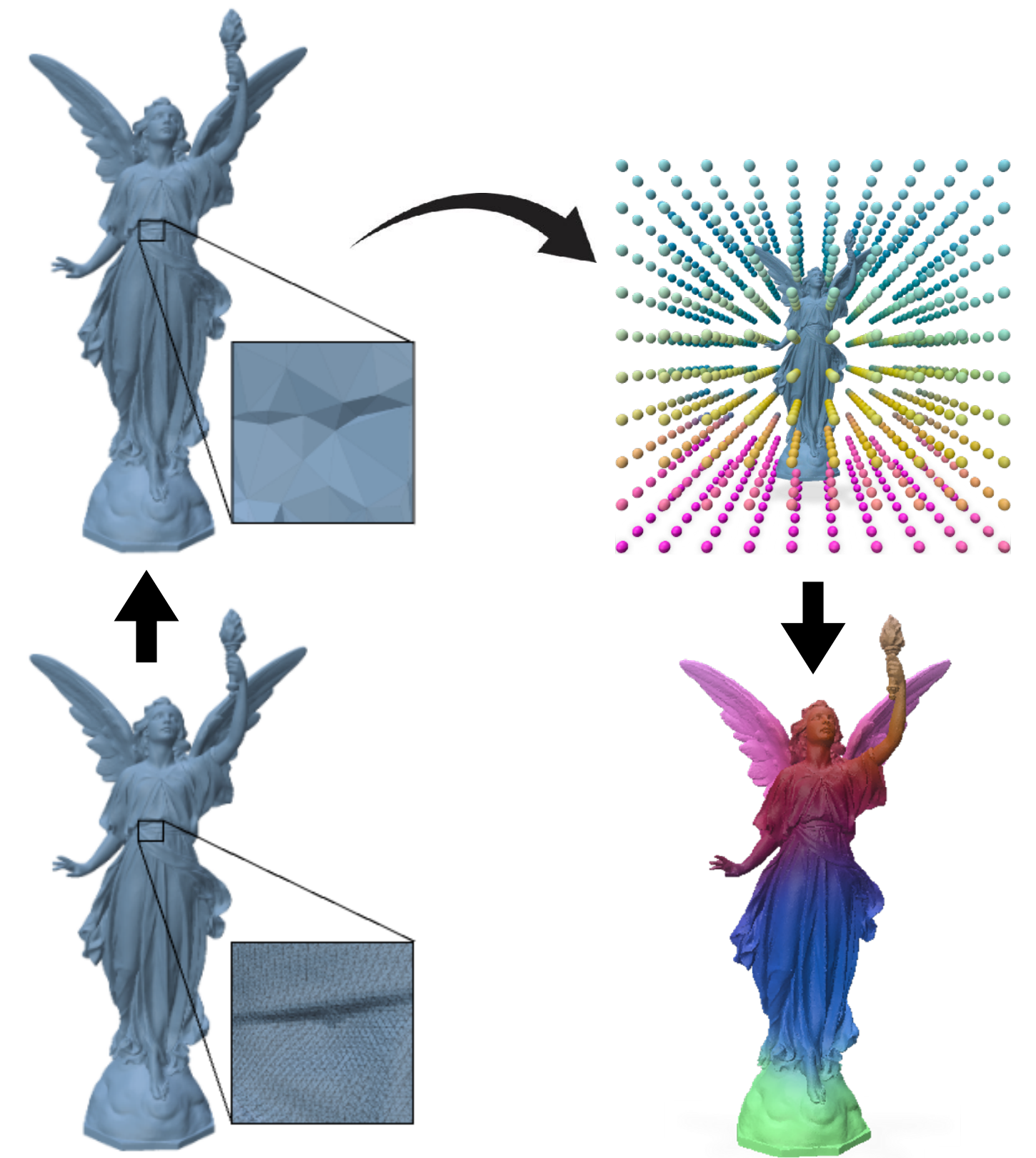
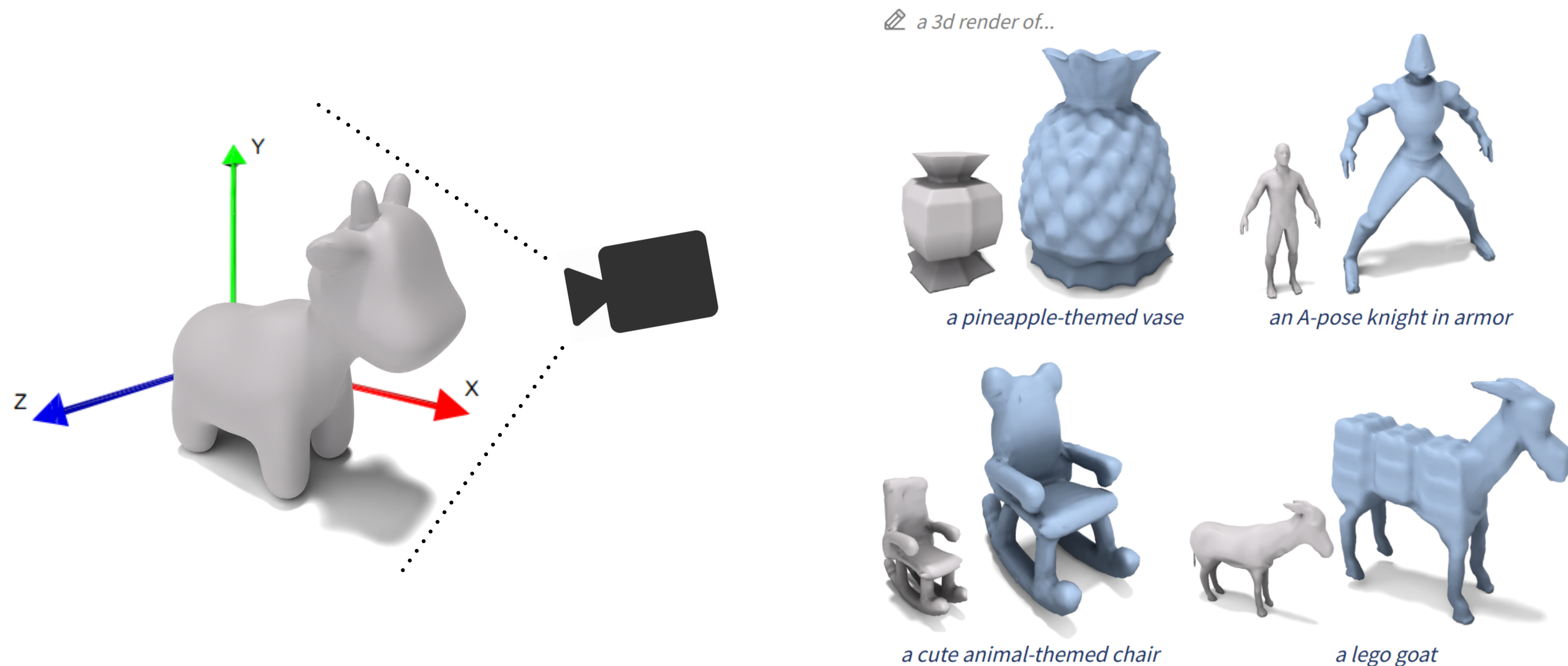


# Geometry Processing from 2D Image Priors



# Dale Decatur, Richard Liu, Nam Anh Dinh

## SGP 2026

# Prevalence of Data-Driven Methods



# Prevalence of Data-Driven Methods



**ChatGPT**

# Prevalence of Data-Driven Methods



# ChatGPT



# Prevalence of Data-Driven Methods



# ChatGPT





# Prevalence of Data-Driven Methods



# ChatGPT



# Gemini



# 2D vs 3D Models



2D



3D

“An astronaut riding a horse”



