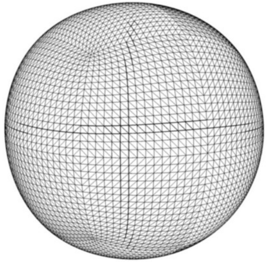
Appearance-Driven Automatic 3D Model Simplification

The Goal

Find the shape and appearance that match as much as possible the reference scene

AKA : Inverse rendering! (big study area)



???

Reference
(can be an image)

Initial guess

Something happens

Optimized mesh!

One approach to solve multiple problems

Inverse rendering applied to…

- Geometric simplification
- Joint shape-appearance simplification
- Simplification of aggregate geometry
- Animation
- Conversion between rendering systems
- Conversion between shape representations

One approach to solve multiple problems

Inverse rendering applied to…

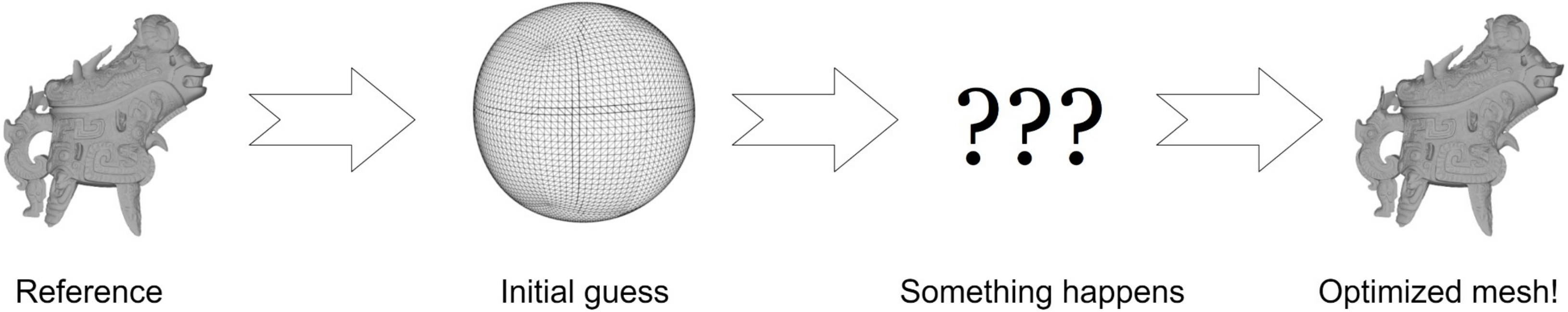
- Geometric simplification : Optimizing for the shape of a lower-resolution mesh
- Joint shape-appearance simplification : optimizing normal maps
- Simplification of aggregate geometry : optimizing foliage
- Animation : optimizing skinning weights
- Conversion between rendering systems : Optimizing the scene representation to match images rendered by an entirely different system
- Conversion between shape representations : Finding a mesh geometry and associated appearance model that captures the appearance of objects given by other shape representations, such as signed distance fields (SDF)

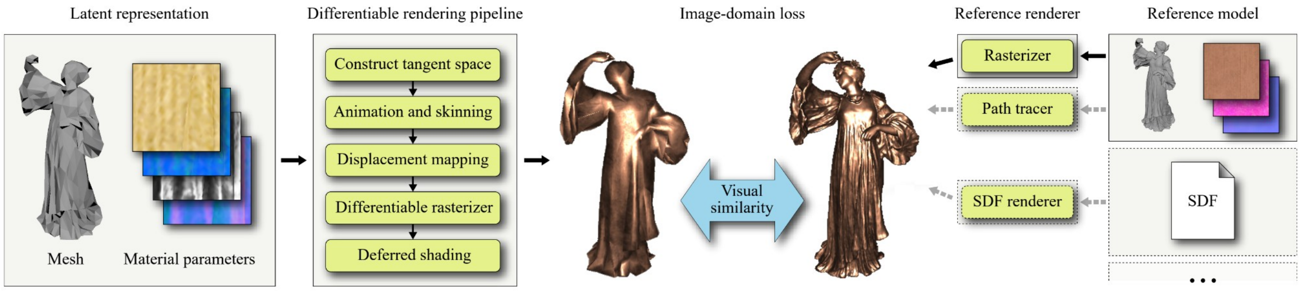
One approach to solve multiple problems

Inverse rendering applied to…

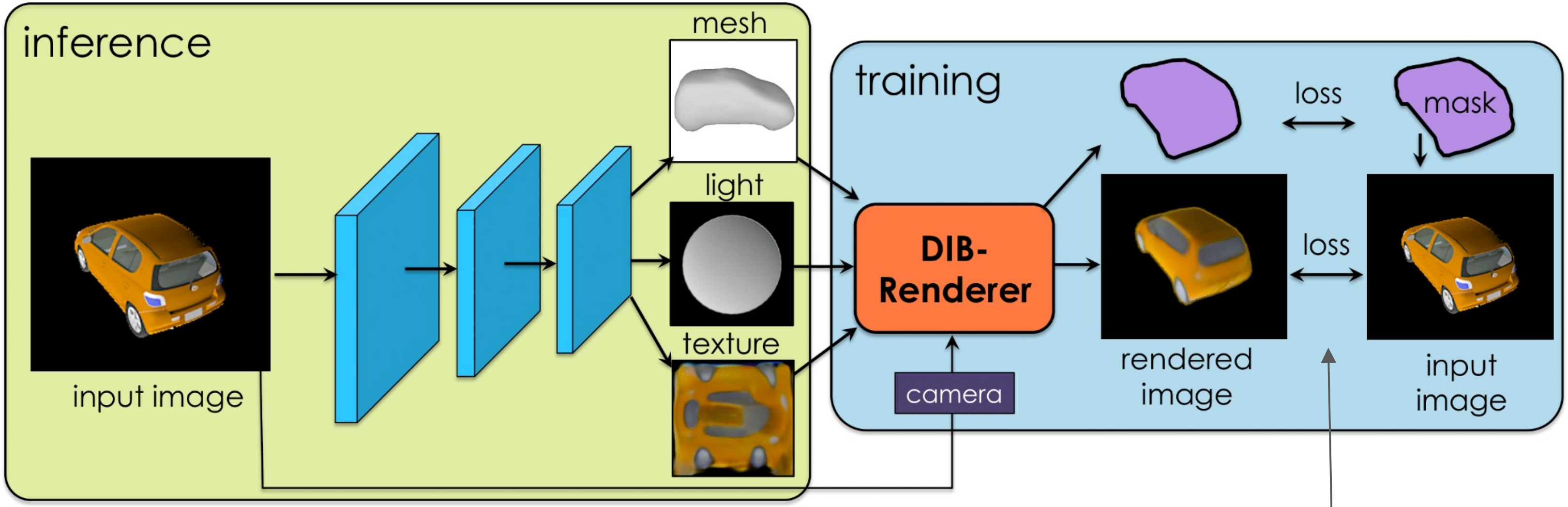
- Geometric simplification
- ~~Joint shape-appearance simplification~~
- Simplification of aggregate geometry
- ~~Animation~~
- ~~Conversion between rendering systems~~
- ~~Conversion between shape representations~~

