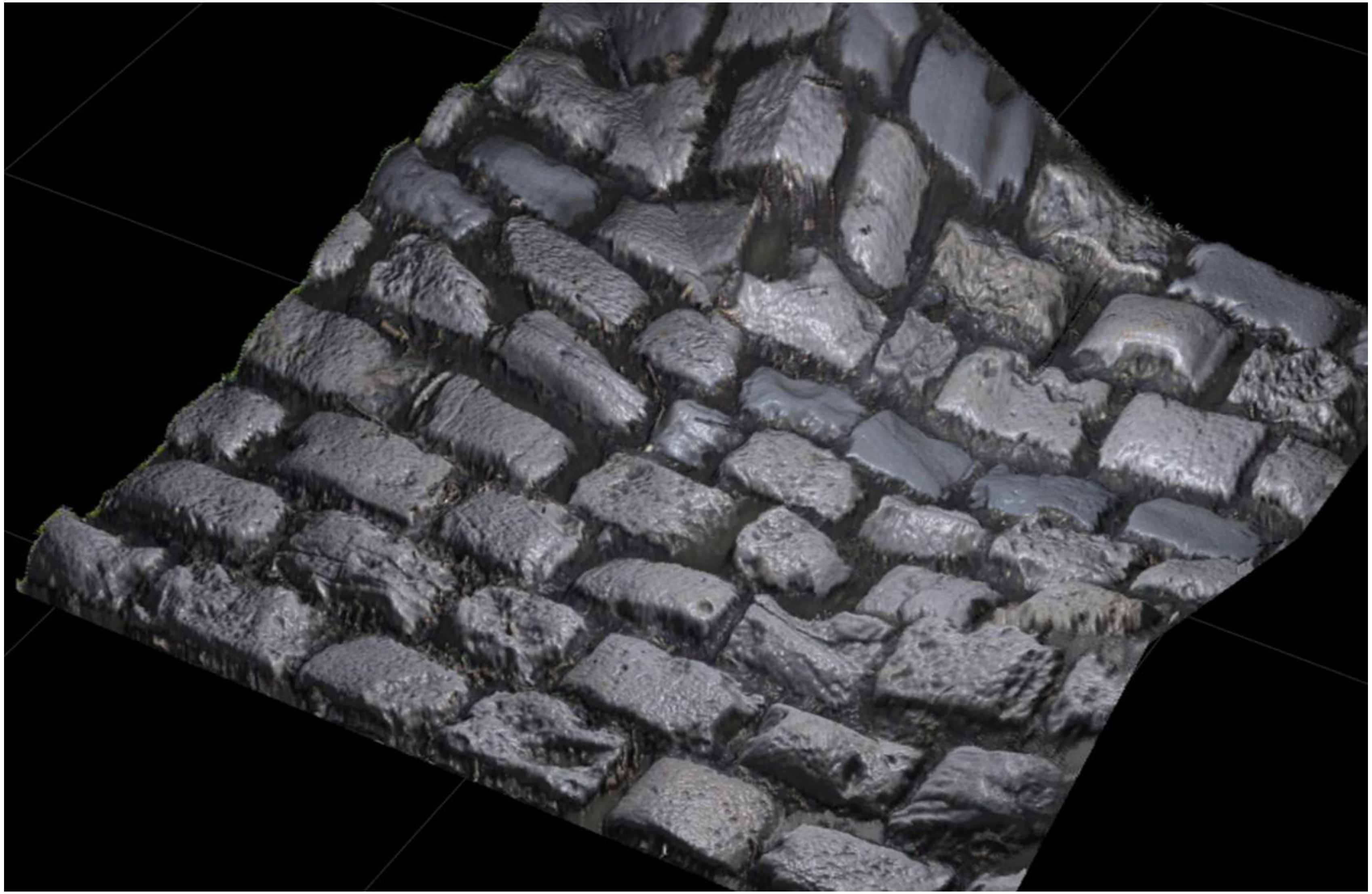
WATERTIGHT PRISMS



Interfaces between two prisms are bi-linear patches.

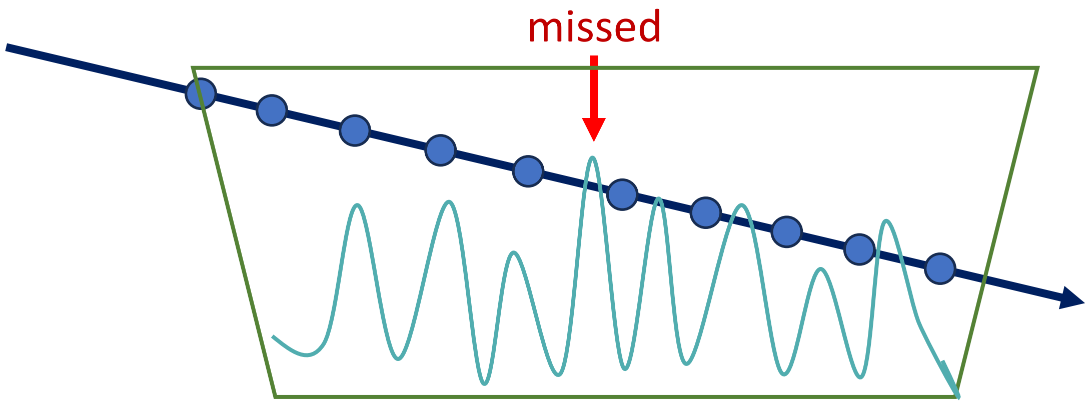


2019, Reshetov,
Cool Patches: A Geometric Approach to Ray/Bilinear Patch Intersections



