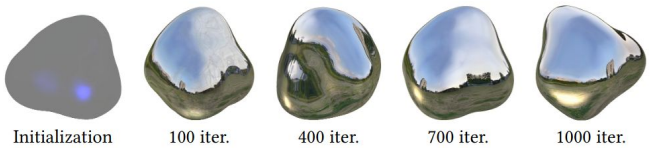
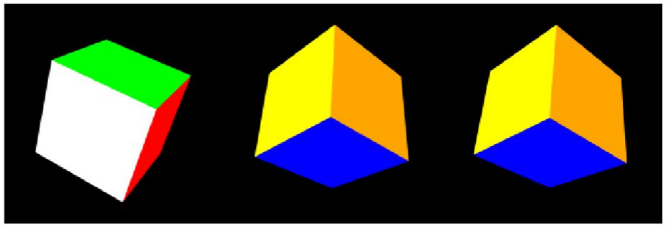
Modular Primitives for High-Performance Differentiable Rendering

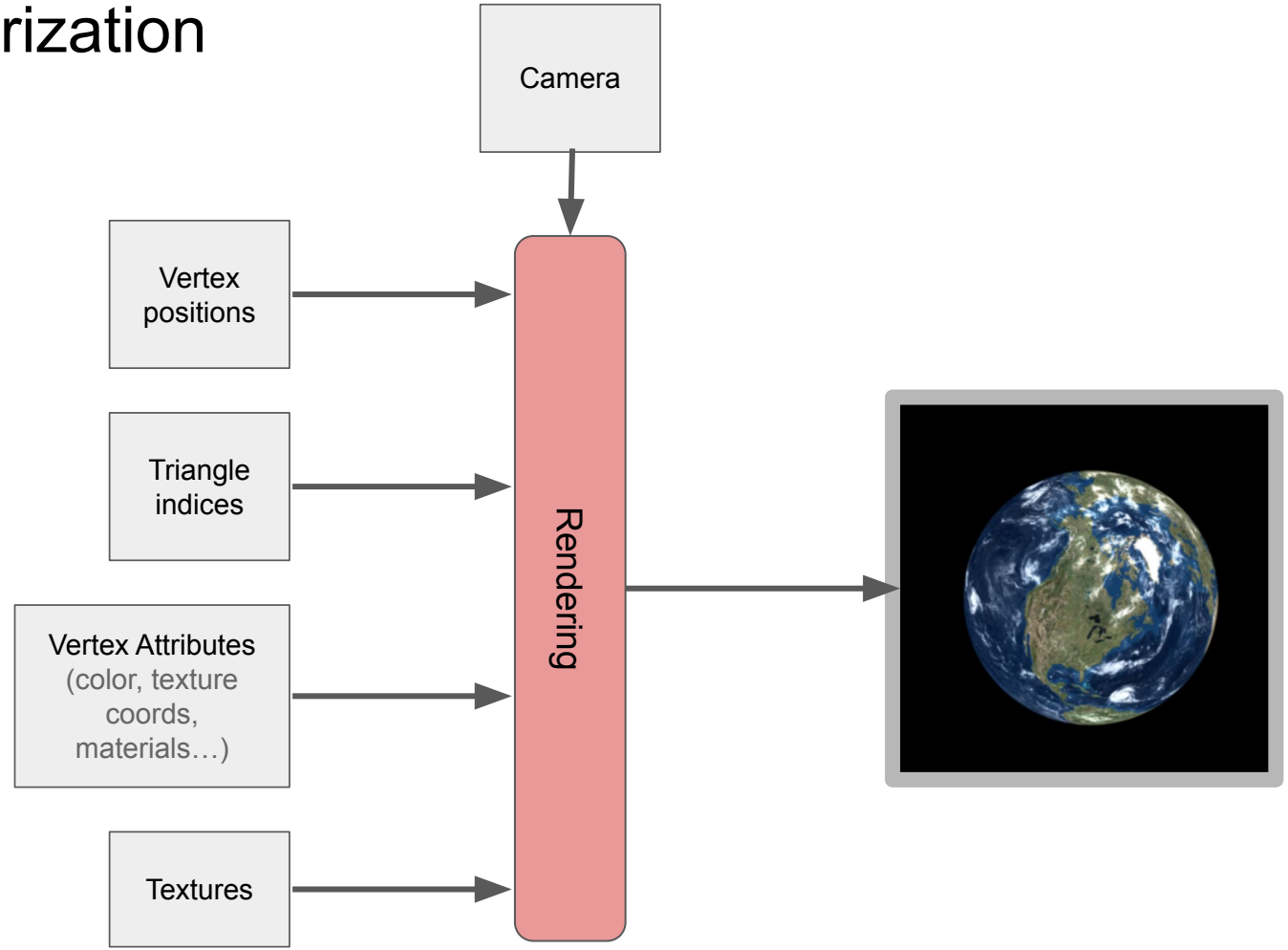
Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, Timo Aila