But how?


???
Reference
Initial guess
Something happens
Optimized mesh!

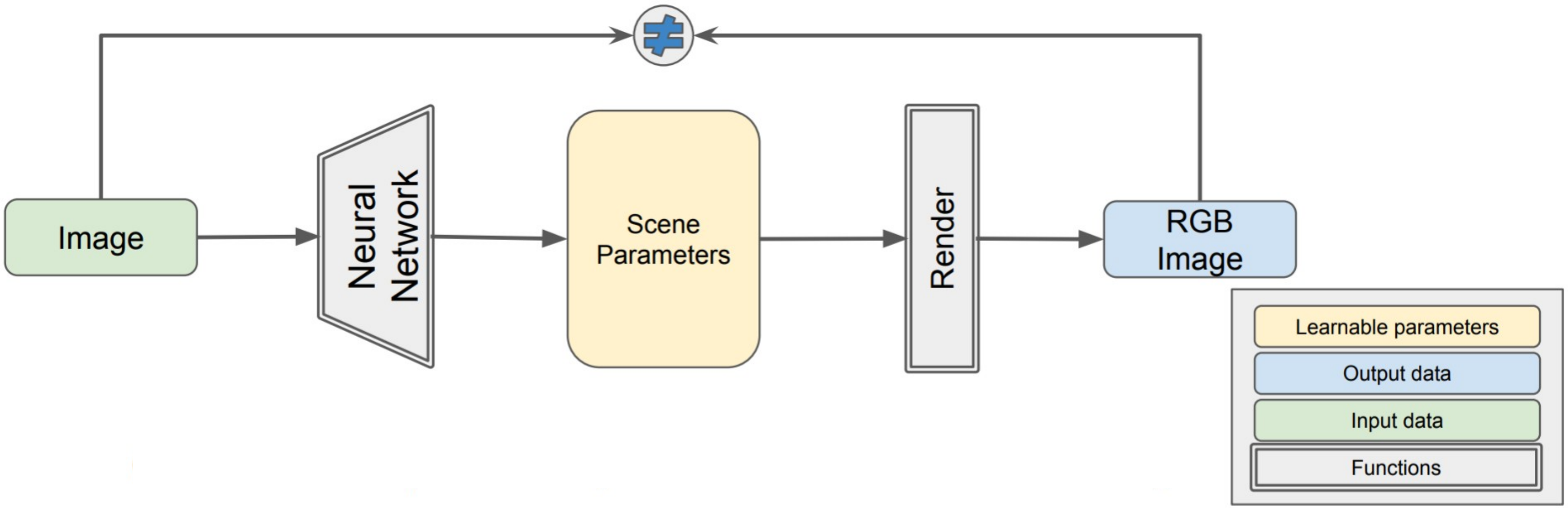

Latent representation
Mesh
Material parameters
Differentiable rendering pipeline
Construct tangent space
Animation and skinning
Displacement mapping
Differentiable rasterizer
Deferred shading
Image-domain loss
Visual similarity
Reference renderer
Rasterizer
Path tracer
SDF renderer
Reference model
SDF
...