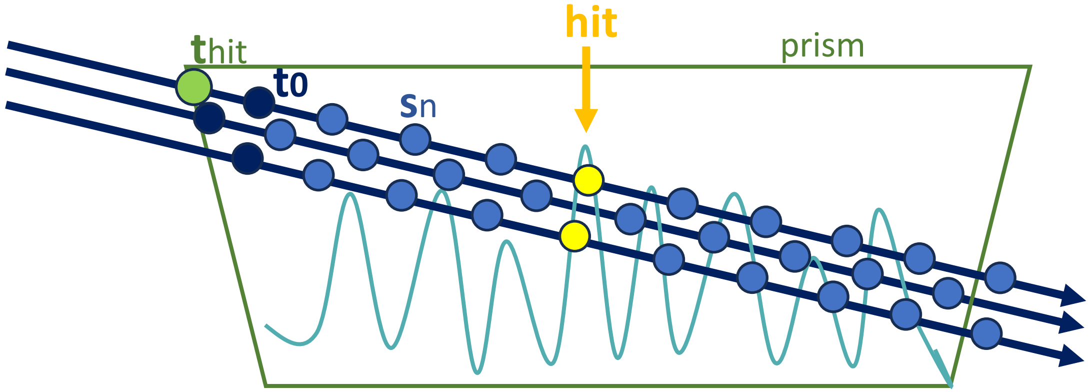
THIN FEATURE SAMPLING

Uniform Sampling
Ray marching is known to miss thin features.



Distributed Sampling
We can distribute samples *along* the ray to integrate over many frames.

Thin Feature Sampling: $t_0 = t_{hit} + Rdt$



There are still many basic, classical problems in Computer Graphics to be solved.

SMOOTH DISPLACED NORMALS

Uses N’

Phong Shading

For non-displaced surfaces,
replace the *flat* normal Ng with the *interpolated* normal N’ for smooth lighting.

$N_L = Ng$

SMOOTH DISPLACED NORMALS

Phong Shading
For non-displaced surfaces,
replace the *flat* normal Ng with the
interpolated normal N’ for smooth lighting.

$$N_L = Ng - Ng + N' = N'$$

Uses N’

SMOOTH DISPLACED NORMALS

Uses N’

Phong Shading

For non-displaced surfaces,
replace the *flat* normal Ng with the
interpolated normal N’ for smooth lighting.

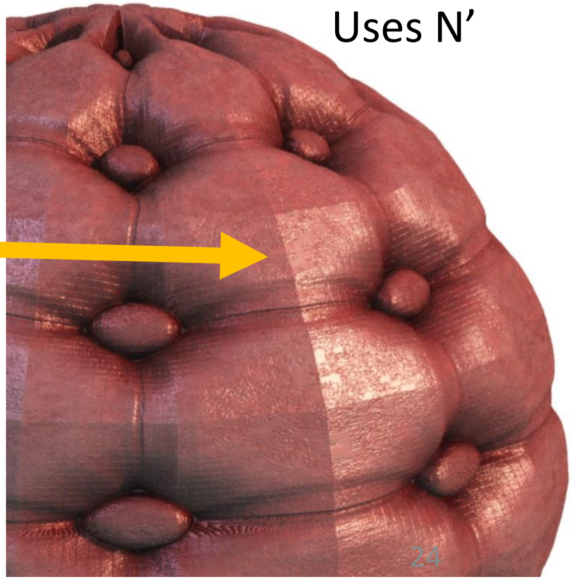
$$N_L = Ng \quad - Ng + N' \quad = \; N'$$

Displaced Shading

Displaced surfaces are *already* defined using the interpolated normal..

Def. $$\mathbf{S}(u,v) = \mathbf{P}(u,v) + D(u,v)\underline{\mathbf{N}'(u,v)}$$

Uses N’



* Why then do *displaced* surfaces show base mesh
discontinuities again?

SMOOTH DISPLACED NORMALS

PROOF:

The displaced normal, evaluated by finite differencing of S, contains the *flat* geometric normal Ng even though S is defined over the interpolated normal N’.

We can reuse the idea of Phong interpolation, and *replace* the flat normal Ng with interpolated N’ for displaced surfaces.

B Displaced and Corrected Surface Normals

PROOF. Construct a smoothed displaced surface normal $\mathbf{N}'_s$ that contains the interpolated (smoothed) base triangle normal $\mathbf{N}'$ for base triangles $\mathbf{P}$, and scalar displacement function D.

We show first that the evaluated normal $\mathbf{N}_s$ of a displaced surface S, sampled at a point (u, v) on a triangle using finite differences, contains only the flat geometric normal N_g of the base triangles even though the surface is constructed using the interpolated normal $\mathbf{N}'$.

Given the displaced surface:

$$\mathbf{S}(u,v,w) = \mathbf{P}(u,v) + D(u,v)\mathbf{N}'(u,v) \tag{4}$$

The surface normal is given by the cross product of the tangent and bi-tangent:

$$\mathbf{N}_s = \frac{d\mathbf{S}}{du} \times \frac{d\mathbf{S}}{dv} \tag{5}$$

Therefore, taking the partial derivatives of equation (4):

$$\frac{d\mathbf{S}}{du} = \frac{d\mathbf{P}}{du} + \frac{dD}{du}\mathbf{N}' \tag{6}$$

$$\frac{d\mathbf{S}}{dv} = \frac{d\mathbf{P}}{dv} + \frac{dD}{dv}\mathbf{N}' \tag{7}$$

Rewriting equation (5) in terms of these:

$$\mathbf{N}_s = \left(\frac{d\mathbf{P}}{du} + \frac{dD}{du}\mathbf{N}'\right) \times \left(\frac{d\mathbf{P}}{dv} + \frac{dD}{dv}\mathbf{N}'\right) \tag{8}$$