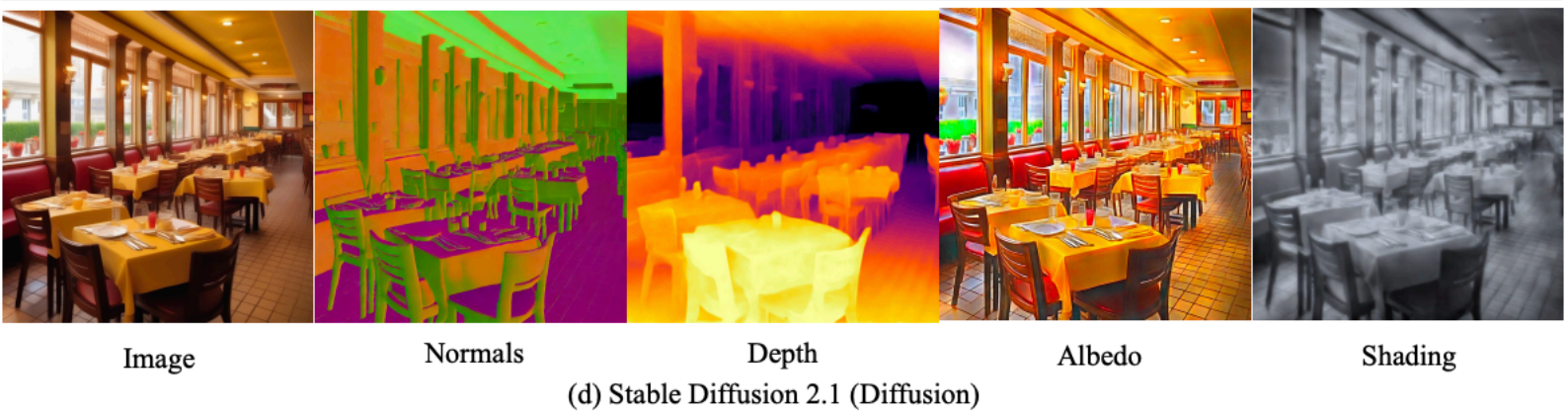
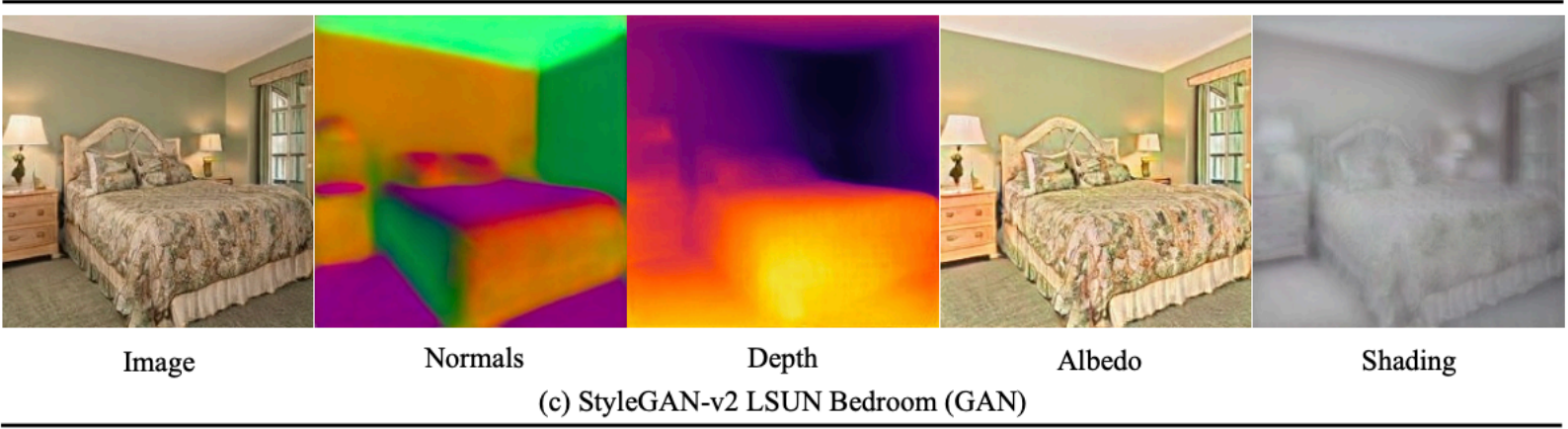
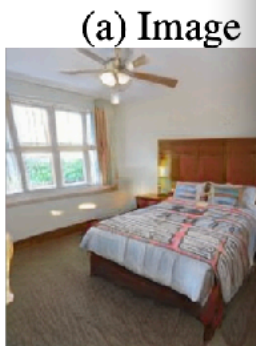
# 3D Understanding in 2D Models

## StyleGAN knows Normal, Depth, Albedo, and More

### GENERATIVE MODELS: WHAT DO THEY KNOW? DO THEY KNOW THINGS? LET'S FIND OUT!

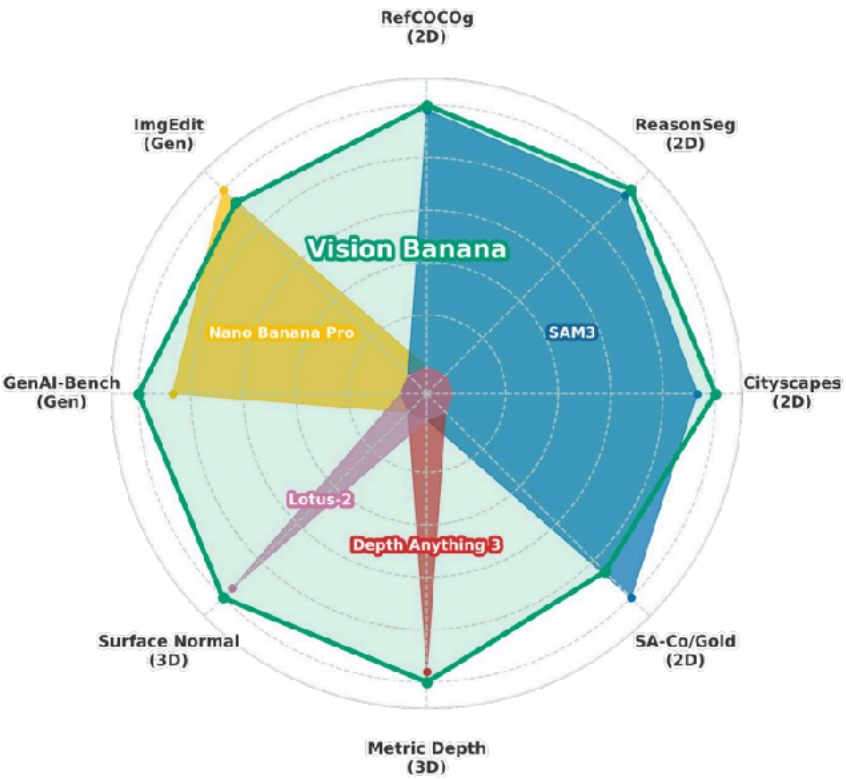
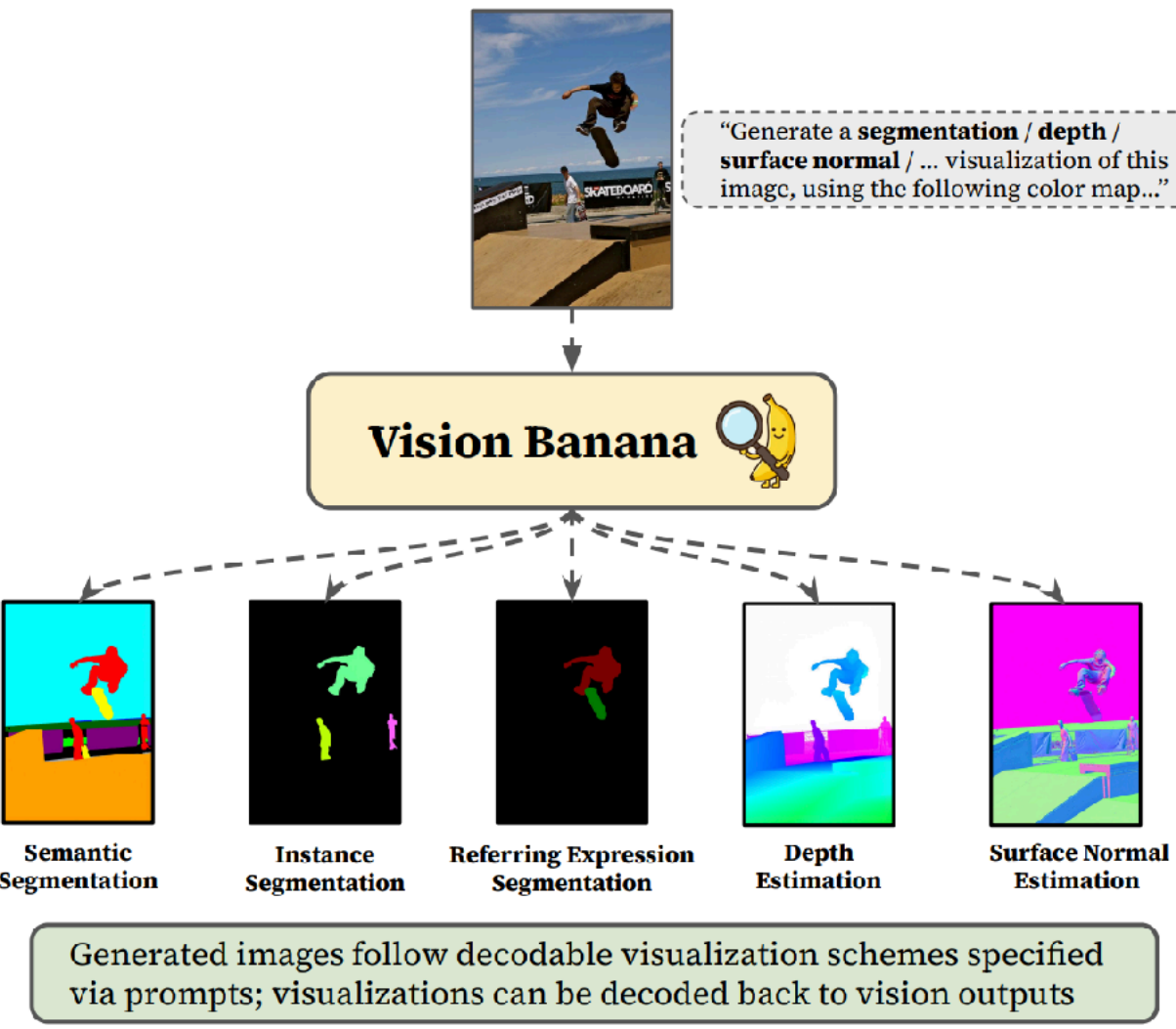
Xiaodan Du<sup>1</sup> Nicholas Kolkin<sup>2</sup> Gregory Shakhnarovich<sup>1</sup> Anand Bhattad<sup>1</sup>

<sup>1</sup> Toyota Technological Institute at Chicago, <sup>2</sup> Adobe Research  
<sup>1</sup> {xdu, greg, bhattad}@ttic.edu, <sup>2</sup> kolkin@adobe.com



## Shadows Don't Lie and Lines Can't Bend! Generative Models don't know Projective Geometry...for now

Ayush Sarkar<sup>\*1</sup> Hanlin Mai<sup>\*1</sup> Amitabh Mahapatra<sup>\*1</sup> Svetlana Lazebnik<sup>1</sup>  
D.A. Forsyth<sup>1</sup> Anand Bhattad<sup>2</sup>  
<sup>1</sup>University of Illinois Urbana-Champaign <sup>2</sup>Toyota Technological Institute at Chicago



**Vision Banana** is a generalist vision model in both visual generation and understanding, surpassing or rivaling specialist models.



# 3D Understanding in 2D Models

2D generative models understand 3D quantities (depth, normal, etc.)