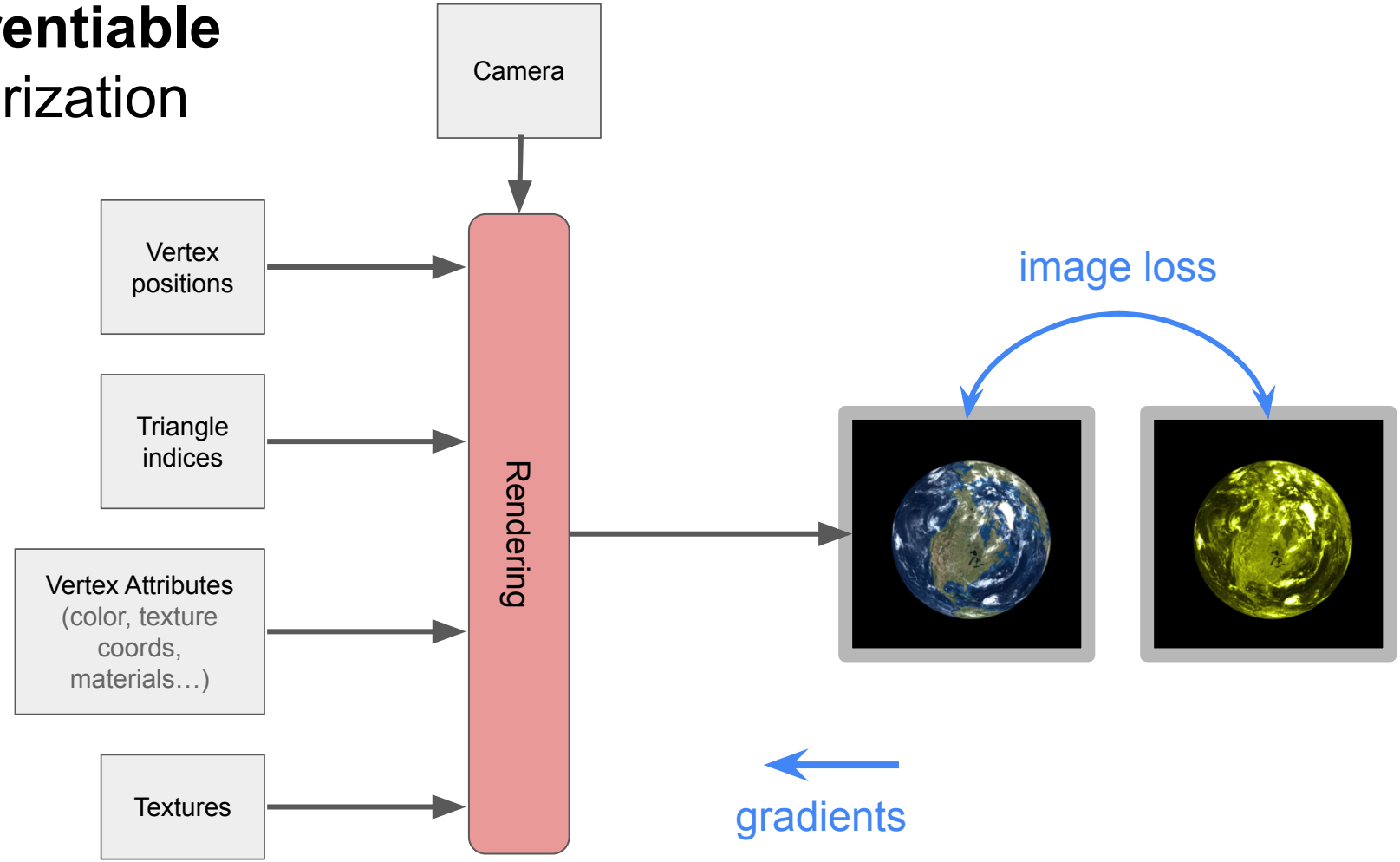


Groupe de Lecture, Gaspard Thévenon, 14 Janvier 2025

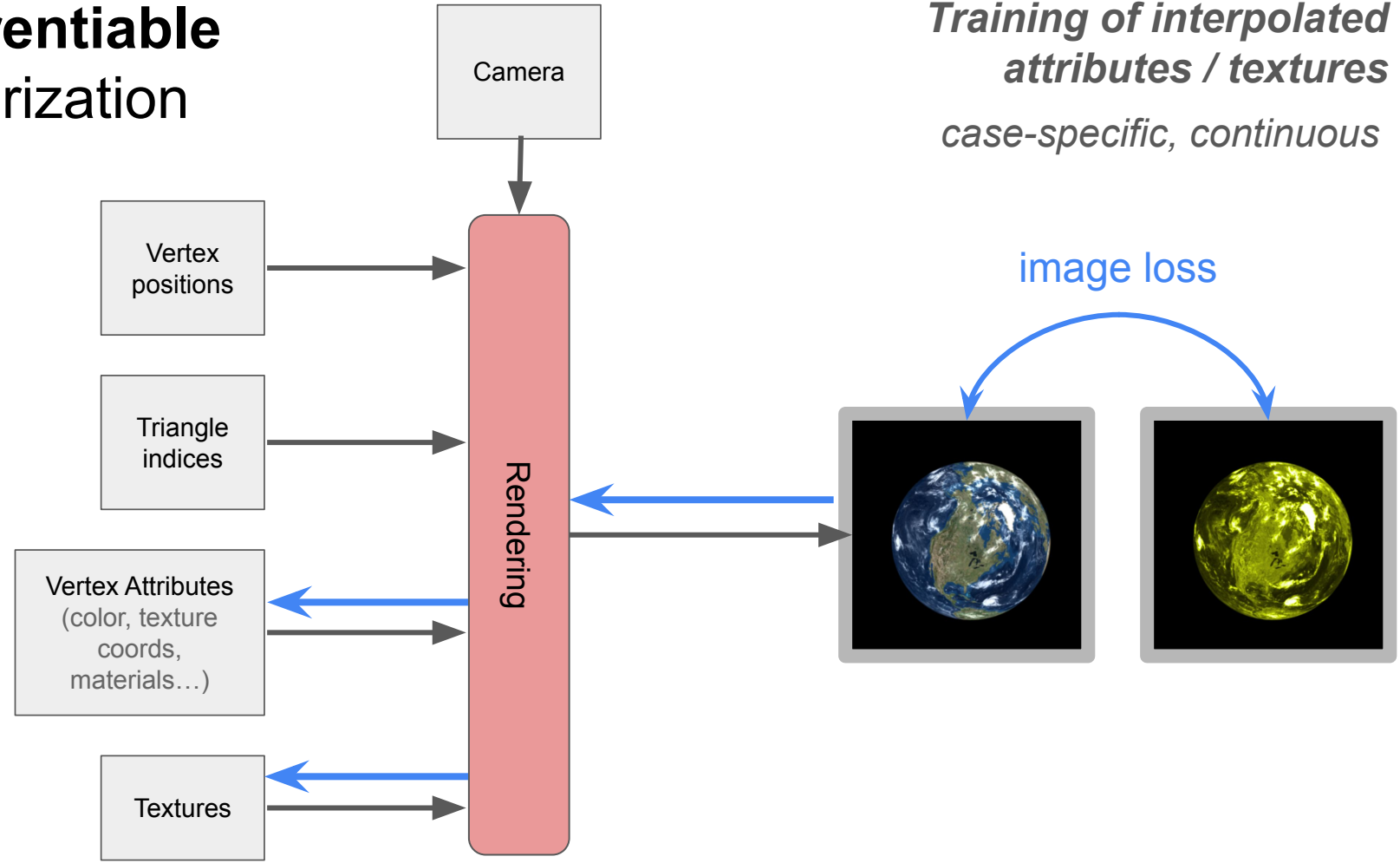
Rasterization



Differentiable Rasterization



Differentiable Rasterization



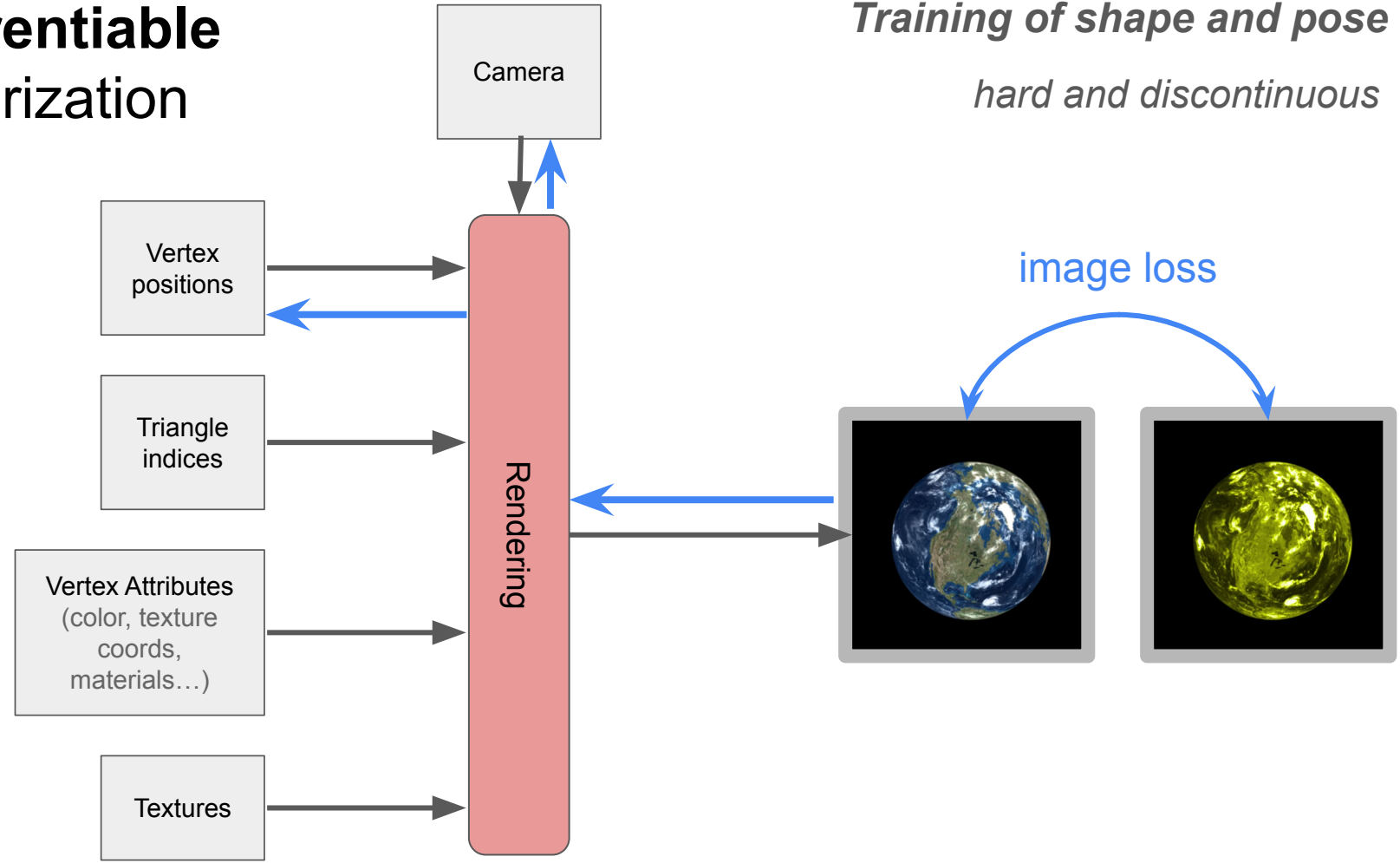
Training of interpolated attributes / textures