Using the distributive property of the cross product, ie. $(\mathbf{A}+\mathbf{B}) \times (\mathbf{C}+\mathbf{D}) = (\mathbf{A}\times\mathbf{C}) + (\mathbf{A}\times\mathbf{D}) + (\mathbf{B}\times\mathbf{C}) + (\mathbf{B}\times\mathbf{D})$:

$$\mathbf{N}_s = \left(\frac{d\mathbf{P}}{du} \times \frac{d\mathbf{P}}{dv}\right) + \left(\frac{d\mathbf{P}}{du} \times \frac{dD}{dv}\mathbf{N}'\right) + \left(\frac{dD}{du}\mathbf{N}' \times \frac{d\mathbf{P}}{dv}\right) + \left(\frac{dD}{du}\mathbf{N}' \times \frac{dD}{dv}\mathbf{N}'\right) \tag{9}$$

The first term is just the geometric (flat) normal of the base triangle:

$$\left(\frac{d\mathbf{P}}{du} \times \frac{d\mathbf{P}}{dv}\right) = \mathbf{N}_g \tag{10}$$

The next two terms in (9) can be rewritten using the property of cross product scalar factoring, i.e., $s_1\mathbf{A} \times s_2\mathbf{B} = s_1 s_2(\mathbf{A}\times\mathbf{B})$:

$$\left(\frac{d\mathbf{P}}{du} \times \frac{dD}{dv}\mathbf{N}'\right) = \frac{dD}{dv}\left(\frac{d\mathbf{P}}{du} \times \mathbf{N}'\right) \tag{11}$$

$$\left(\frac{dD}{du}\mathbf{N}' \times \frac{d\mathbf{P}}{dv}\right) = \frac{dD}{du}\left(\mathbf{N}' \times \frac{d\mathbf{P}}{dv}\right) \tag{12}$$

We notice that $\frac{d\mathbf{P}}{du}$ and $\frac{d\mathbf{P}}{dv}$ are the tangent and bitangent of the base triangle at $\mathbf{P}$. The tangent and bitangent are complementary with respect to the normal, therefore:

$$\frac{dD}{dv}\left(\frac{d\mathbf{P}}{du} \times \mathbf{N}'\right) = \frac{dD}{dv}\frac{d\mathbf{P}}{dv} \tag{13}$$

$$\frac{dD}{du}\left(\mathbf{N}' \times \frac{d\mathbf{P}}{dv}\right) = \frac{dD}{du}\frac{d\mathbf{P}}{du} \tag{14}$$

The last term in equation (9) can also be reduced by cross product scalar factoring:

$$\left(\frac{dD}{du}\mathbf{N}' \times \frac{dD}{dv}\mathbf{N}'\right) = \frac{dD}{du}\frac{dD}{dv}(\mathbf{N}' \times \mathbf{N}') \tag{15}$$

Since the cross product of a vector with itself is zero, we have:

$$(\mathbf{N}' \times \mathbf{N}') = 0 \tag{16}$$

Rewriting equation (9) using equations (10), (13), and (14), we have:

$$\mathbf{N}_s = \mathbf{N}_g + \left(\frac{dD}{dv}\frac{d\mathbf{P}}{dv}\right) + \left(\frac{dD}{du}\frac{d\mathbf{P}}{du}\right) \tag{17}$$

We wish to construct a smoothly interpolated displaced surface normal $\mathbf{N}'_s$. Construction follows the idea of Phong shading, which replaces the geometric normal $\mathbf{N}_g$ with the smooth interpolated normal $\mathbf{N}'$ when rendering non-displaced triangles. For displaced surfaces we can replace it as:

$$\mathbf{N}_s{}' = \mathbf{N}_s - \mathbf{N}_g + \mathbf{N}' \tag{18}$$

Substituting equation (17) in for $\mathbf{N}_s$.

$$\mathbf{N}_s{}' = [\mathbf{N}_g + \left(\frac{dD}{dv}\frac{d\mathbf{P}}{dv}\right) + \left(\frac{dD}{du}\frac{d\mathbf{P}}{du}\right)] - \mathbf{N}_g + \mathbf{N}' \tag{19}$$