## StyleGAN knows Normal, Depth, Albedo, and More

### GENERATIVE MODELS: WHAT DO THEY KNOW? DO THEY KNOW THINGS? LET'S FIND OUT!

Xiaodan Du<sup>1</sup> Nicholas Kolkin<sup>2</sup> Gregory Shakhnarovich<sup>1</sup> Anand Bhattad<sup>1</sup>

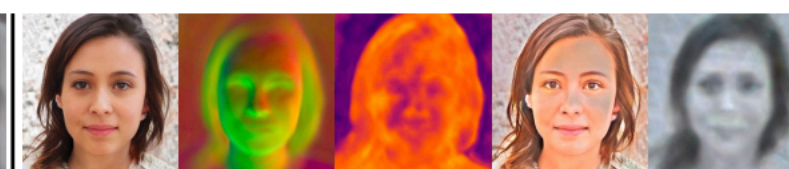
<sup>1</sup> Toyota Technological Institute at Chicago, <sup>2</sup> Adobe Research

<sup>1</sup> {xdu, greg, bhattad}@ttic.edu, <sup>2</sup> kolkin@adobe.com

(a) Image



Image Normals Depth Albedo Shading  
(a) VQGAN FFHQ (Autoregressive)



(b) StyleGAN-XL FFHQ (GAN)

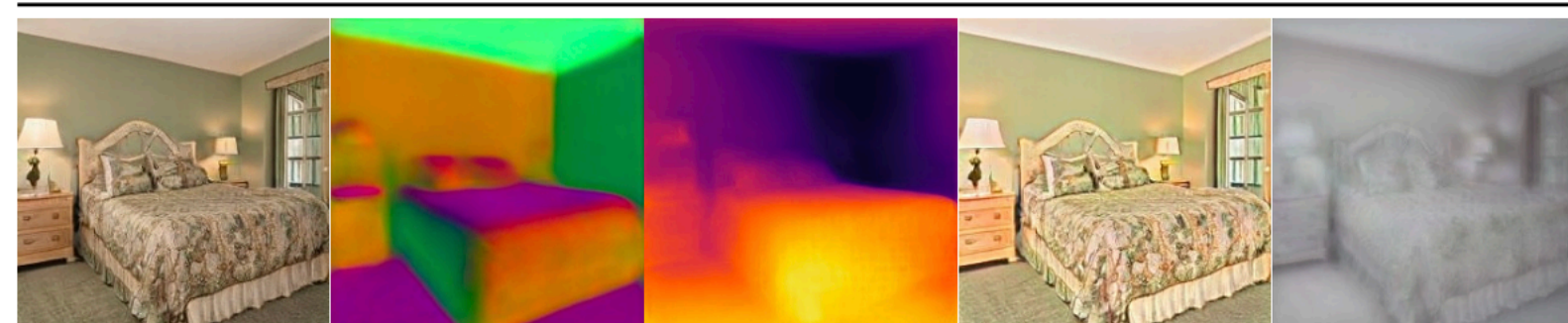


Image Normals Depth Albedo Shading  
(c) StyleGAN-v2 LSUN Bedroom (GAN)

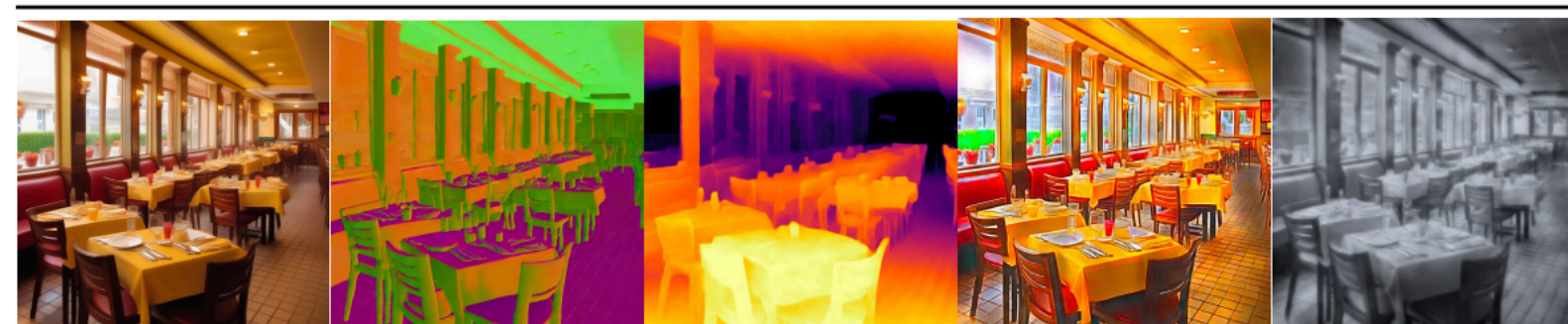


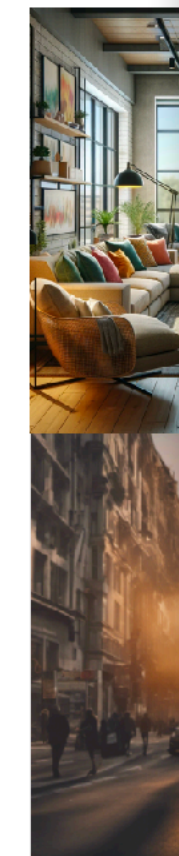
Image Normals Depth Albedo Shading  
(d) Stable Diffusion 2.1 (Diffusion)

## Shadows Don't Lie and Lines Can't Bend! Generative Models don't know Projective Geometry...for now

Ayush Sarkar<sup>\*1</sup> Hanlin Mai<sup>\*1</sup> Amitabh Mahapatra<sup>\*1</sup> Svetlana Lazebnik<sup>1</sup>

D.A. Forsyth<sup>1</sup> Anand Bhattad<sup>2</sup>

<sup>1</sup>University of Illinois Urbana-Champaign <sup>2</sup>Toyota Technological Institute at Chicago



Generative



"Generate a **segmentation** / **depth** / **surface normal** / ... visualization of this image, using the following color map..."

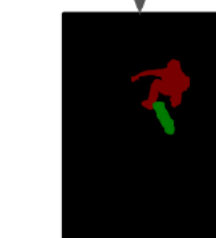
Vision Banana



Semantic Segmentation



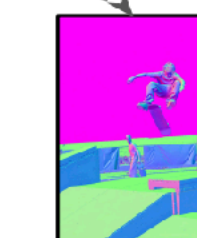
Instance Segmentation



Referring Expression Segmentation

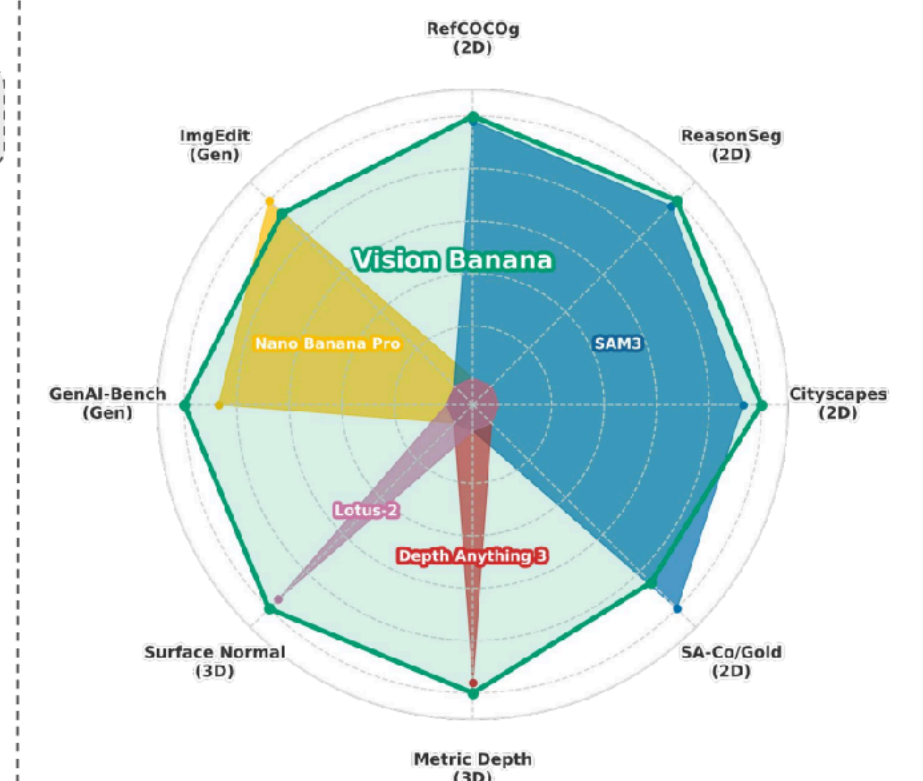


Depth Estimation



Surface Normal Estimation

Generated images follow decodable visualization schemes specified via prompts; visualizations can be decoded back to vision outputs



**Vision Banana** is a generalist vision model in both visual generation and understanding, surpassing or rivaling specialist models.