case-specific, continuous

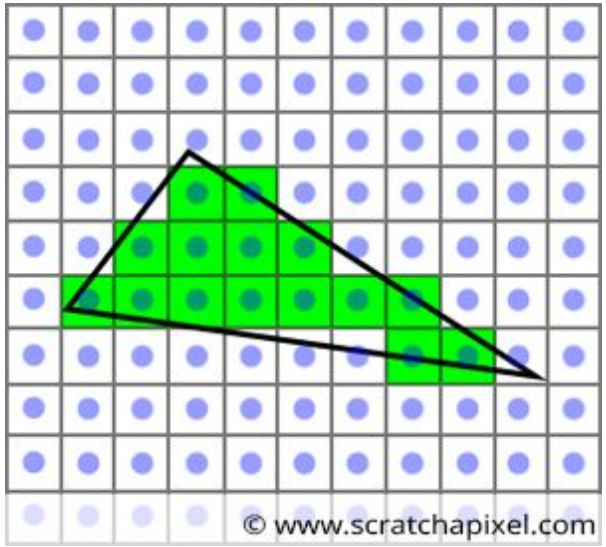
Differentiable Rasterization

Training of shape and pose

hard and discontinuous



Discontinuity of the visibility function



The ***triangle-pixel visibility function*** (“is this triangle visible on this pixel ?”) is ***discontinuous*** (values in {0,1})

-> rasterization is by nature not differentiable.

So for position gradients, one needs sophisticated techniques.

Existing methods

	OpenDR [2014]	NMR [2018]	SoftRas [2019]	DIB-R [2019]	Li et al. [2018]	Mitsuba2 [2019]
Performance*	✔	✗	✗	✗	✖	✖
Flexibility	✖	✖	✖	✖	✔	✔
Correct Gradients	✖	✖	✔	✗	✔	✗
Noise-free	✔	✔	✔	✔	✖	✖
No tuning	✔	✔	✖	✖	✔	✗
GI	✖	✖	✖	✖	✔	✔

Existing methods

	OpenDR [2014]	NMR [2018]	SoftRas [2019]	DIB-R [2019]	Li et al. [2018]	Mitsuba2 [2019]
Performance*	✔	✗	✗	✗	✖	✖
Flexibility	✖	✖	✖	✖	✔	✔
Correct Gradients	✖	✖	✔	✗	✔	✗
Noise-free	✔	✔	✔	✔	✖	✖
No tuning	✔	✔	✖	✖	✔	✗
GI	✖	✖	✖	✖	✔	✔

Existing methods

	OpenDR [2014]	NMR [2018]	SoftRas [2019]	DIB-R [2019]	Li et al. [2018]	Mitsuba2 [2019]
Performance*	✔	✗	✗	✗	✖	✖
Flexibility	✖	✖	✖	✖	✔	✔
Correct Gradients	✖	✖	✔	✗	✔	✗
Noise-free	✔	✔	✔	✔	✖	✖
No tuning	✔	✔	✖	✖	✔	✗
GI	✖	✖	✖	✖	✔	✔

Existing methods

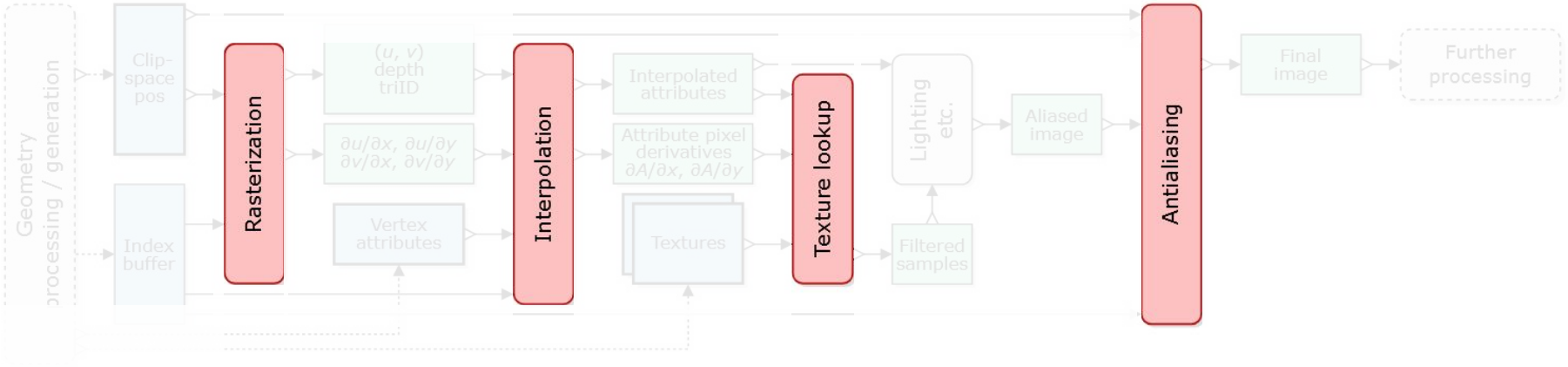
	OpenDR [2014]	NMR [2018]	SoftRas [2019]	DIB-R [2019]	Li et al. [2018]	Mitsuba2 [2019]	**Our**
Performance*	✔	✗	✗	✗	✖	✖	✔ ✔
Flexibility	✖	✖	✖	✖	✔	✔	✔ ✔
Correct Gradients	✖	✖	✔	✗	✔	✗	✔ ✔
Noise-free	✔	✔	✔	✔	✖	✖	✔ ✔ ✔
No tuning	✔	✔	✖	✖	✔	✗	✔ ✔ ✔
GI	✖	✖	✖	✖	✔	✔	✖

nvdiffrast

Goals:

- Flexibility, integration in AD frameworks
- Performance, use of existing OpenGL/hardware pipelines
- Correctness of gradients