Differentiable Rendering

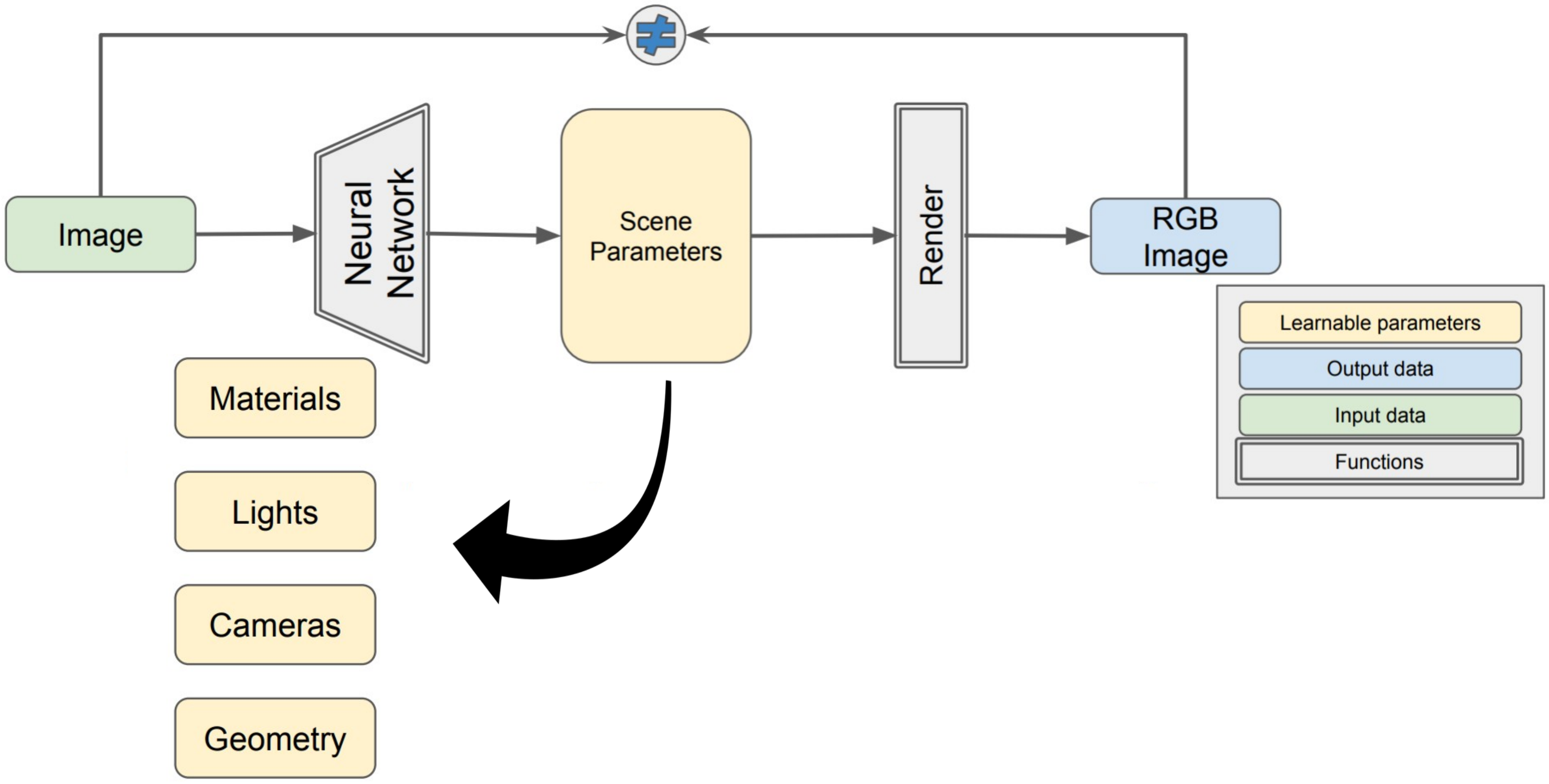


Difference!