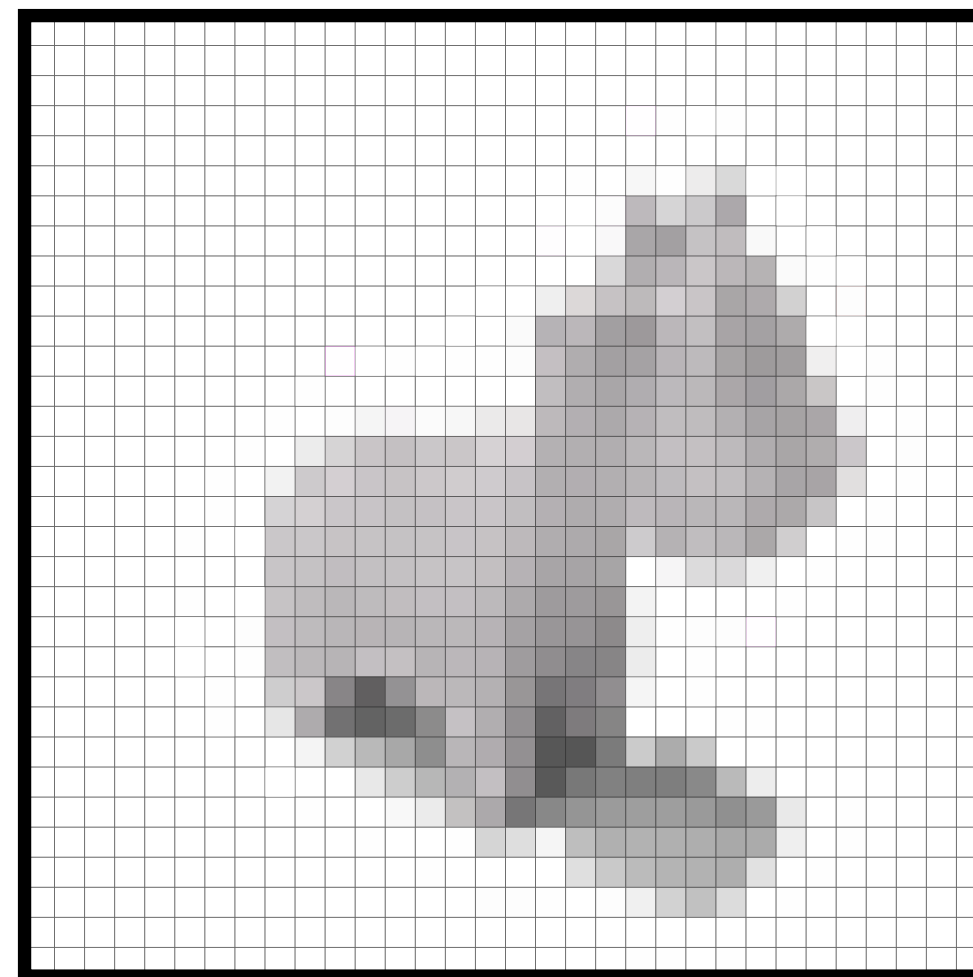


# Bridging the Gap Between 2D and 3D

The render is the key bridge between 2D and 3D



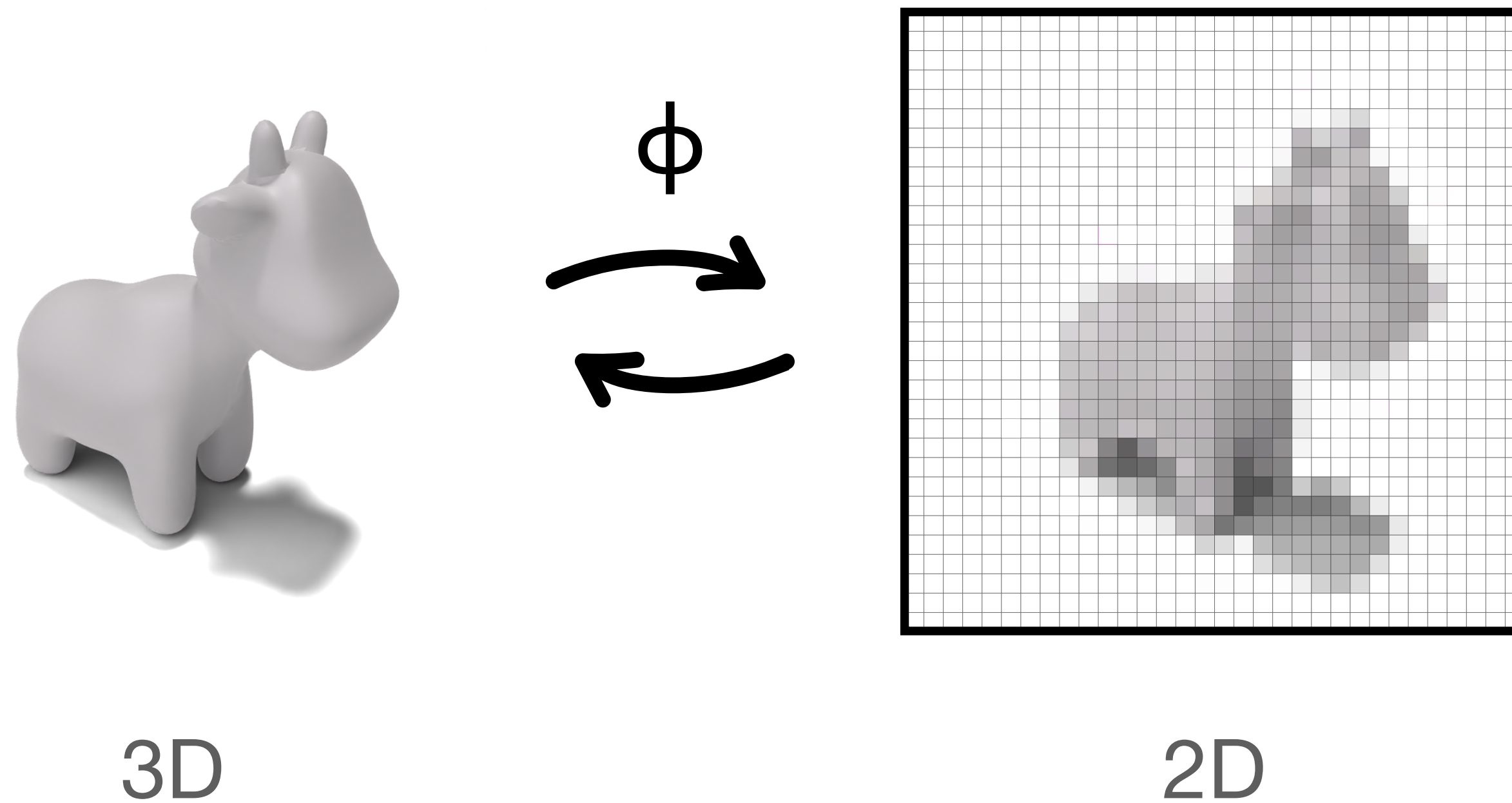
3D



2D

# Bridging the Gap Between 2D and 3D

The render is the key bridge between 2D and 3D



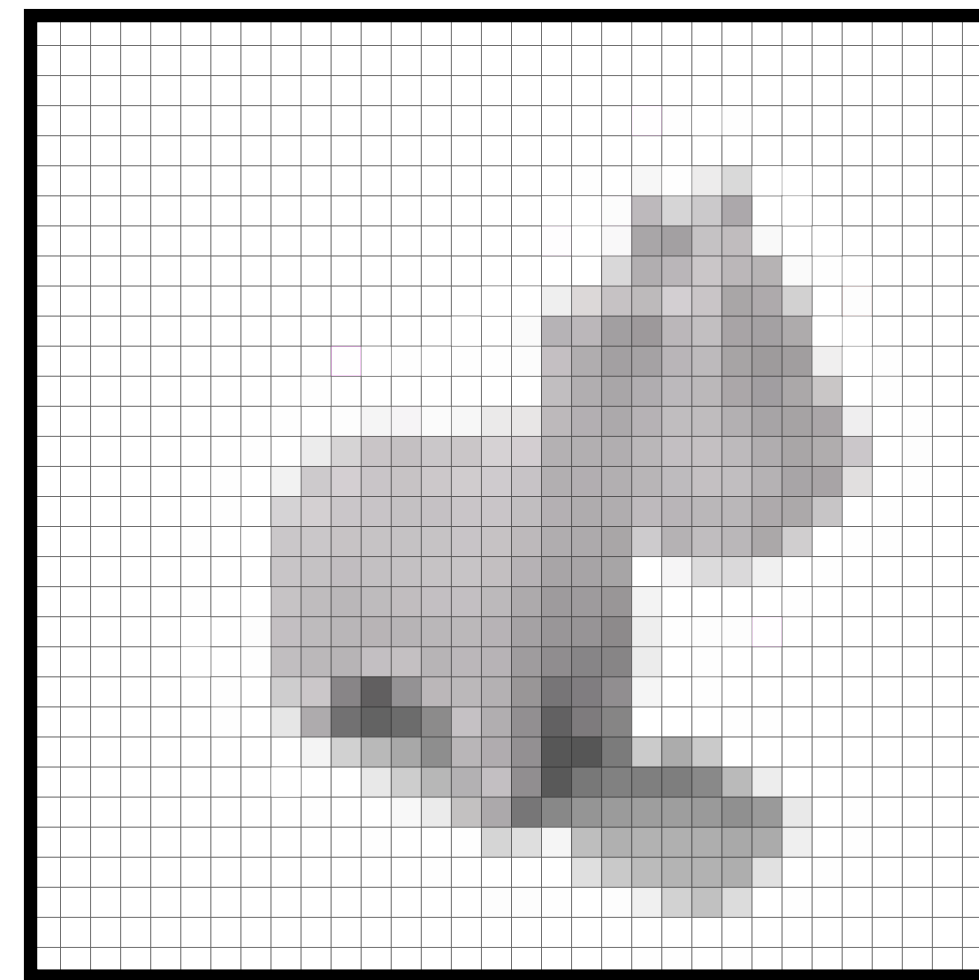
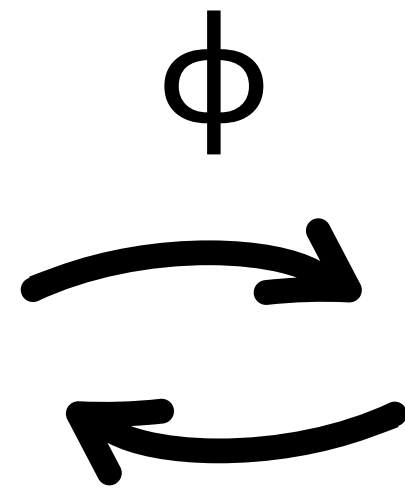