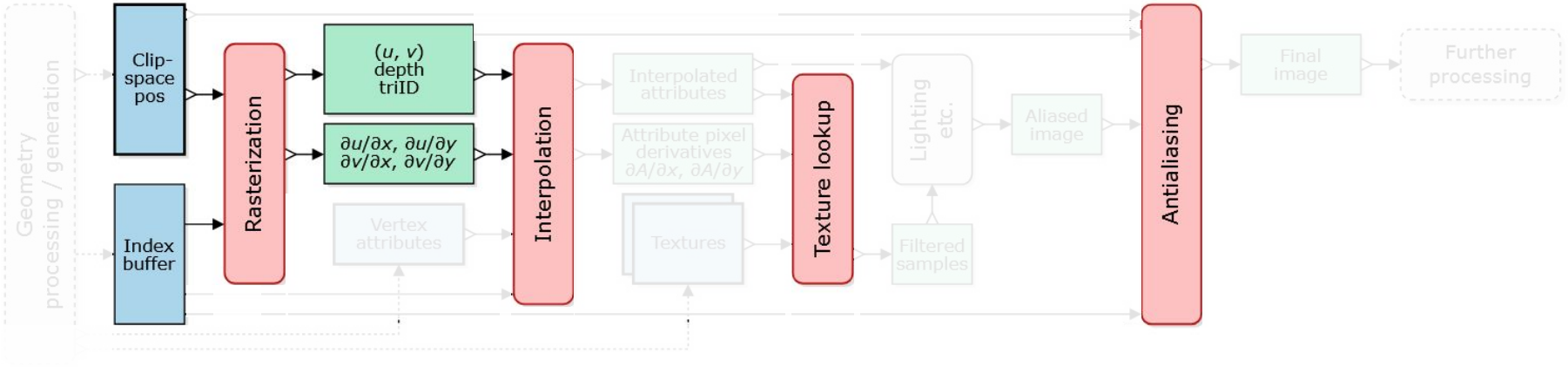
nvdiffrast - forward

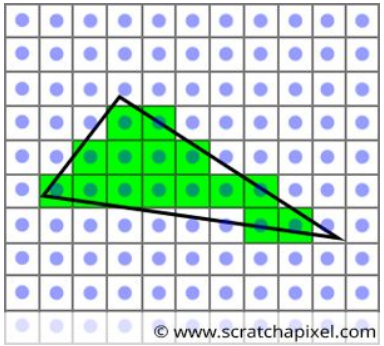


The library offers 4 modular primitives

Parameters are tensors (TF, PyTorch)

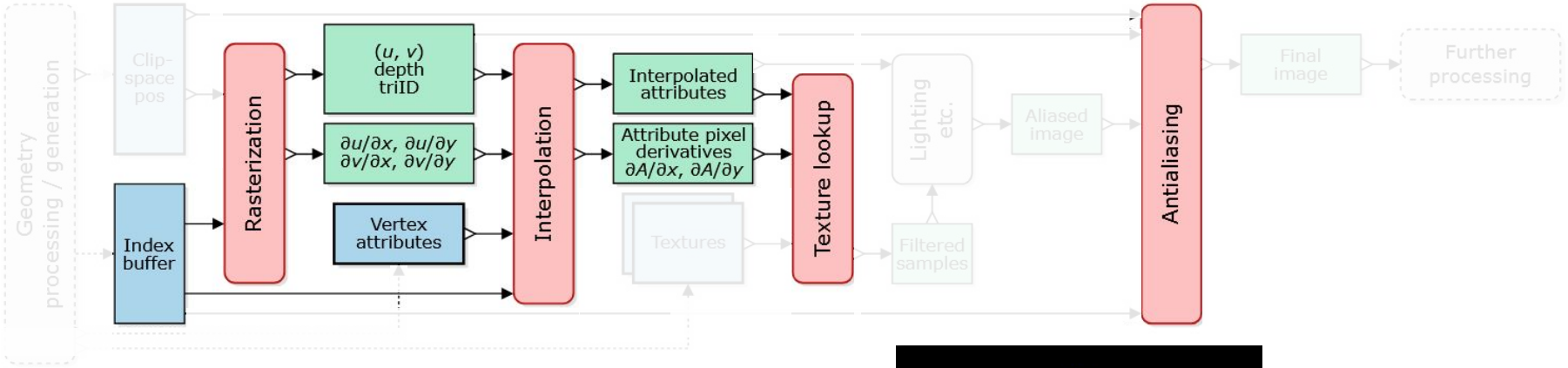
nvdiffrast - forward

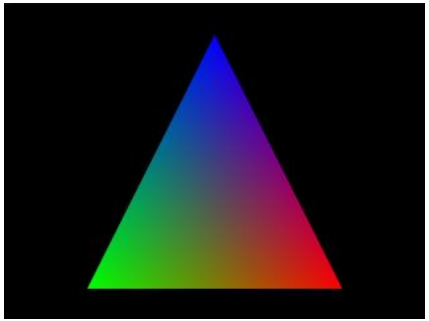




Rasterization

nvdiffrast - forward

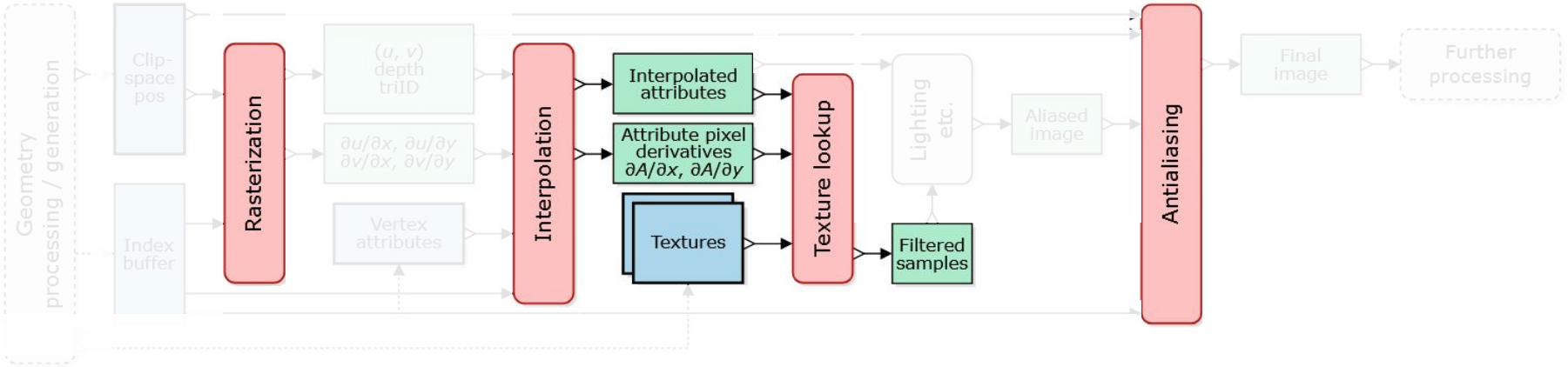




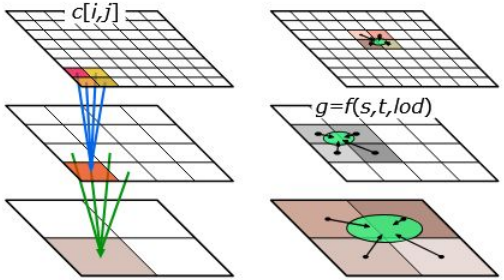
$$\partial A/\partial\{x,y\} = [\partial\{u,v\}/\partial\{x,y\}][\partial A/\partial\{u,v\}]$$