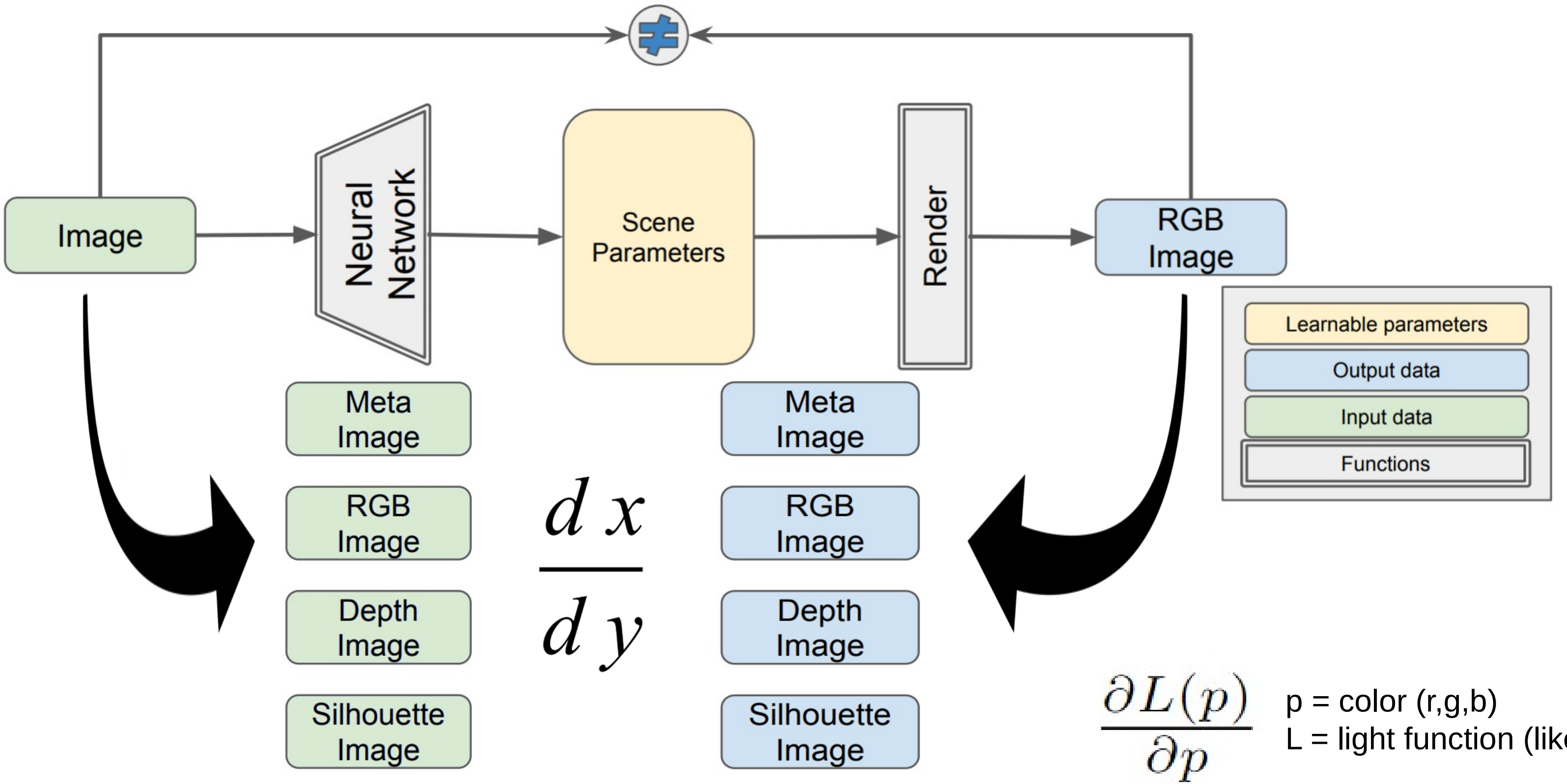
Differentiable Rendering



Differentiable Rendering



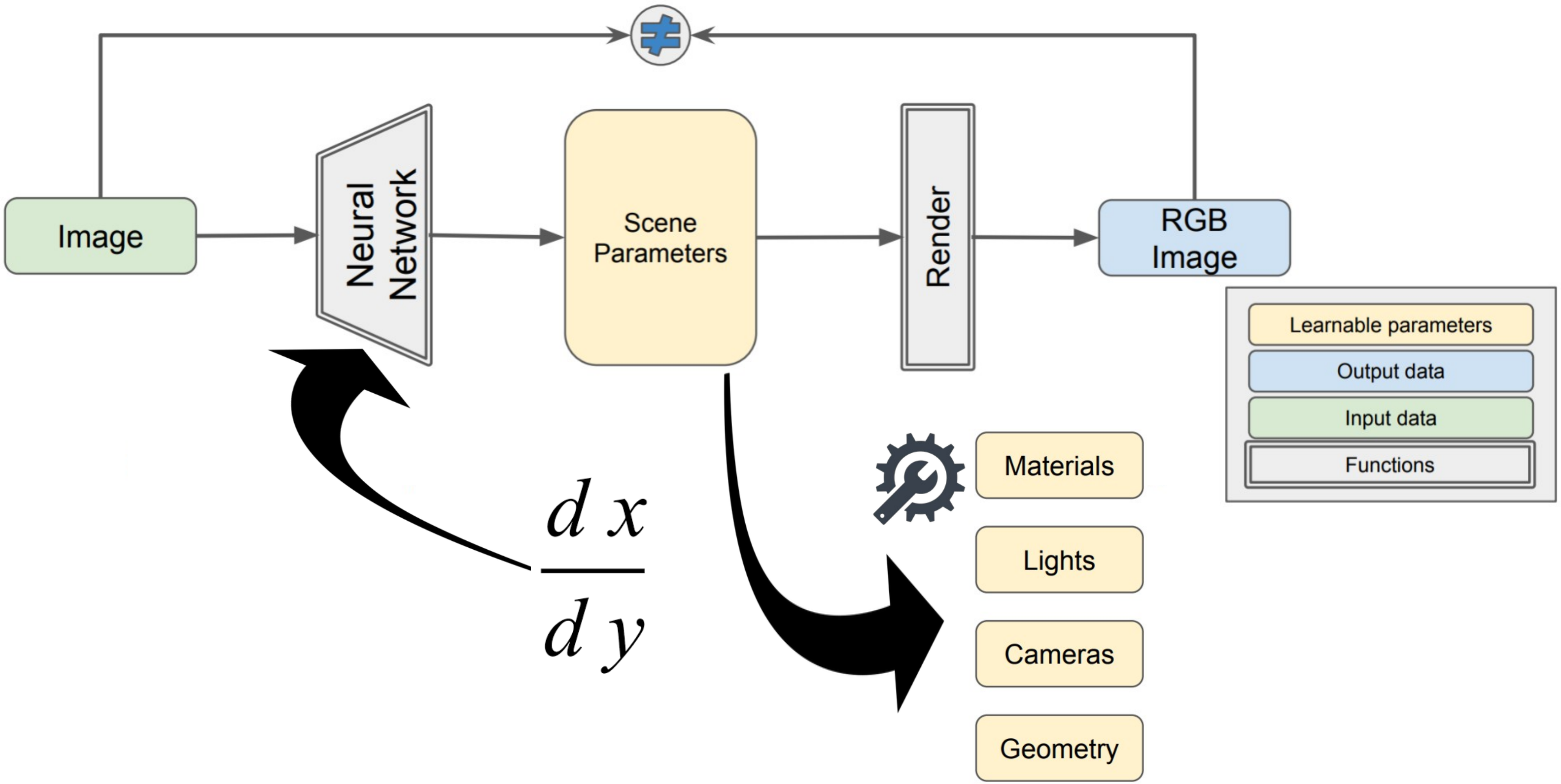
Differentiable Rendering



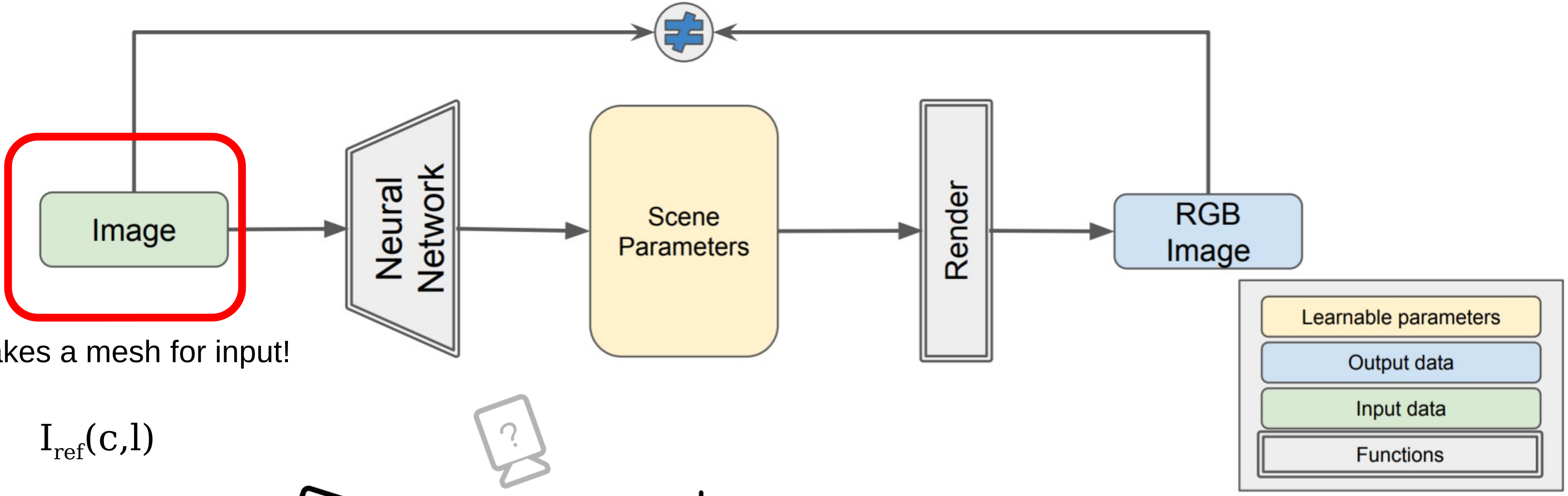
$$\frac{\partial L(p)}{\partial p}$$

p = color (r,g,b)