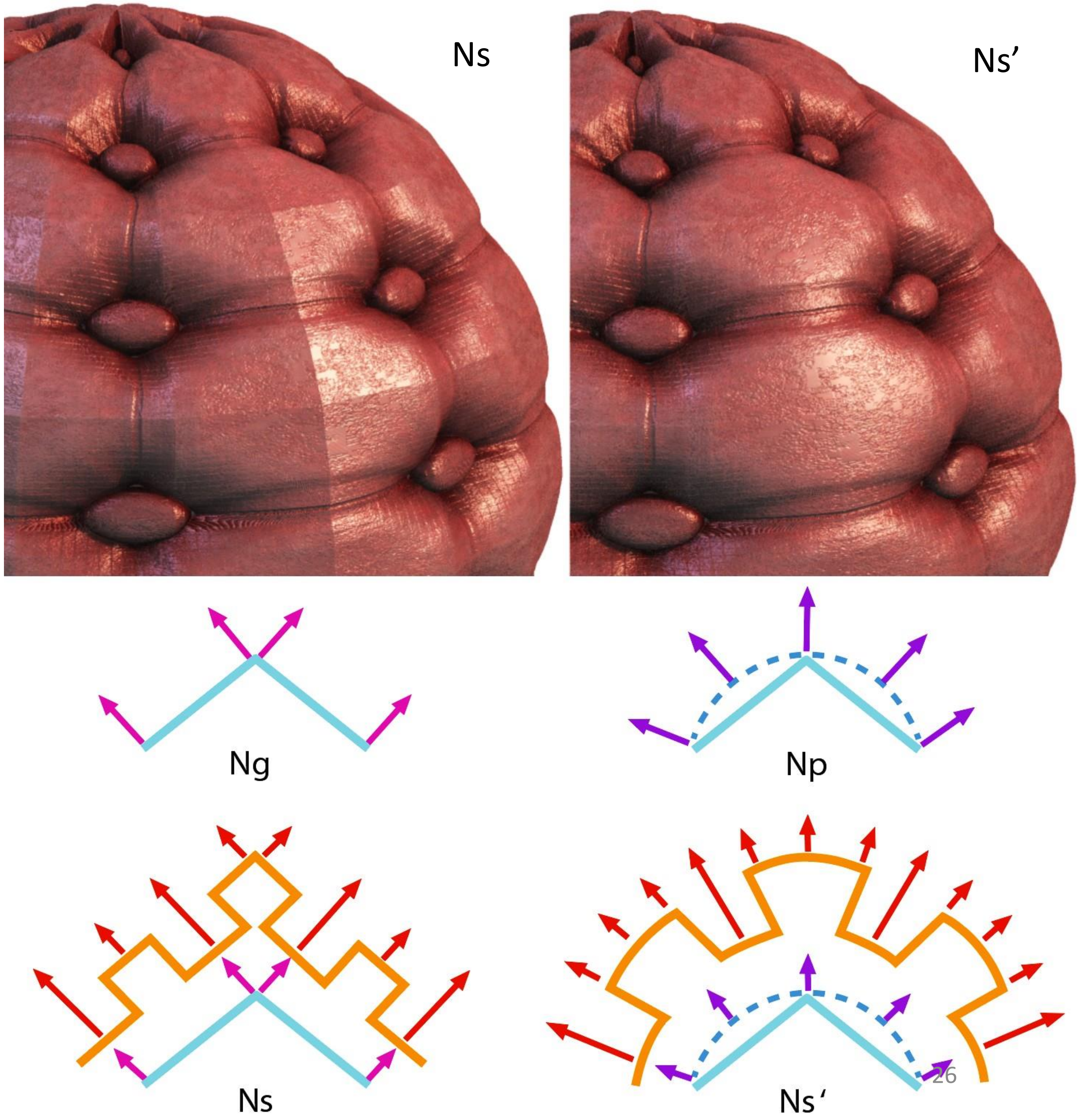
$$\mathbf{N}_s{}' = \mathbf{N}' + \left(\frac{dD}{dv}\frac{d\mathbf{P}}{dv}\right) + \left(\frac{dD}{du}\frac{d\mathbf{P}}{du}\right) \tag{20}$$