Attributes interpolation

nvdiffrast - forward

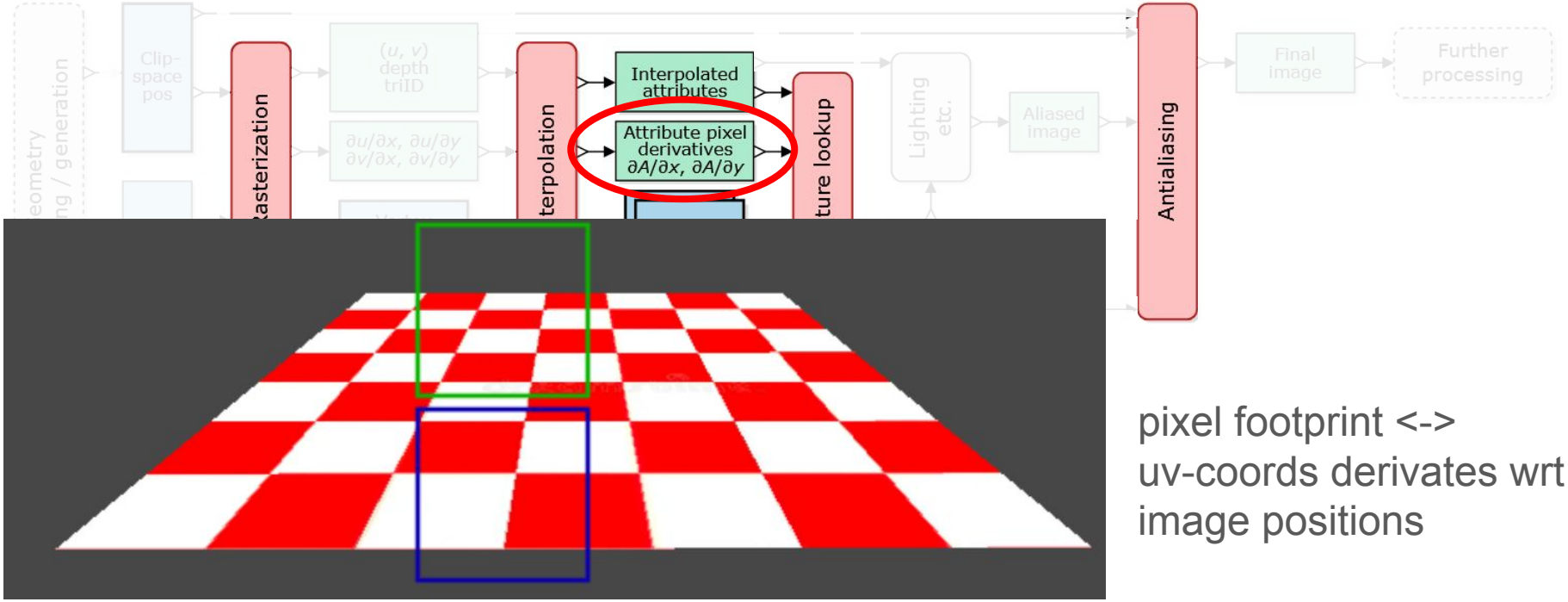


Supports prefiltered texture sampling
(aka mip-maps)



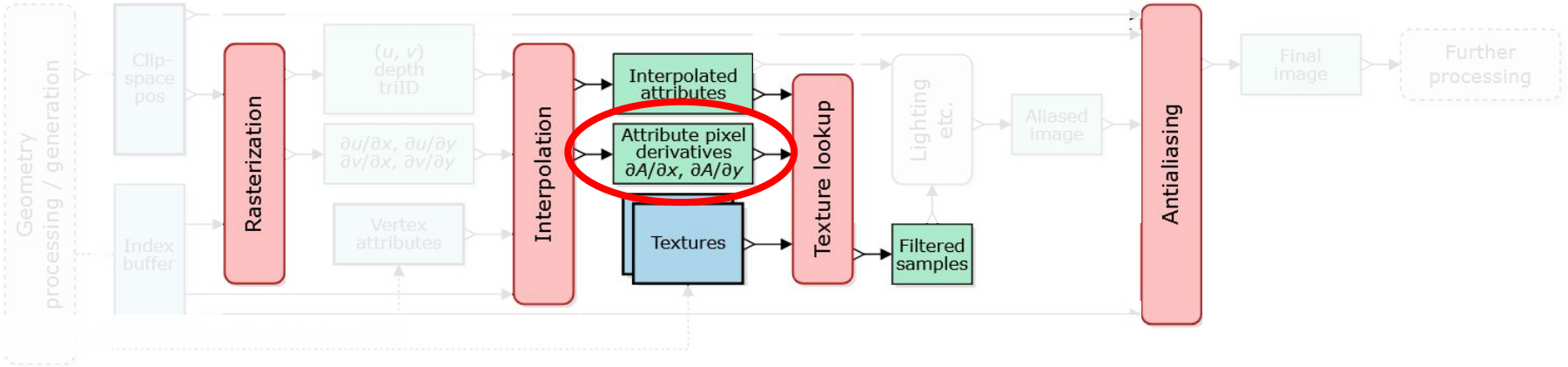
Texture lookup

nvdiffrast - forward



pixel footprint <->
uv-coords derivates wrt
image positions

nvdiffrast - forward

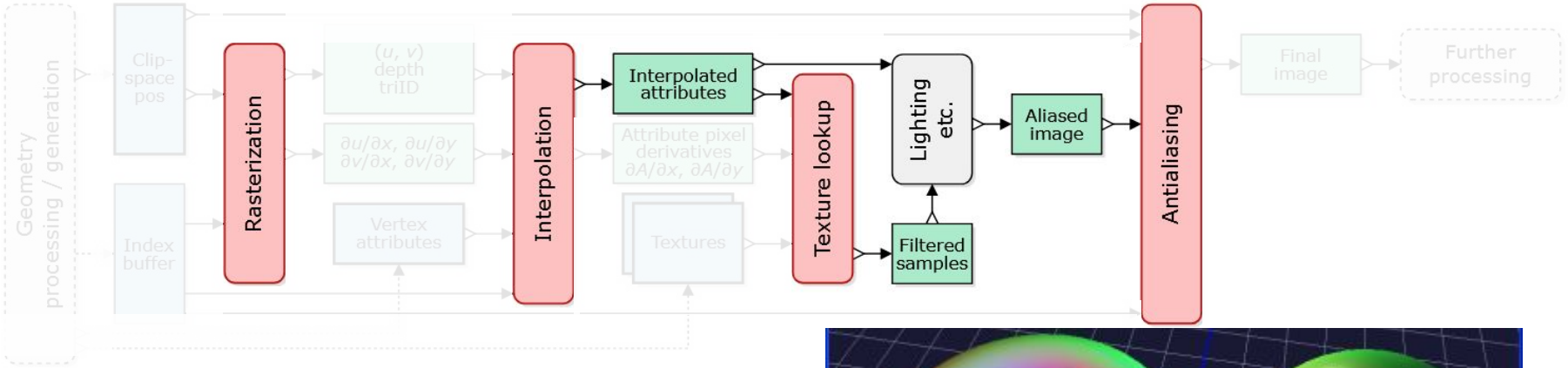


$$MipLevel(x, y) = log_2(\rho(x, y))$$

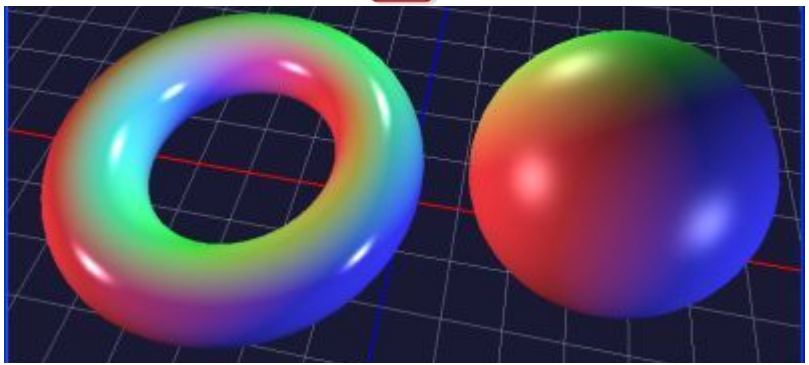
$$\rho = max\left\{\sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2}, \sqrt{\left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2}\right\}$$