L = light function (like phong)

Differentiable Rendering



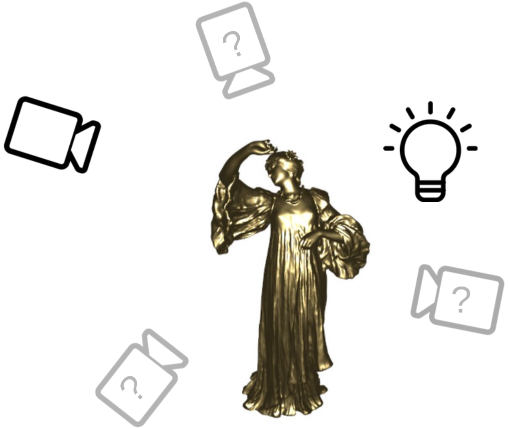
How did they do it?



Takes a mesh for input!

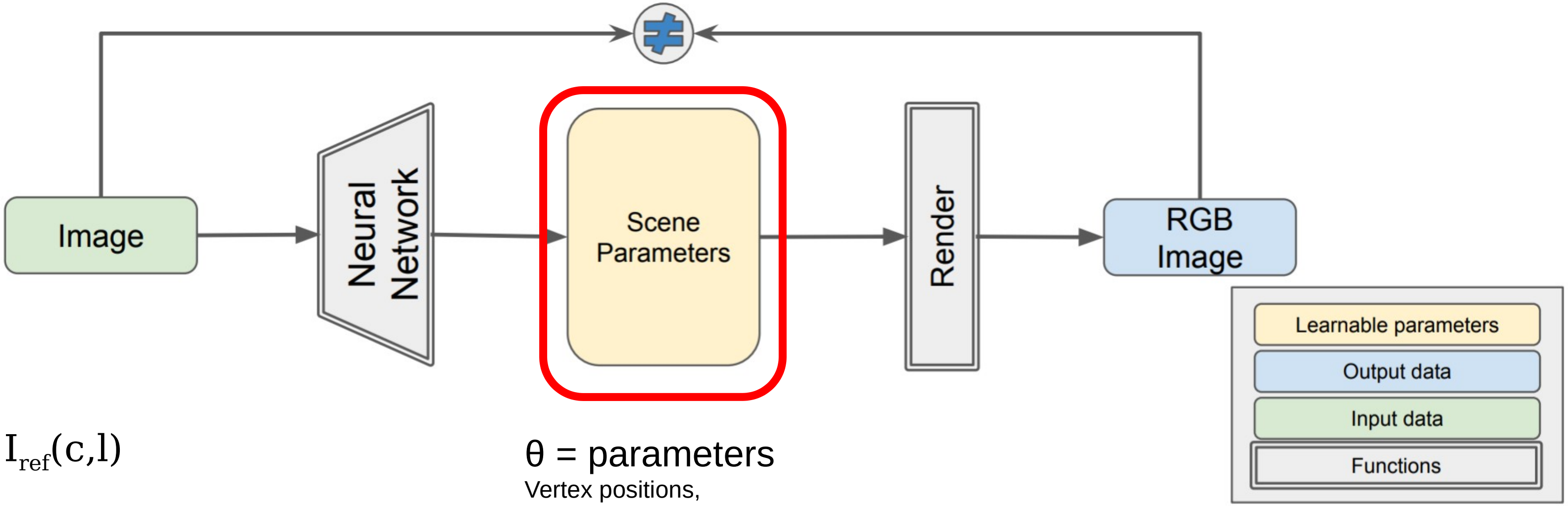
$I_{ref}(c,l)$

c = camera
l = light



= hundreds of images!

How did they do it?