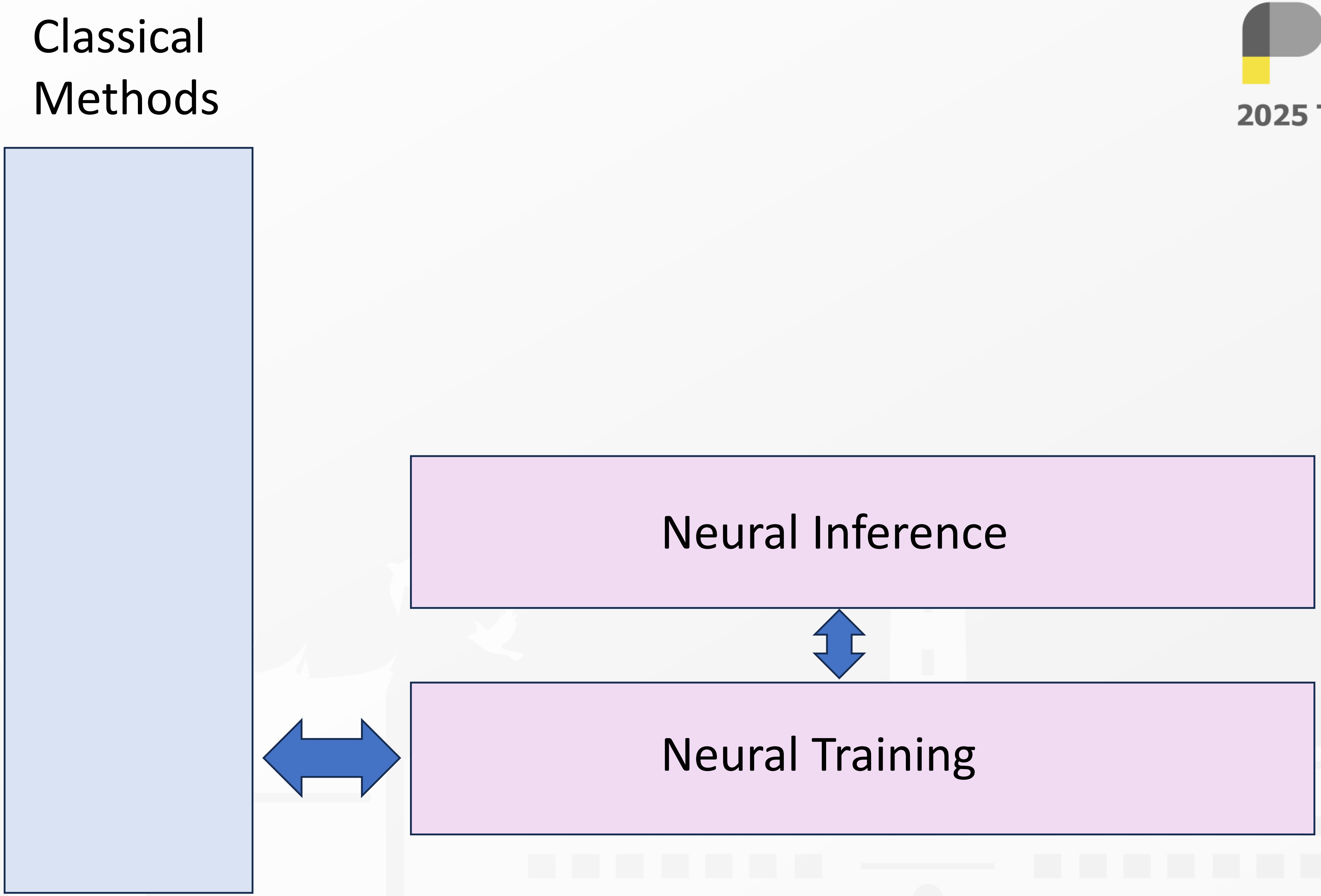
SMOOTH DISPLACED NORMALS

Eqn. Smooth Displaced Normal

From the evaluated displaced normal (Ns), subtract the flat geometric normal (Ng) and add back the interpolated normal (N’).

$$Ns' = Ns - Ng + N'$$




Classical
Methods
Neural Inference
Neural Training

Classical Methods

LLM Proof Assistance

Q: List some mathematical equalities could help to expand the cross product found in the surface normal?

3. def. distributive property of the cross product

(A+B) x (C+D) = (A x C) + (A x D) + (B x C) + (B x D)

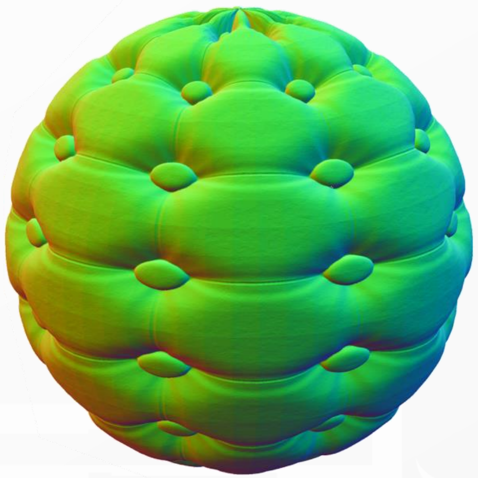
RESULTS

VIDEO

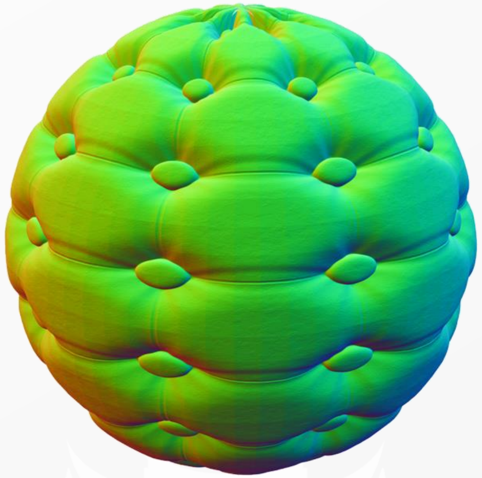
30 seconds, 4096 x 1280 pixels, 15k triangles, 1.8 MB GPU mem, >24 GB tessellated

RESULTS

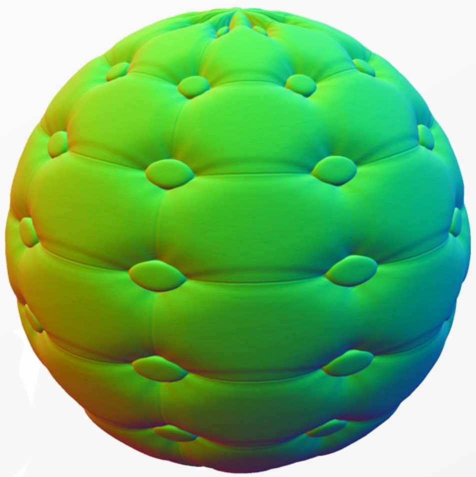
VISUAL RESULTS



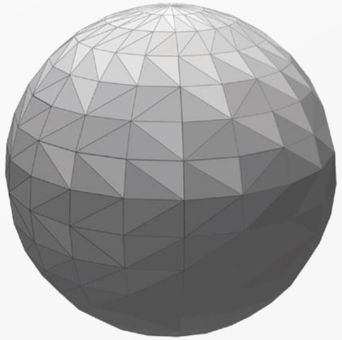
Micro-Meshes
(Nvidia 2023)

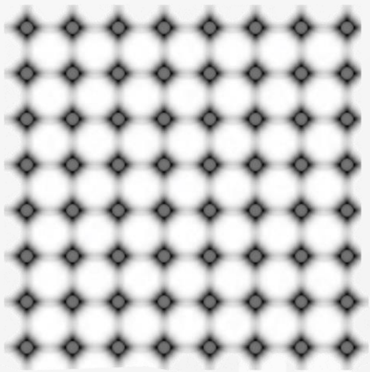


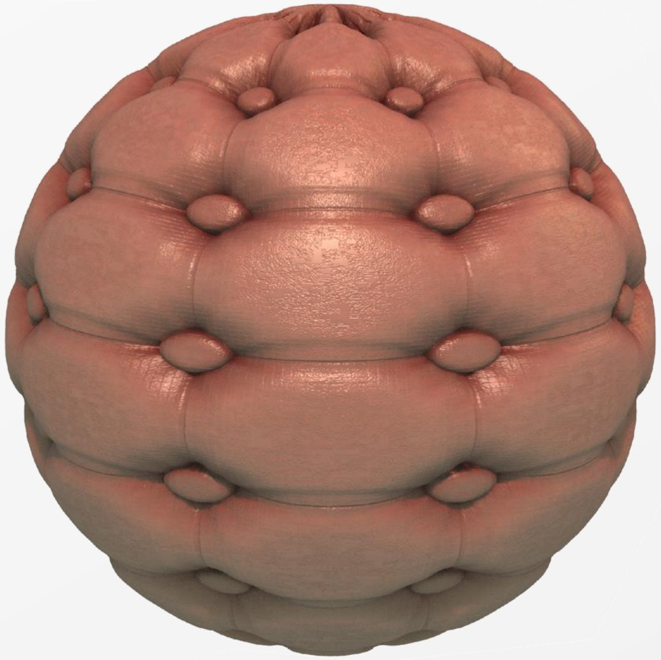
RMIP
(Thonat 2023)