# Bridging the Gap Between 2D and 3D

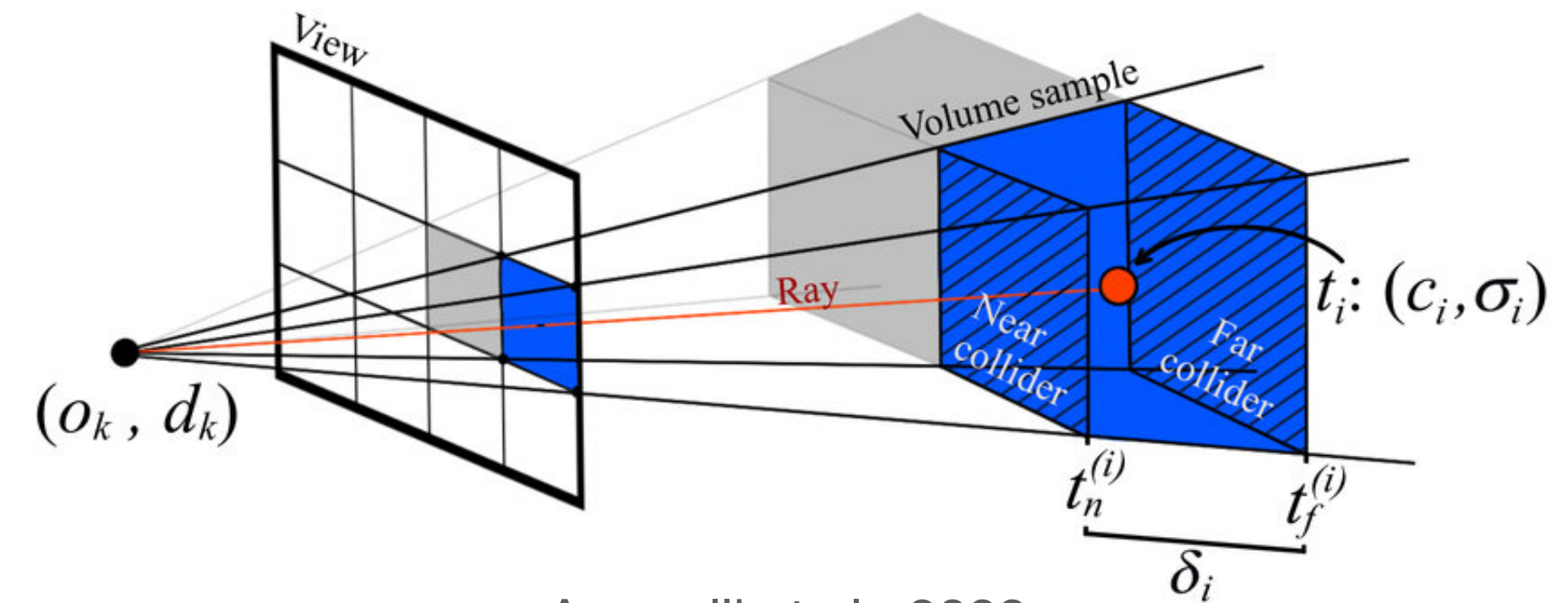
The render is the key bridge between 2D and 3D



3D



2D



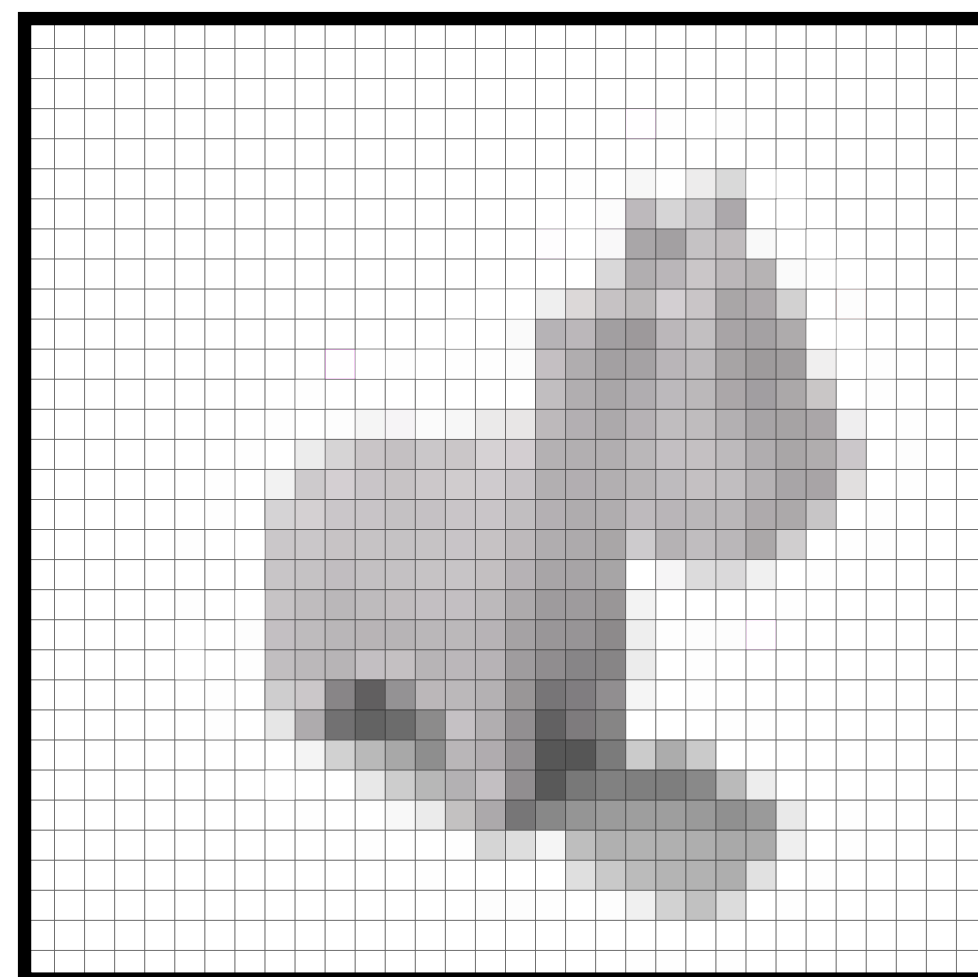
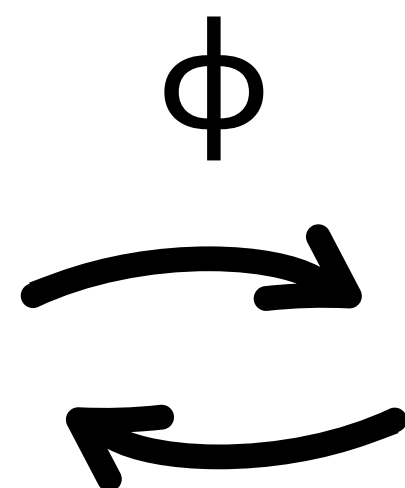
Azzarelli et al., 2023

# Bridging the Gap Between 2D and 3D

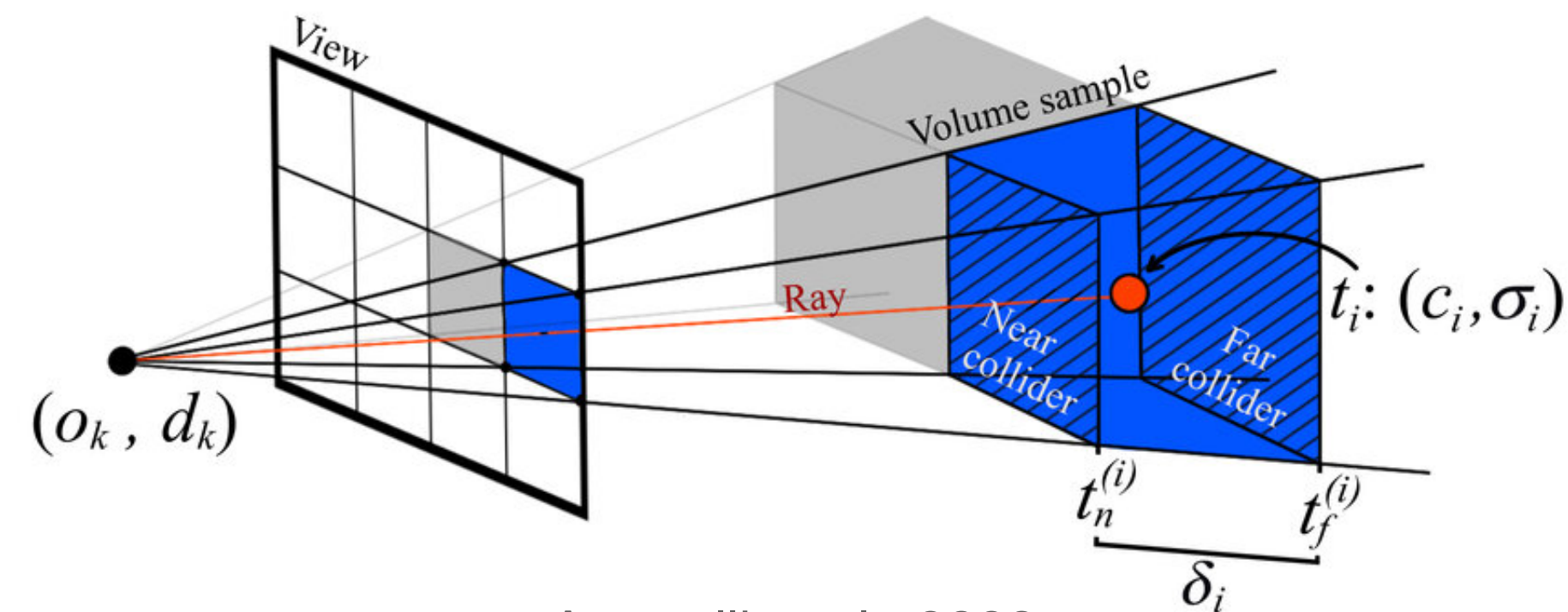
The render is the key bridge between 2D and 3D