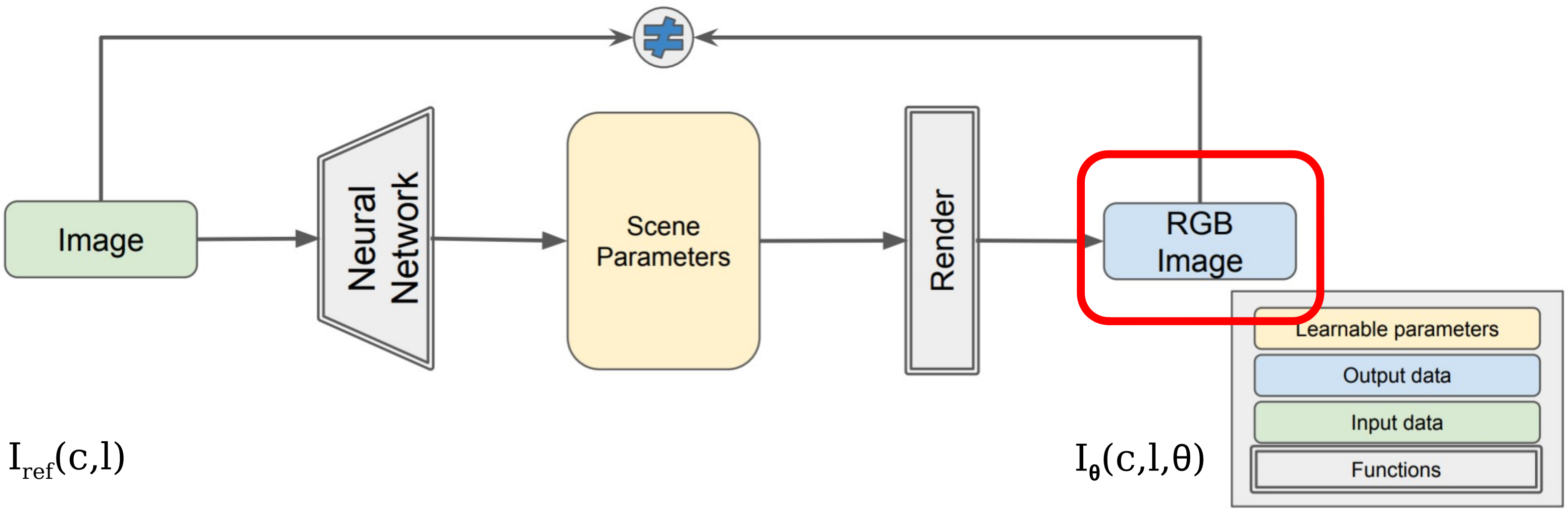


$I_{ref}(c,l)$

c = camera
l = light

θ = parameters
Vertex positions,
Material properties, etc

How did they do it?



$I_{ref}(c,l)$

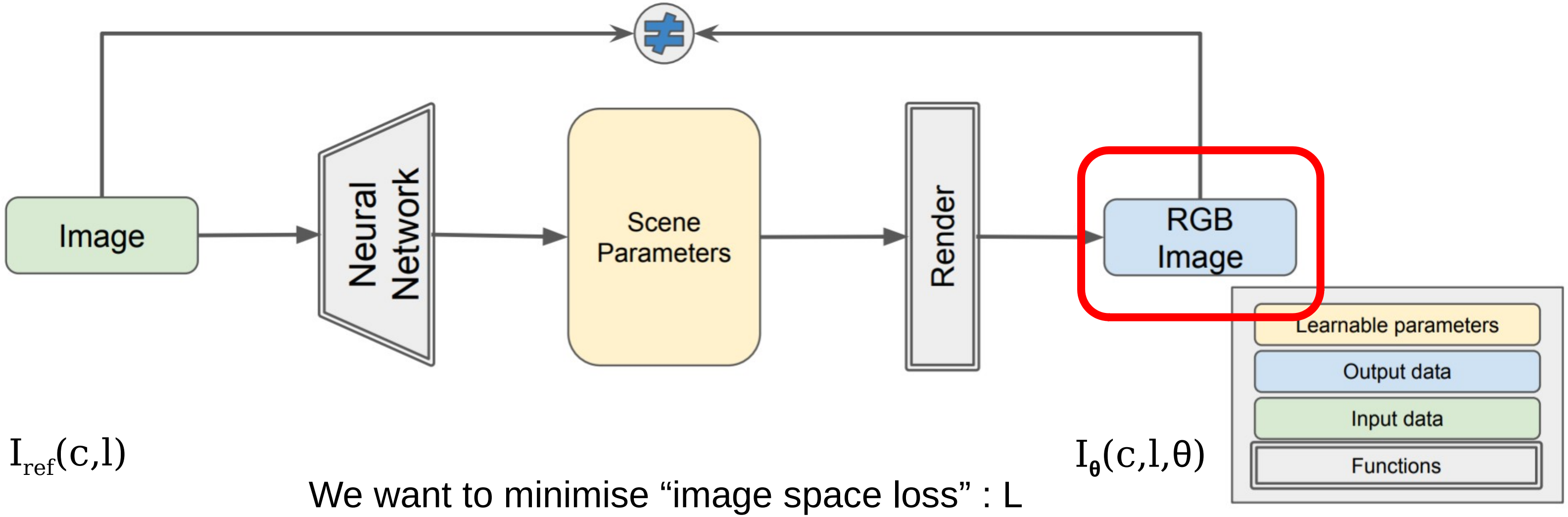
c = camera
l = light

VS

$I_{\theta}(c,l,\theta)$

c = camera
l = light
θ = parameters

How did they do it?