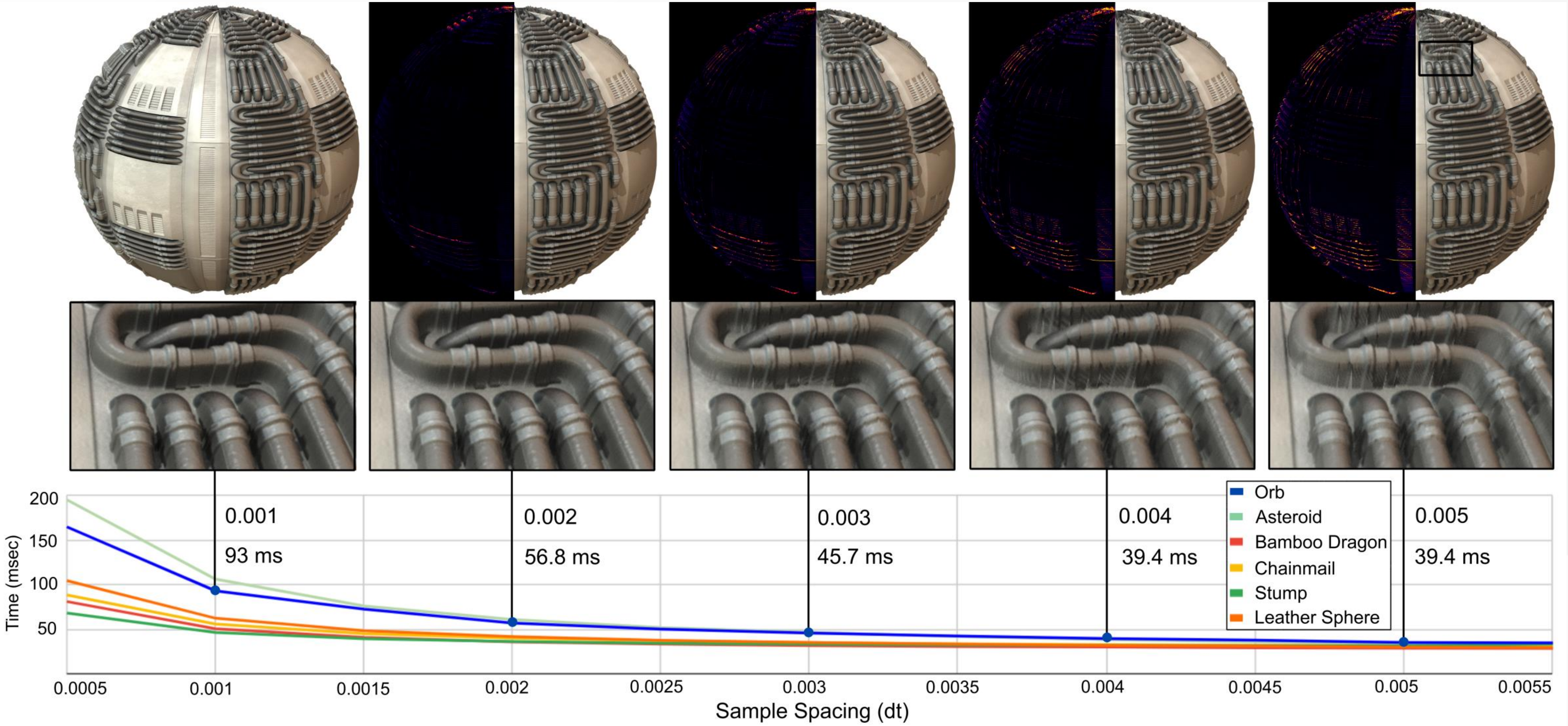
PDM
(Ours)