https://pema.dev/2025/05/09/mipmaps-too-much-detail/

nvdiffrast - forward

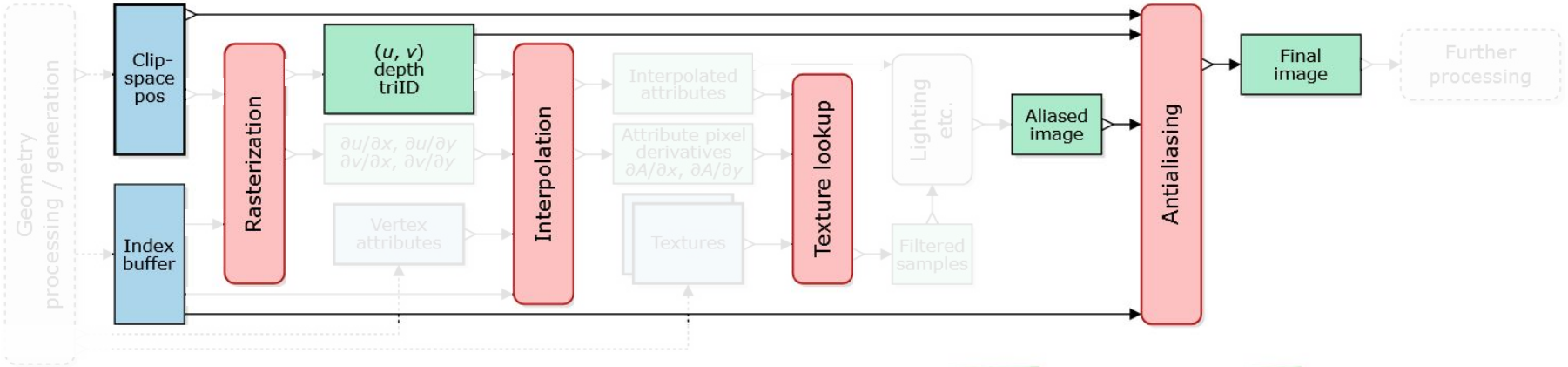


not included
must be done manually (in AD framework)
by the user

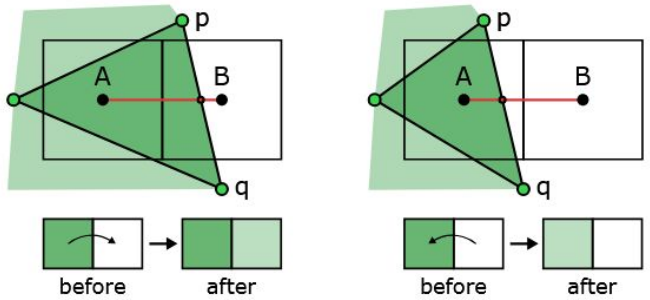


Custom shading

nvdiffrast - forward

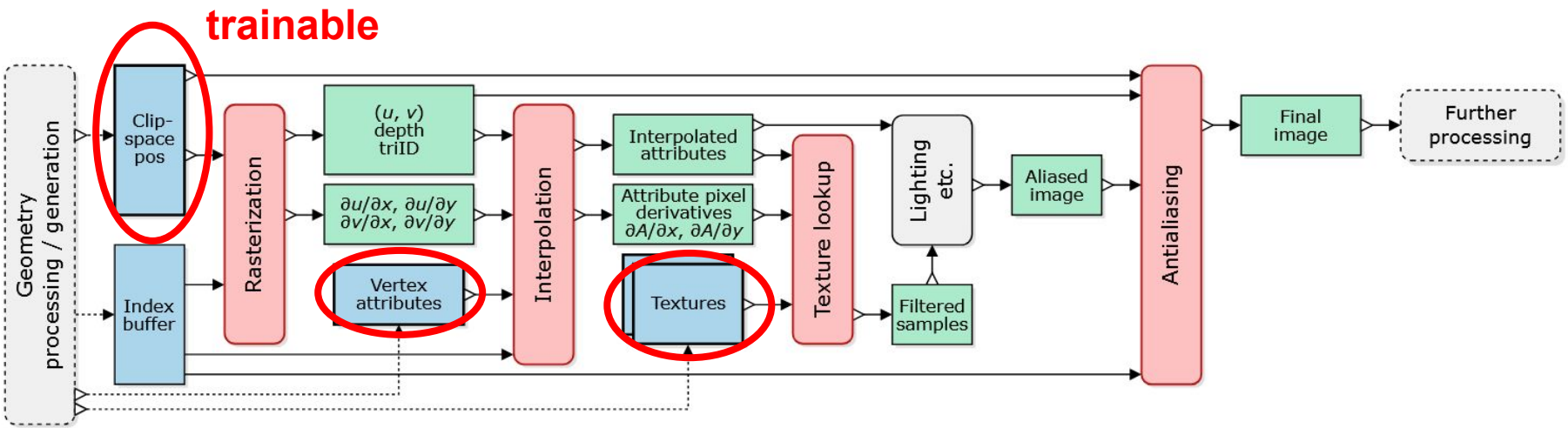


- Fetch pixel pairs with triID changes
- Keep silhouette edges, crossing segments between the adjacent pixels
- Adjust colors wrt the relative triangle coverage of the pixels