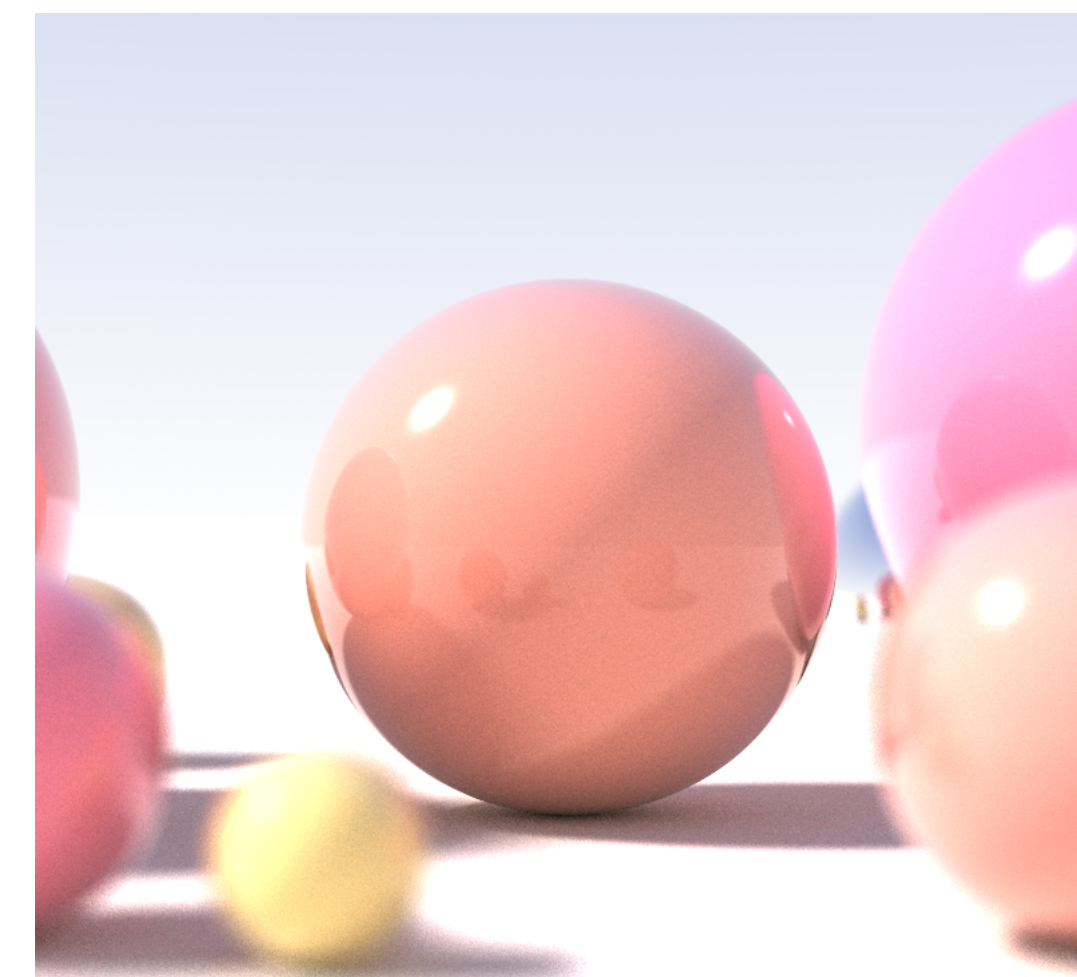
3D



2D



Azzarelli et al., 2023



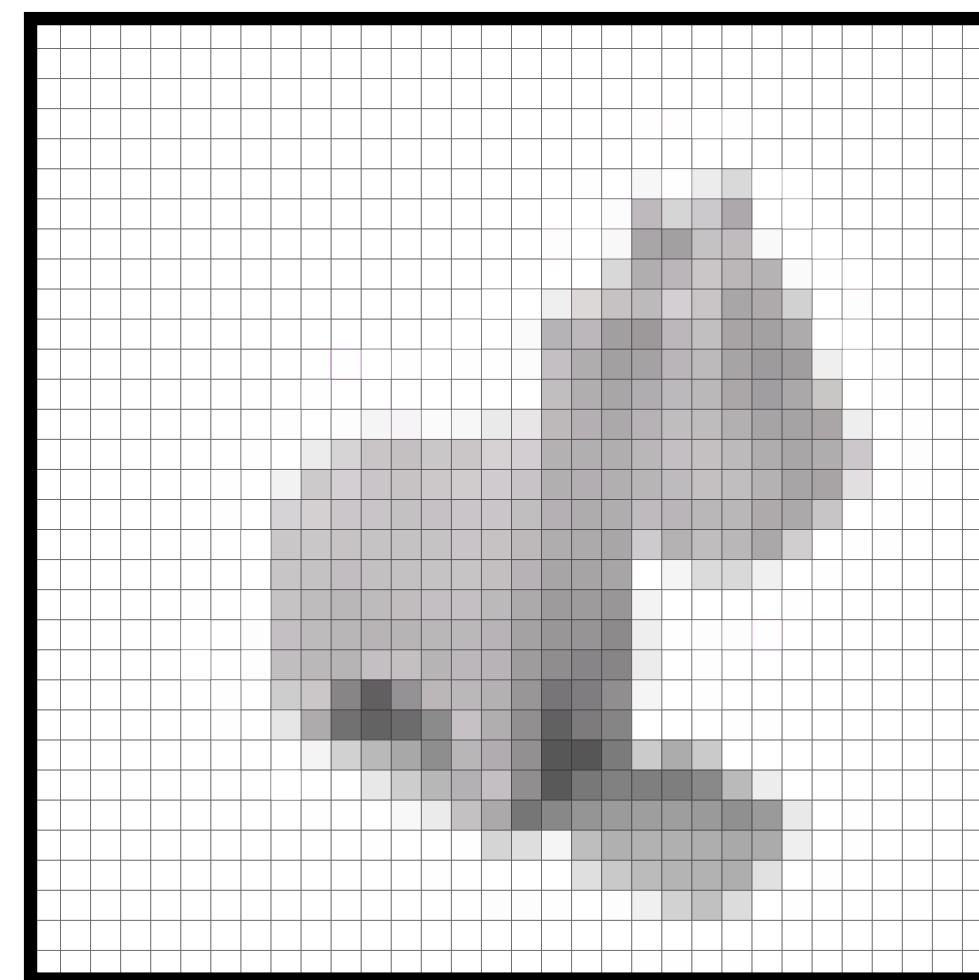
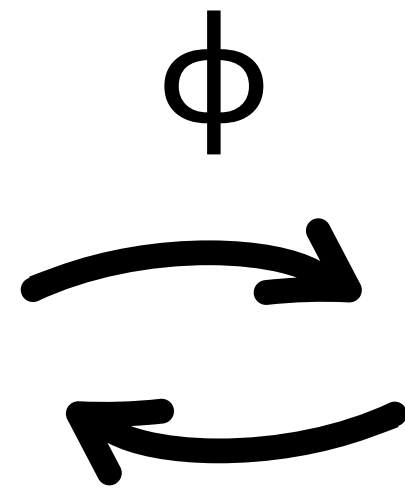