$I_{ref}(c,l)$

$I_{\theta}(c,l,\theta)$

We want to minimise “image space loss” : L

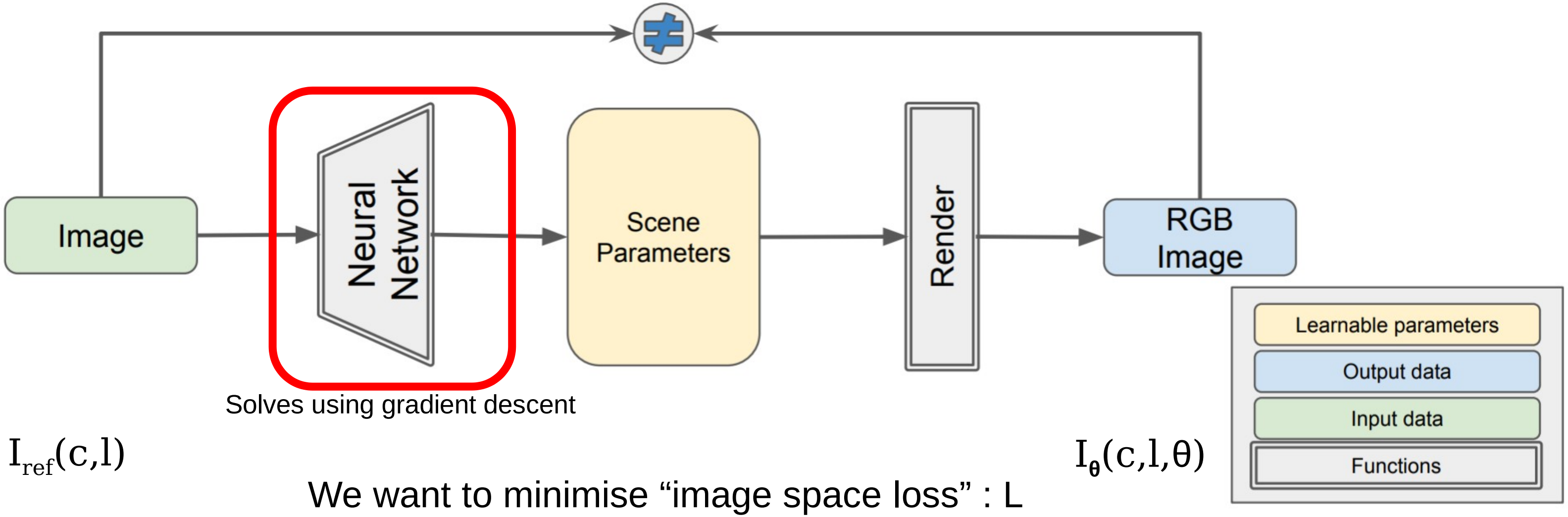
c = camera
l = light

$$L\left(I_{\theta}(c,l), I_{\text{ref}}(c,l)\right)$$

c = camera
l = light
θ = parameters

https://en.wikipedia.org/wiki/Loss_function

How did they do it?



$I_{ref}(c,l)$

$I_{\theta}(c,l,\theta)$

We want to minimise "image space loss" : L

c = camera
l = light

$$\underset{\theta}{\text{argmin}}\ \mathbb{E}_{c,l}\left[L\left(I_\theta(c,l), I_{\text{ref}}(c,l)\right)\right]$$

c = camera
l = light
θ = parameters

Laplacian Mesh processing

O. Sorkine / Laplacian Mesh Processing

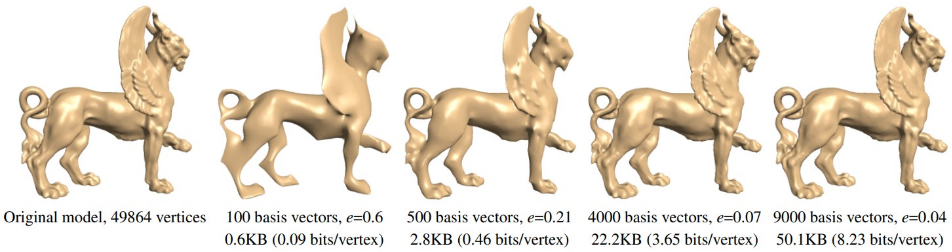
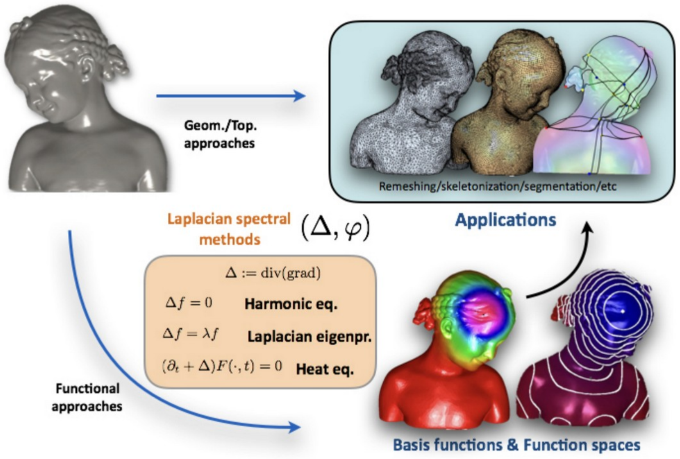


Figure 7: *Reconstruction of the* Feline *model using an increasing number of geometry-aware basis vectors. The sizes of the encoded geometry files are displayed below the models. The letter e denotes the L^2 error value, given in units of 10^{-4}.*

2005 classic paper
from medieval times



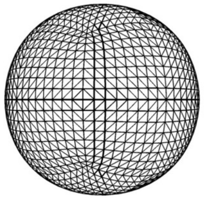
Property	Harm. basis	Hamil. basis	Lapl. eigenf.	Lapl. compr. modes	Hamil. eigenf.	Spectral basis	Diff. basis
Partition of unity	⊗	⊗	⊗	⊗	⊗	⊗	⊗
Non-negativity	●	○	○	○	○	○	●
Intrinsic def.	●	●	●	●	●	●	●
Locality	○	○	○	●	○	⊗	⊗
Orthogonality	⊗	⊗	●	●	●	⊗	⊗
Isometry-inv.	●	○	●	●	○	●	●
Numer. stability	●	●	⊗	⊗	●	●	●
Comput. cost $\mathcal{O}(\cdot)$	n	n	$kn\log n$	$kn\log n$	$kn\log n$	$rn\log n$	$rn\log n$
Storage overhead $\mathcal{O}(\cdot)$	n	n	kn^2	kn	kn^2	n	n

https://www.sciencedirect.com/science/article/abs/pii/S0167839618300815

https://people.eecs.berkeley.edu/~jrs/meshpapers/Sorkine.pdf

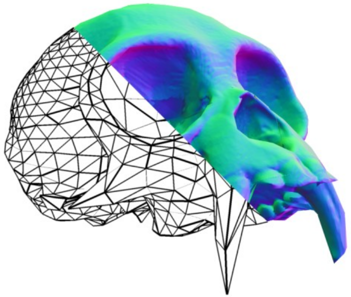
Simplifying a mesh using differentiable rendering

Ref (735k tris)



Initial guess (3k tris)