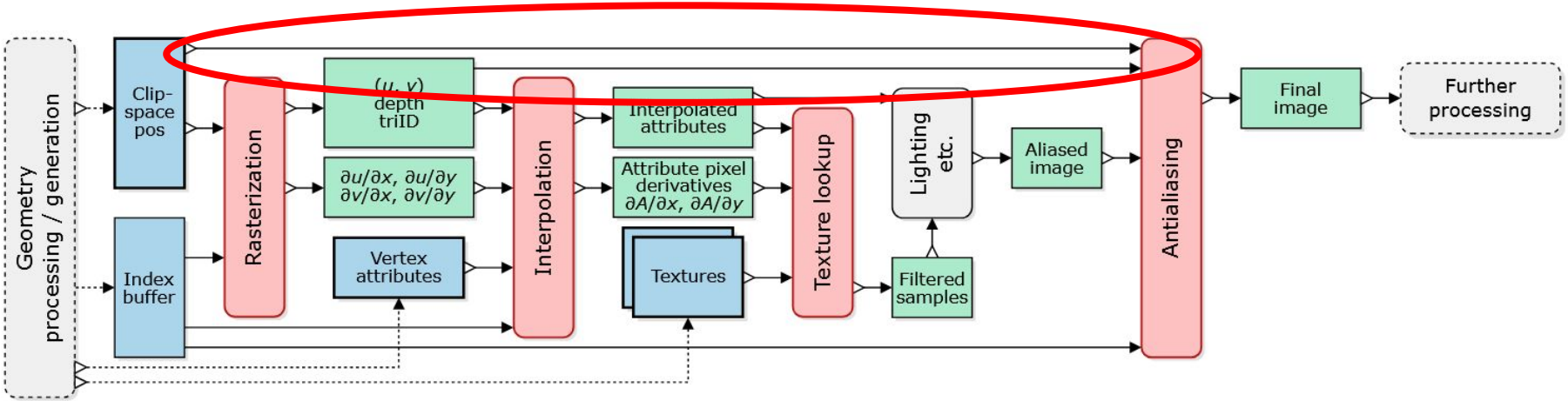
Anti-aliasing

nvdiffrast - backward

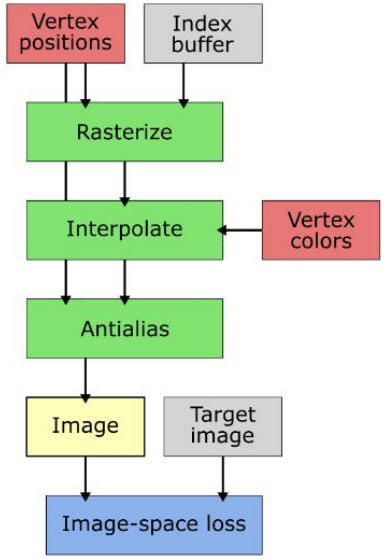


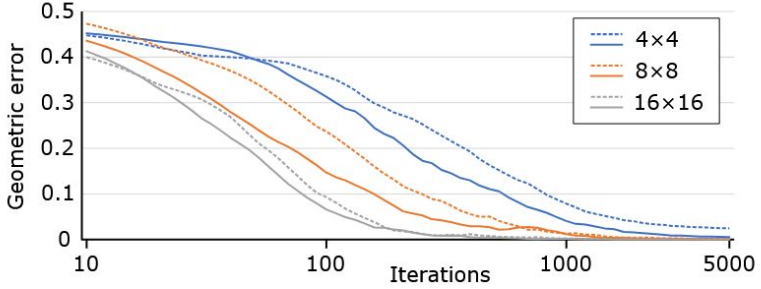
nvdiffrast - backward

allow position training with visibility grads

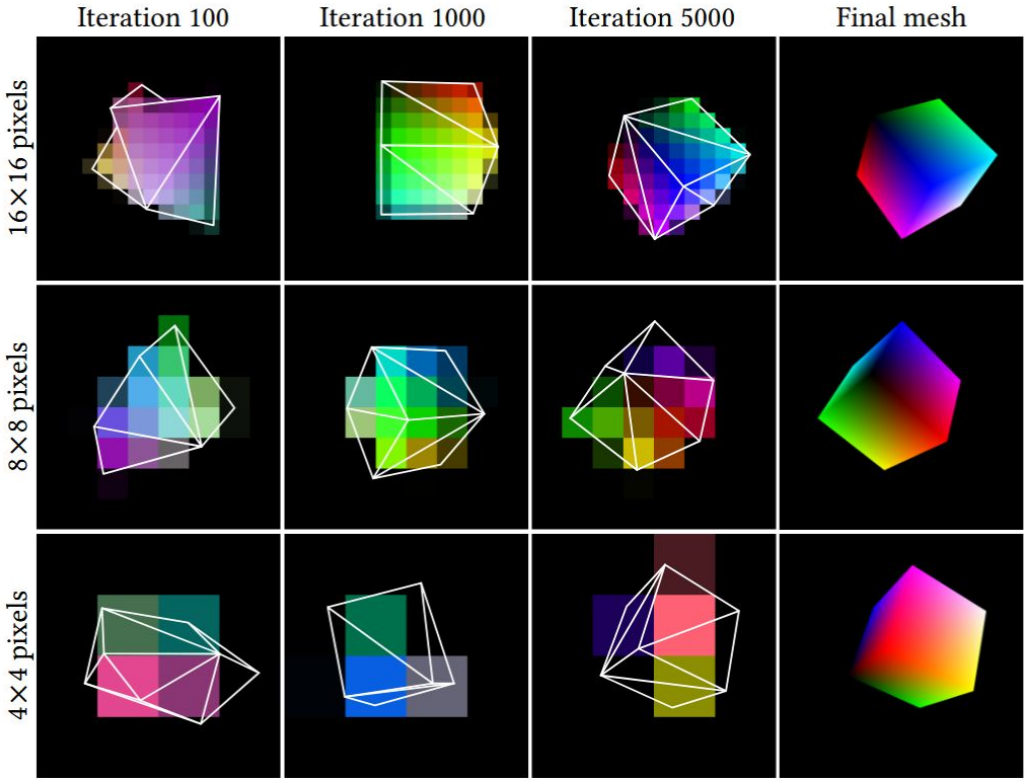


Examples



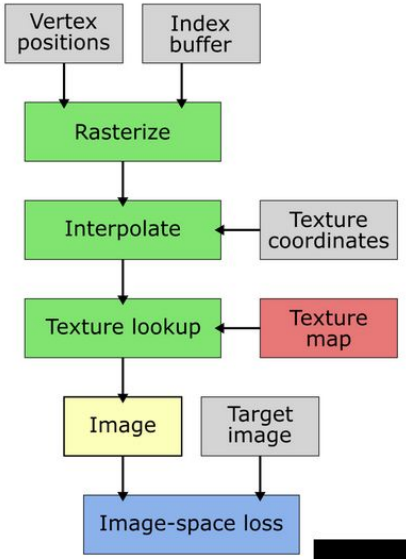


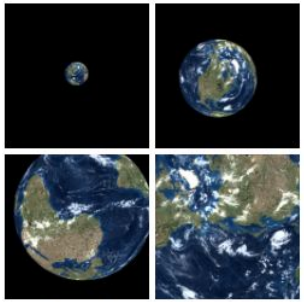
Optimization of vertex positions/colors
Resilience to low-resolution