# Bridging the Gap Between 2D and 3D

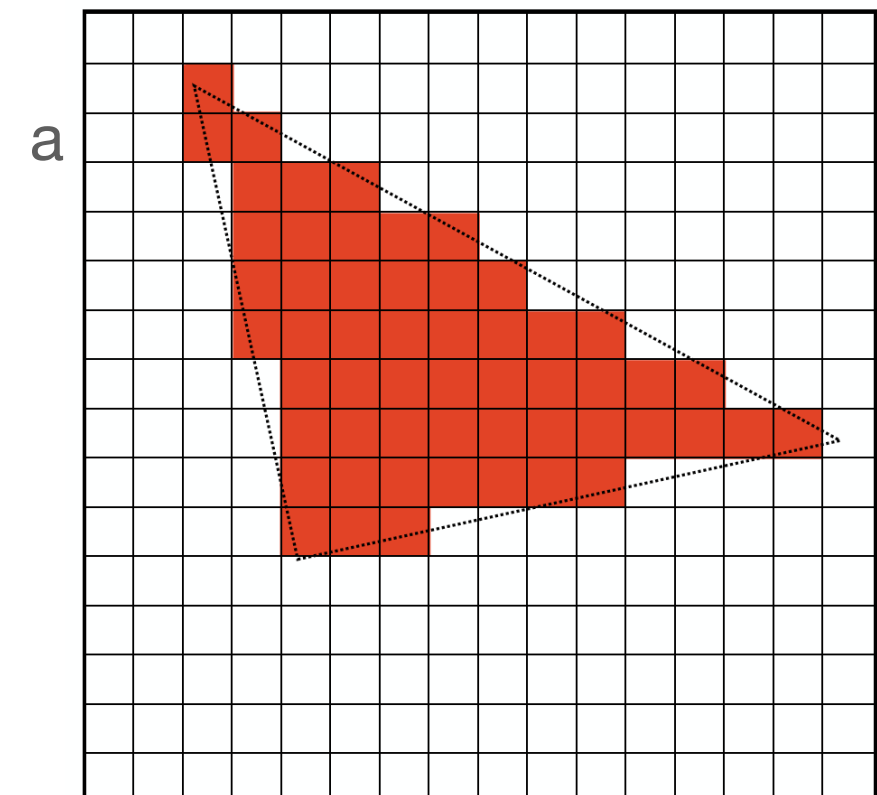
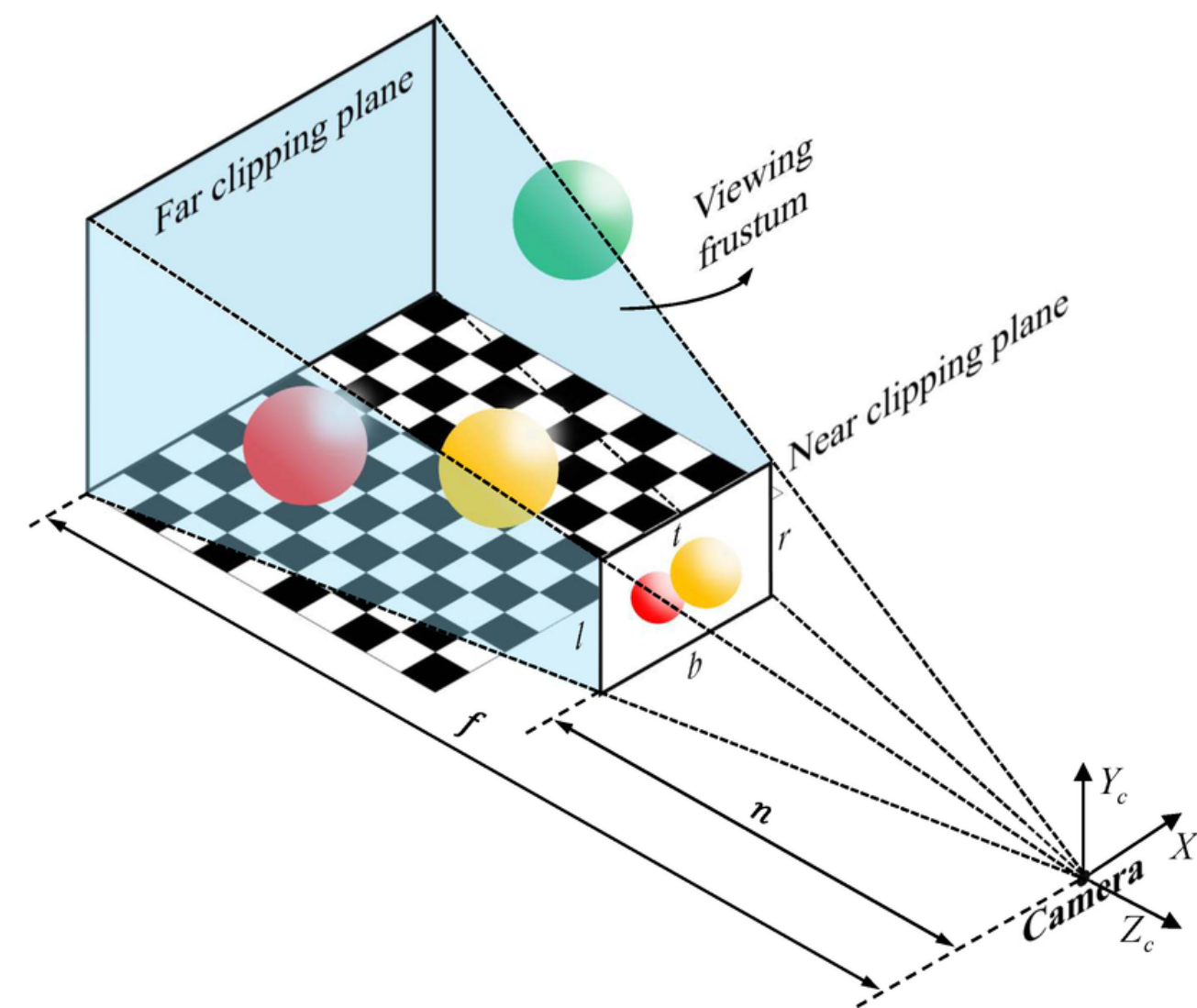
The render is the key bridge between 2D and 3D