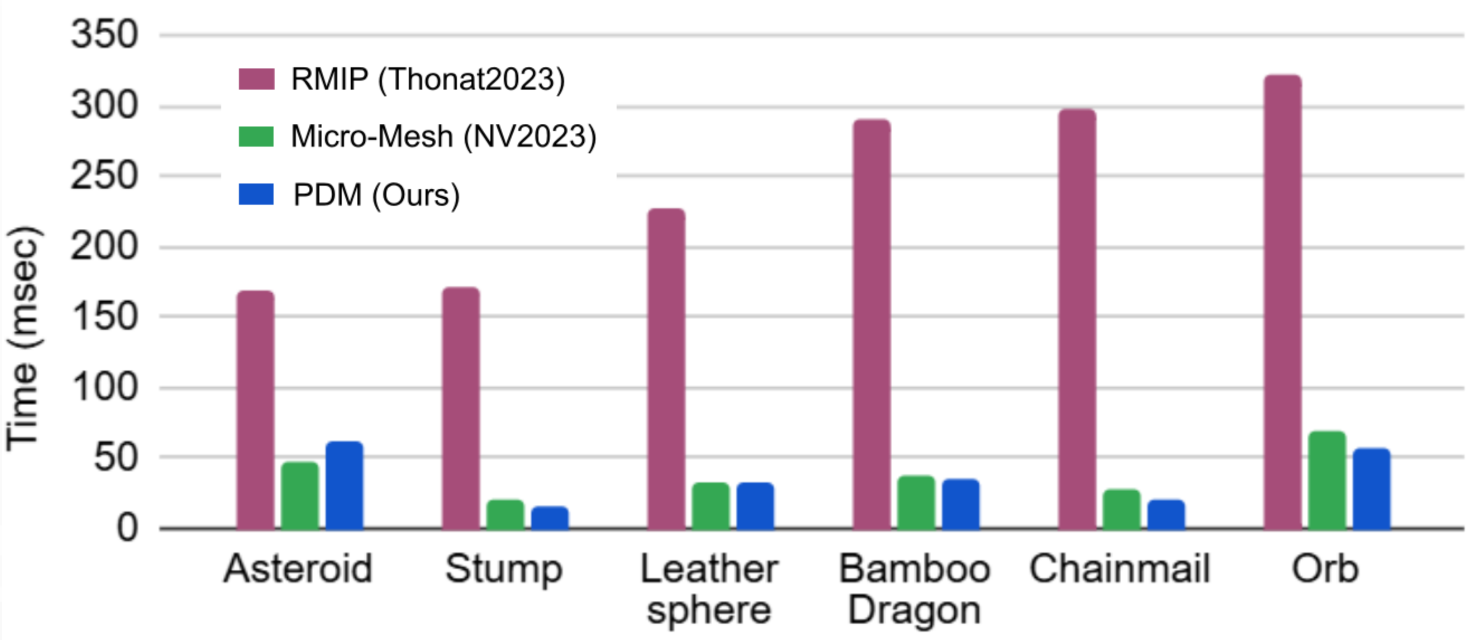
SAMPLE SPACING



Sample spacing (dt) is a user-defined tradeoff between quality and performance.

RESULTS

PERFORMANCE



		Beauty Image (msec)			
Model	# Tri.	Micro Mesh	RMIP 2023	Ours	Speed up
Asteroid	480	48.3	145.6	60.8	2.4
Chainmail	1423	28.9	289.0	20.7	13.9
Leather Sphere	3968	33.7	230.6	31.9	7.2
Orb	3968	68.2	329.1	56.8	5.8
Tree Stump	7710	19.5	166.2	18.1	9.2
Bamboo Dragon	22308	37.7	282.9	35.7	7.9

Hardware: GeForce GTX 4090

2x-14x faster than RMIP on same hardware.
Slightly faster than Micro-Meshes, yet without complex data structures.

RESULTS

MEMORY

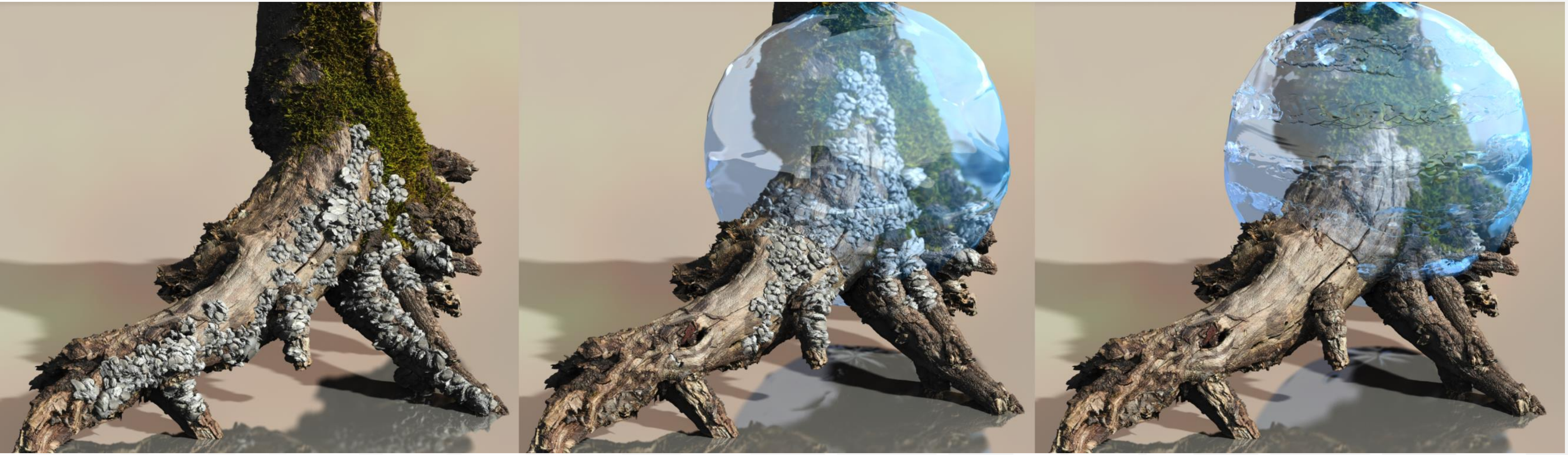
	Common				Algorithm Specific (MB)			Algorithm Overhead %		
Model	Displace Texture	Base Mesh	BVH	Total (MB)	MicroMesh Bary.Map	RMIP Dmap	PDM Prisms	MicroMesh	RMIP	PDM (ours)
Asteroid	33	0.008	0.08	33.1	6.9	27.19	0.049	21%	82%	0.1%
Chainmail	33	0.024	0.24	33.3	10.8	27.56	0.147	32%	83%	0.4%
Stump	33	0.132	1.32	34.7	13.6	30.05	0.794	39%	87%	2.3%
Bamboo Dragon	33	0.383	3.83	37.2	7.0	35.85	2.298	19%	96%	6.2%
Orb	134	0.068	0.68	134.7	23.2	28.57	0.409	17%	21%	0.3%

Average ~730k memory usage!
Fixed cost of 108 bytes per base triangle.