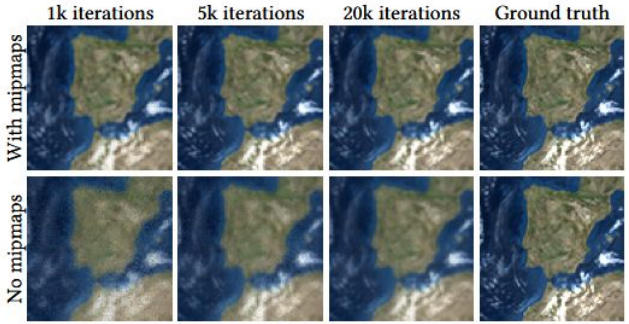
Examples

Optimization of a texture
Importance of mip-maps

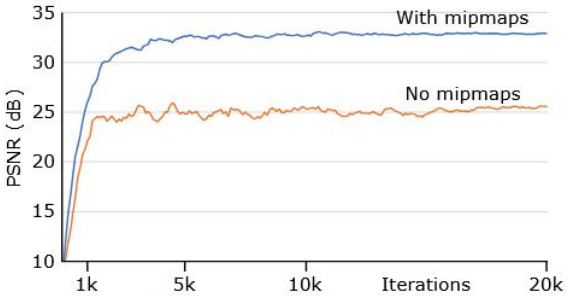




(a) Example reference images

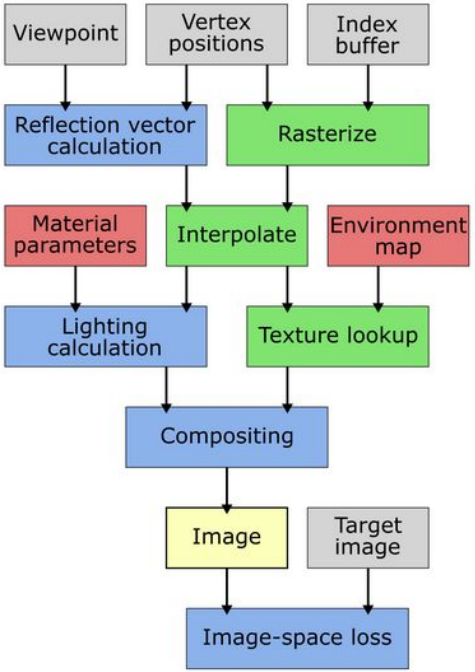


(b) Closeups of learned texture

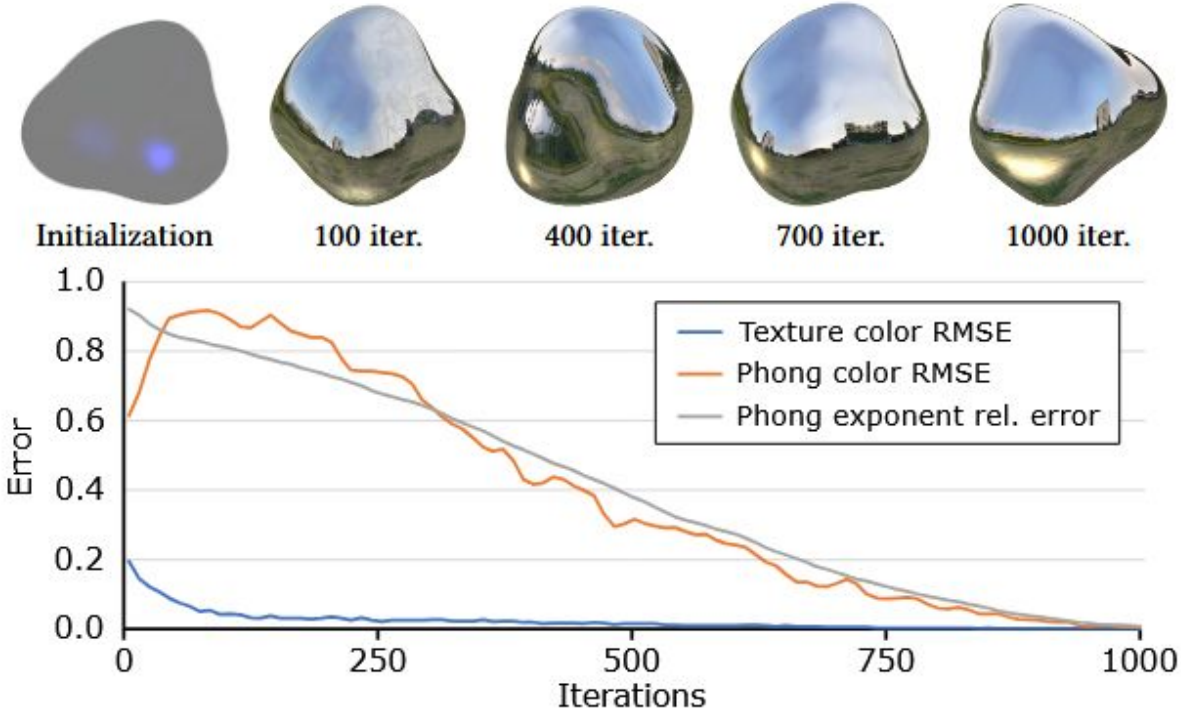


(c) Texture convergence vs. reference

Examples



Optimization of an envmap and material parameters



Times

	Triangles →	284	352	514	1236	4474	5216	5344	10908	21695	25643	43448	91145	196179	308170
	Resolution	Rendering + gradients time* (ms)													
Our method	256×256	2.13	2.03	1.95	1.89	1.95	2.02	2.02	2.08	2.02	2.05	2.00	2.18	2.97	3.12
	512×512	2.26	2.29	2.07	2.08	2.23	2.12	2.08	2.11	2.14	2.20	2.27	2.44	2.66	3.33
	1024×1024	2.70	2.94	2.60	2.56	2.52	2.56	2.61	2.59	2.61	2.64	2.66	3.03	3.36	4.13
	2048×2048	6.21	6.72	4.31	4.95	4.53	4.58	5.31	4.35	4.95	4.46	4.82	5.64	6.46	6.21
	4096×4096	17.73	20.11	12.12	13.18	12.40	12.49	13.29	12.05	12.92	12.39	12.73	14.70	17.73	15.64
SoftRas [2019]	256×256	7.48	6.76	7.84	7.73	18.11	14.29	10.91	29.21	30.93	49.06	65.21	144.57	331.30	788.92
	512×512	10.01	10.42	8.78	10.68	24.05	24.33	24.40	50.09	82.48	99.76	163.51	375.39	865.63	1430.17
	1024×1024	20.73	24.55	15.74	22.00	69.43	73.56	74.07	153.14	277.36	336.54	556.92	1250.74	3012.63	4856.97
	2048×2048	66.25	86.49	46.66	68.43	250.65	276.06	280.30	557.96	1039.49	1234.61	2044.79	4602.11	11487.70	18402.19
	4096×4096	223.88	332.33	163.87	240.71	946.76	1055.14	1082.35	2104.77	4036.47	4768.91	7958.07	17992.60	45499.74	72277.20
PyTorch3D [2020]	256×256	27.12	27.19	26.81	27.95	27.67	27.14	26.94	28.62	27.93	30.08	32.03	37.52	54.82	115.87
	512×512	31.83	30.93	31.08	31.19	31.70	31.82	30.83	32.06	34.84	38.41	41.50	55.53	95.21	158.71
	1024×1024	53.70	53.24	52.03	51.38	52.44	52.17	53.02	53.99	58.98	64.87	83.04	140.34	267.45	438.82
	2048×2048	156.31	153.17	145.34	141.66	144.91	145.48	141.49	148.75	165.77	182.21	259.04	456.07	930.77	1435.20
	4096×4096	571.53	553.69	525.49	513.69	524.58	521.84	508.87	528.99	604.20	677.21	966.76	1754.35	3527.62	5567.26
		Speedup factor													
Our vs SoftRas	256×256	3.51	3.33	4.02	4.09	9.29	7.07	5.40	14.04	15.31	23.93	32.60	66.32	111.55	252.86
	512×512	4.43	4.55	4.24	5.13	10.78	11.48	11.73	23.74	38.54	45.35	72.03	153.85	325.42	429.48
	1024×1024	7.68	8.35	6.05	8.59	27.55	28.73	28.38	59.13	106.27	127.48	209.37	412.79	896.62	1176.02
	2048×2048	10.67	12.87	10.83	13.82	55.33	60.28	52.79	128.27	210.00	276.82	424.23	815.98	1778.28	2963.32
	4096×4096	12.63	16.53	13.52	18.26	76.35	84.48	81.44	174.67	312.42	384.90	625.14	1223.99	2566.26	4621.30
Our vs PyTorch3D	256×256	12.73	13.39	13.75	14.79	14.19	13.44	13.34	13.76	13.83	14.67	16.02	17.21	18.46	37.14
	512×512	14.08	13.51	15.01	15.00	14.22	15.01	14.82	15.19	16.28	17.46	18.28	22.76	35.79	47.66
	1024×1024	19.89	18.11	20.01	20.07	20.81	20.38	20.31	20.85	22.60	24.57	31.22	46.32	79.60	106.25
	2048×2048	25.17	22.79	33.72	28.62	31.99	31.76	26.65	34.20	33.49	40.85	53.74	80.86	144.08	231.11
	4096×4096	32.24	27.53	43.36	38.97	42.30	41.78	38.29	43.90	46.76	54.66	75.94	119.34	198.96	355.96

Questions ?