Differentiable renderer

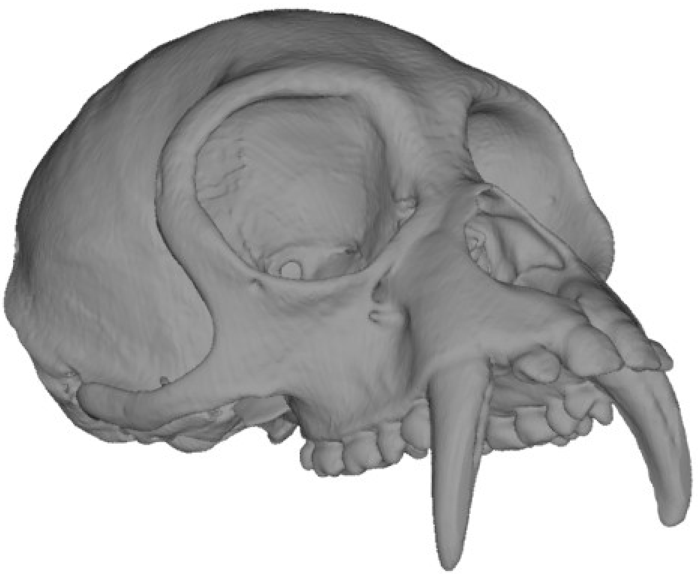


Optimized positions & normals

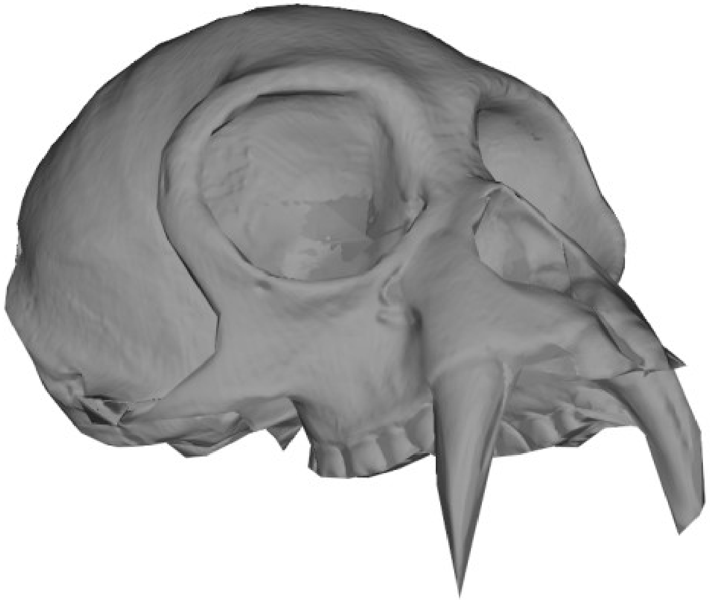
Optimize with parameters :

- vertex position
- tangent space normal maps (used for rendering light)

Simplifying a mesh using differentiable rendering

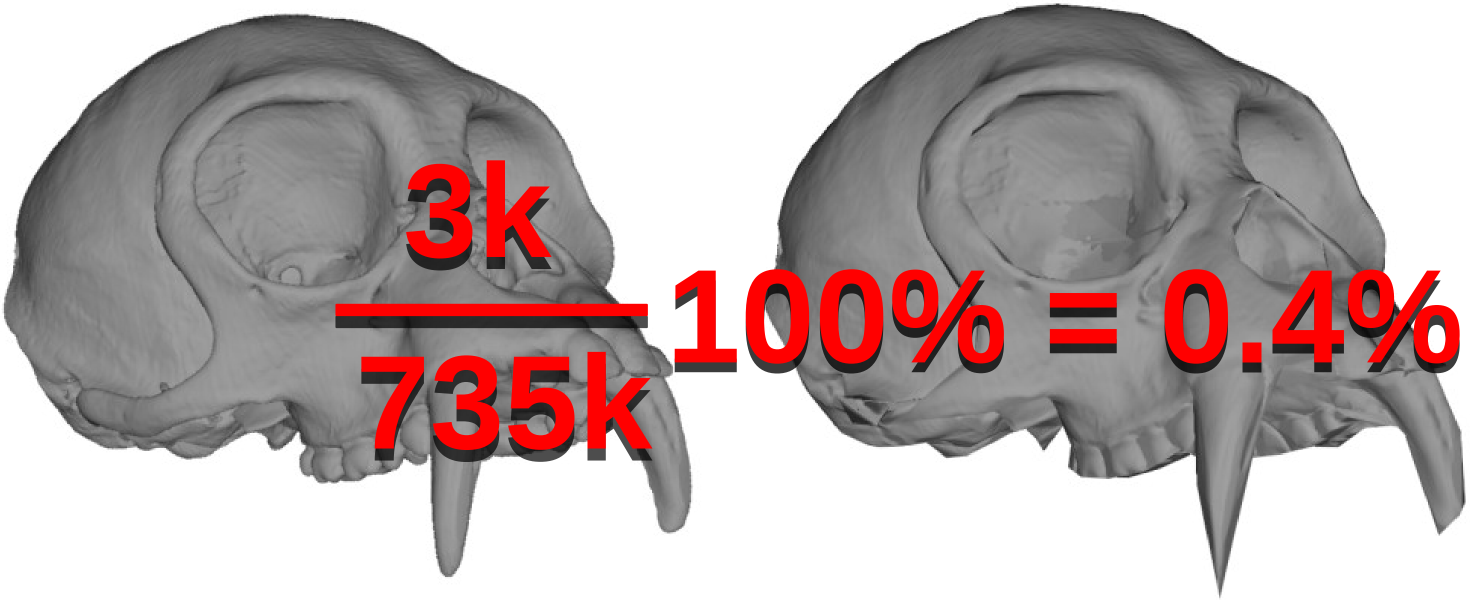


735k Triangles



3k Triangles

Simplifying a mesh using differentiable rendering



735k Triangles

3k Triangles

Conclusion