Memory overhead only: **2-6%**
Compared to Micro-Meshes: 20-40%
or to RMIP: ~80%

RESULTS

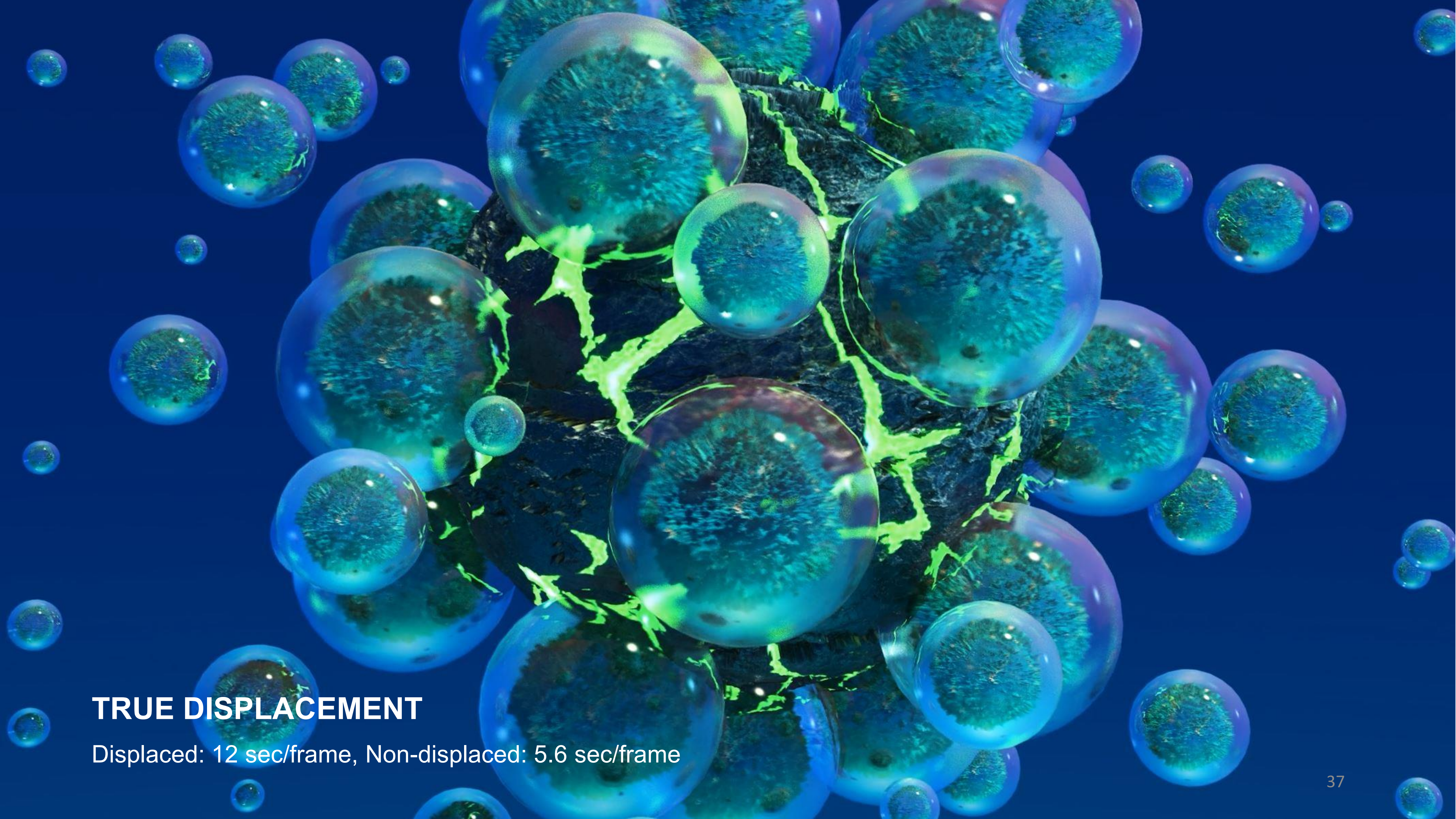
TRUE DISPLACEMENT



Displaced surfaces in **primary** rays, **reflections** and **shadows**.

Displaced surfaces seen through **refractive** objects.

Displace the refractive object itself.



TRUE DISPLACEMENT

Displaced: 12 sec/frame, Non-displaced: 5.6 sec/frame

CONCLUSIONS

2025 TAIPEI

PROJECTIVE DISPLACMENT MAPPING

A novel, accelerated, direct sampling approach to true displacement ray tracing.

FEATURES

- Real-time editing of true displaced surfaces
- Easy integration into RT engines (Optix, Vulkan)
- Use of hardware accelerated BVHs
- Guarantees on watertightness, thin features and smoothness
- Faster, in many cases, than data-structure based methods

LIMITATIONS

- Not intended for massive scenes
- Not for high res. source meshes
- Works best by adding detail to low-poly base meshes
- Intended for look development and authoring

SOURCE CODE

Open Source, Reference for CPU:
https://github.com/quantasci/ProjectiveDisplacement

THANK YOU!

Rama Carl Hoetzlein
https://ramakarl.com

Projective Displacement Mapping
https://github.com/quantasci/ProjectiveDisplacement

Quanta Sciences
https://github.com/quantasci

2025 TAIPEI

UP NEXT :

SINGLE-LINE DRAWING VECTORIZATION