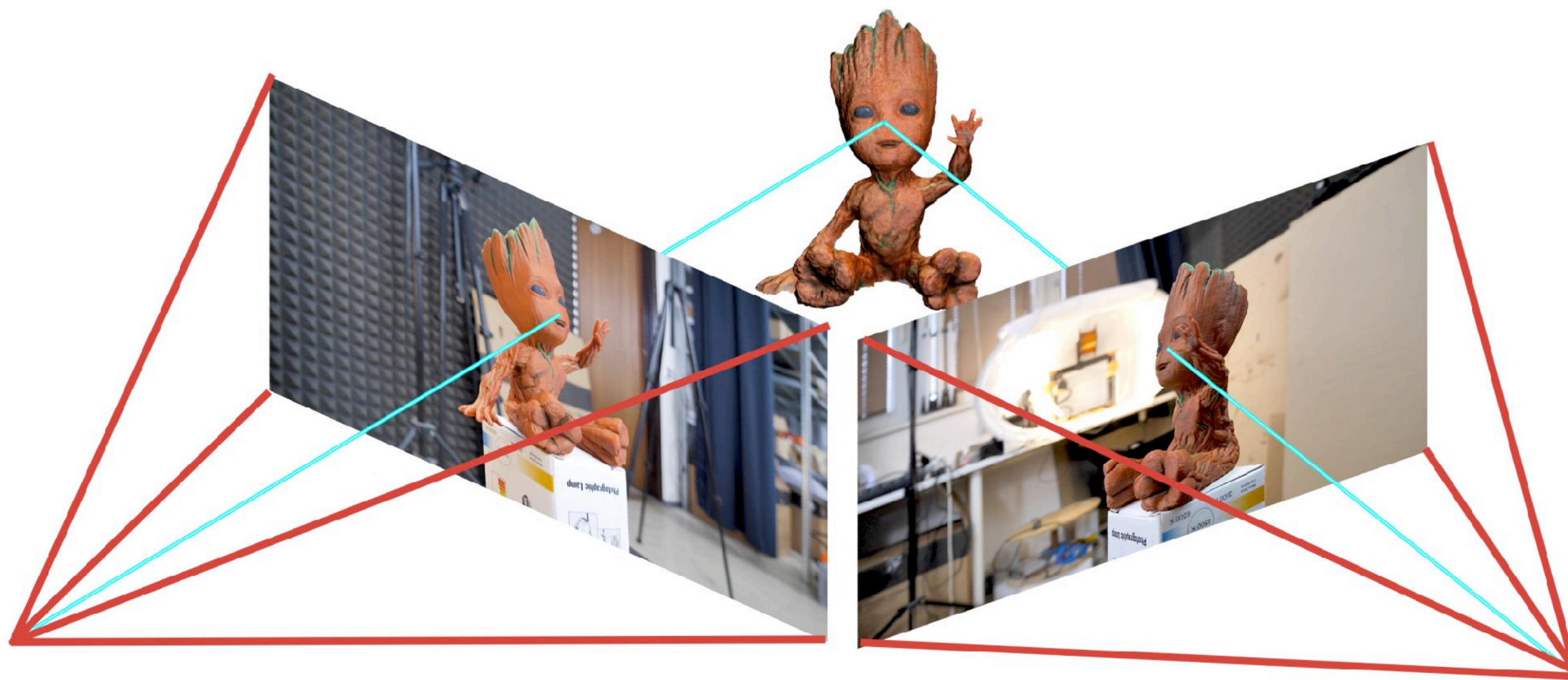
3D



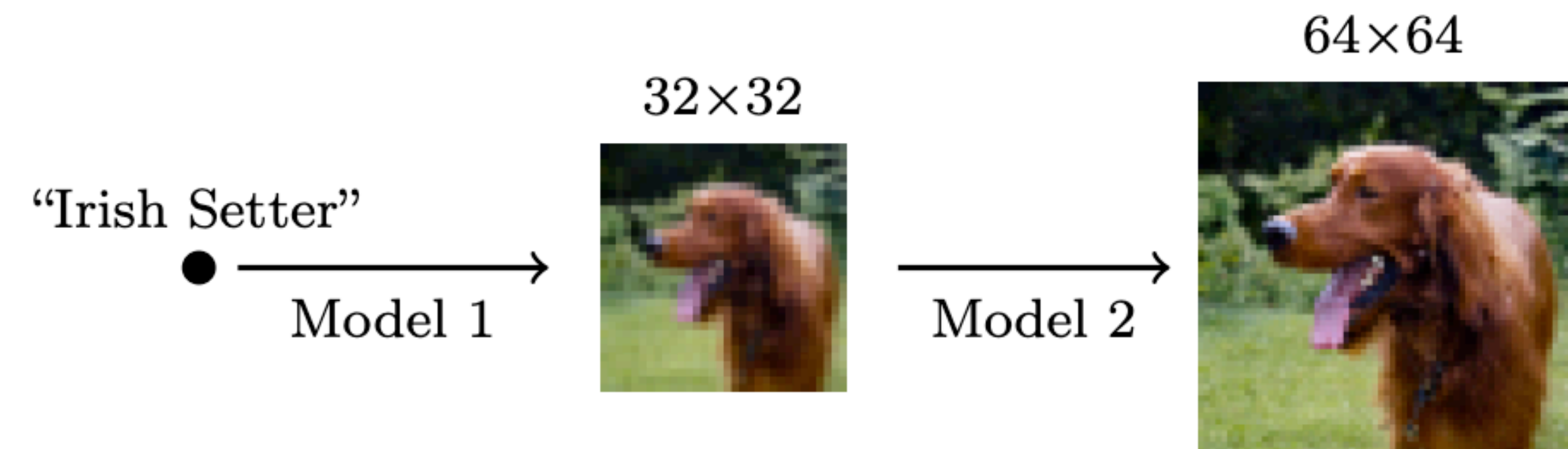
2D



# Challenges When Converting Between 2D and 3D



Multi-view consistency



Resolution

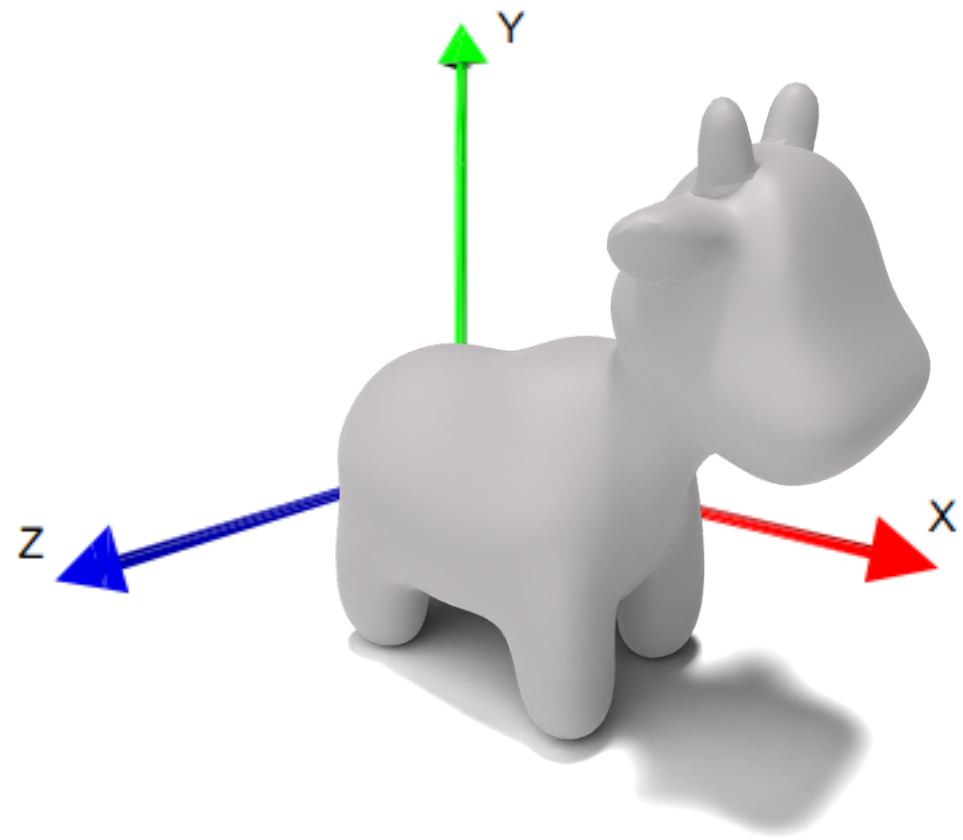


# Bridging the Gap Between 2D and 3D

A quick look at the projection and rasterization pipeline

# Bridging the Gap Between 2D and 3D

A quick look at the projection and rasterization pipeline

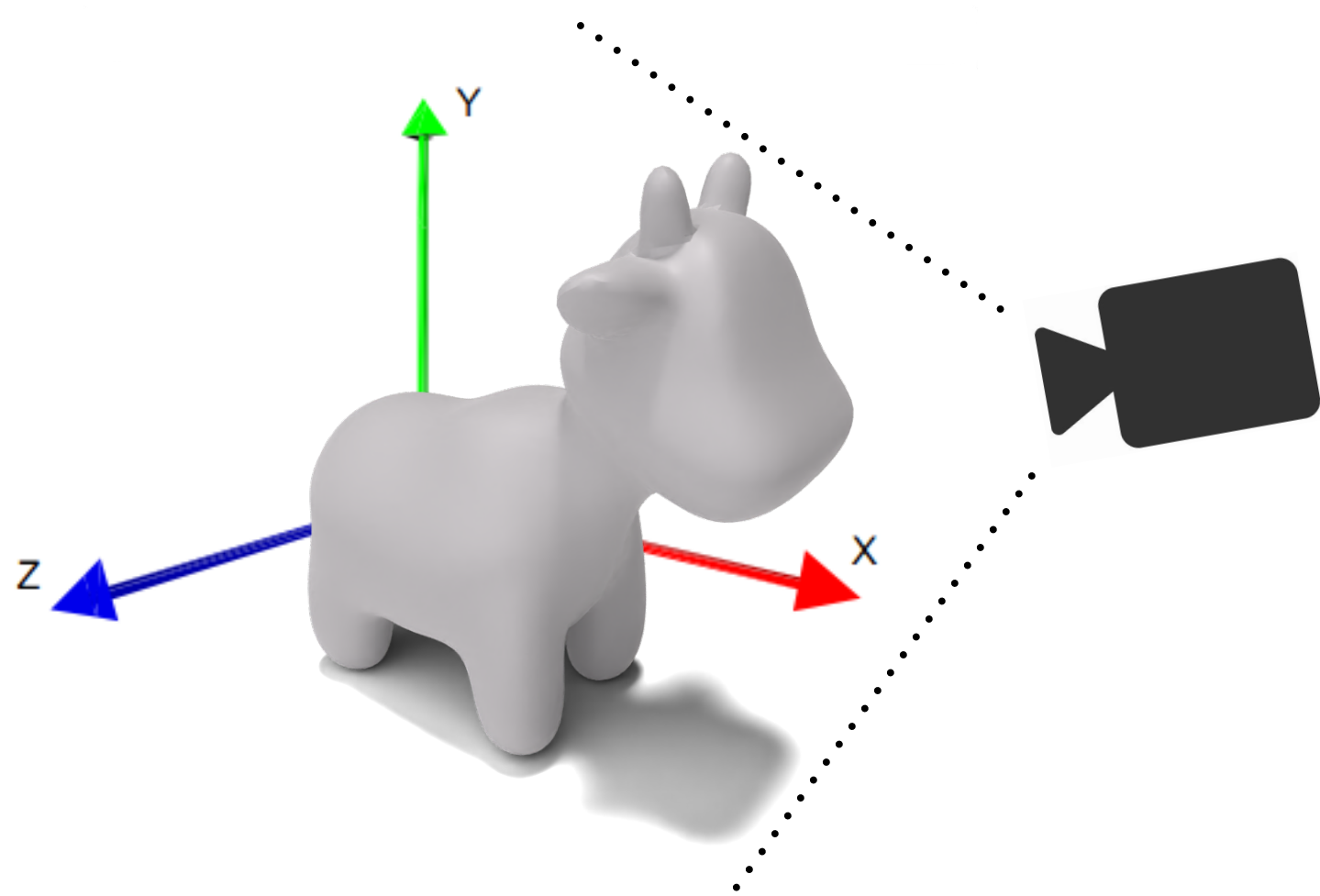


3D World  
Coordinates



# Bridging the Gap Between 2D and 3D

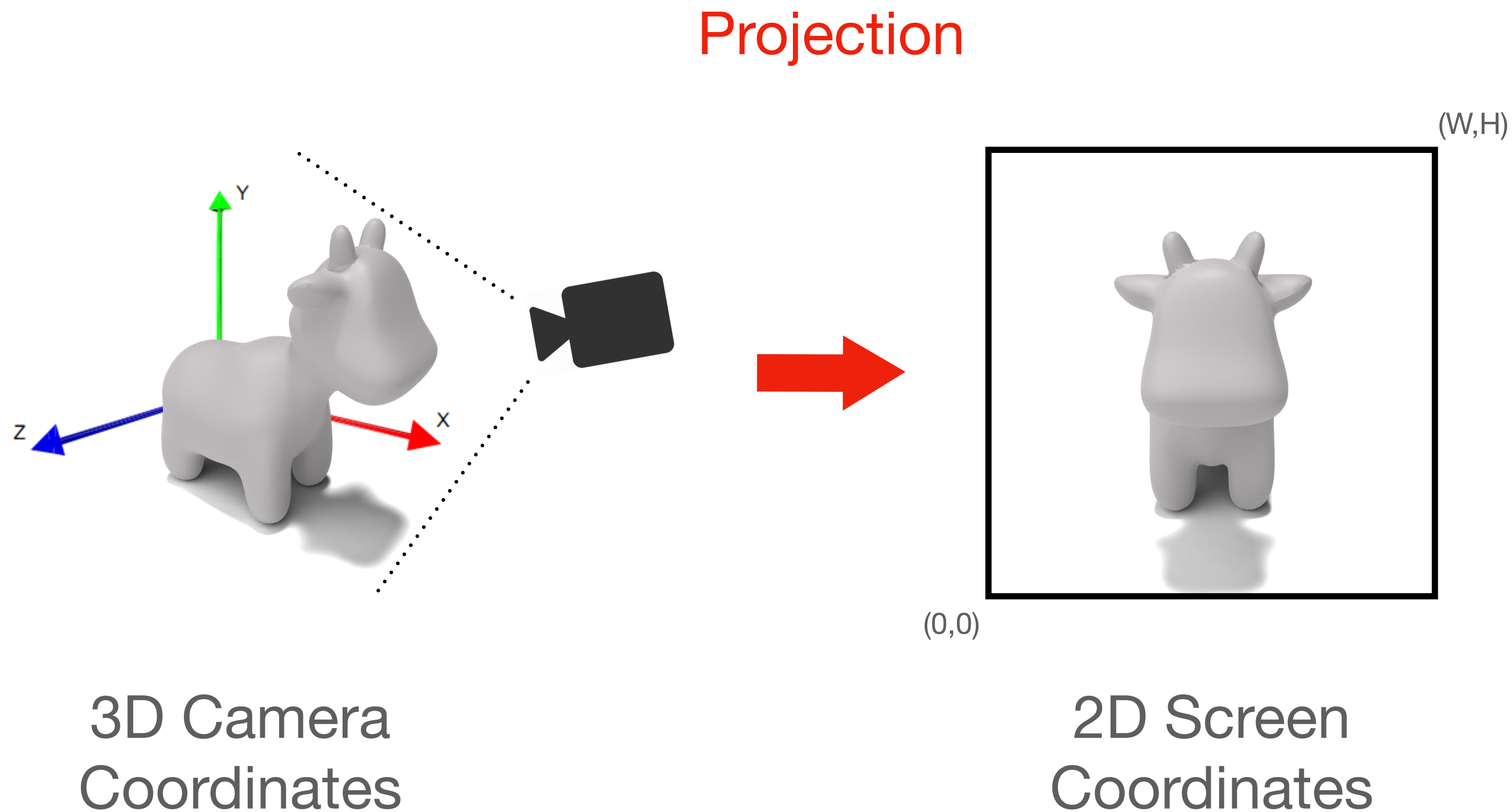
A quick look at the projection and rasterization pipeline



3D Camera  
Coordinates

# Bridging the Gap Between 2D and 3D

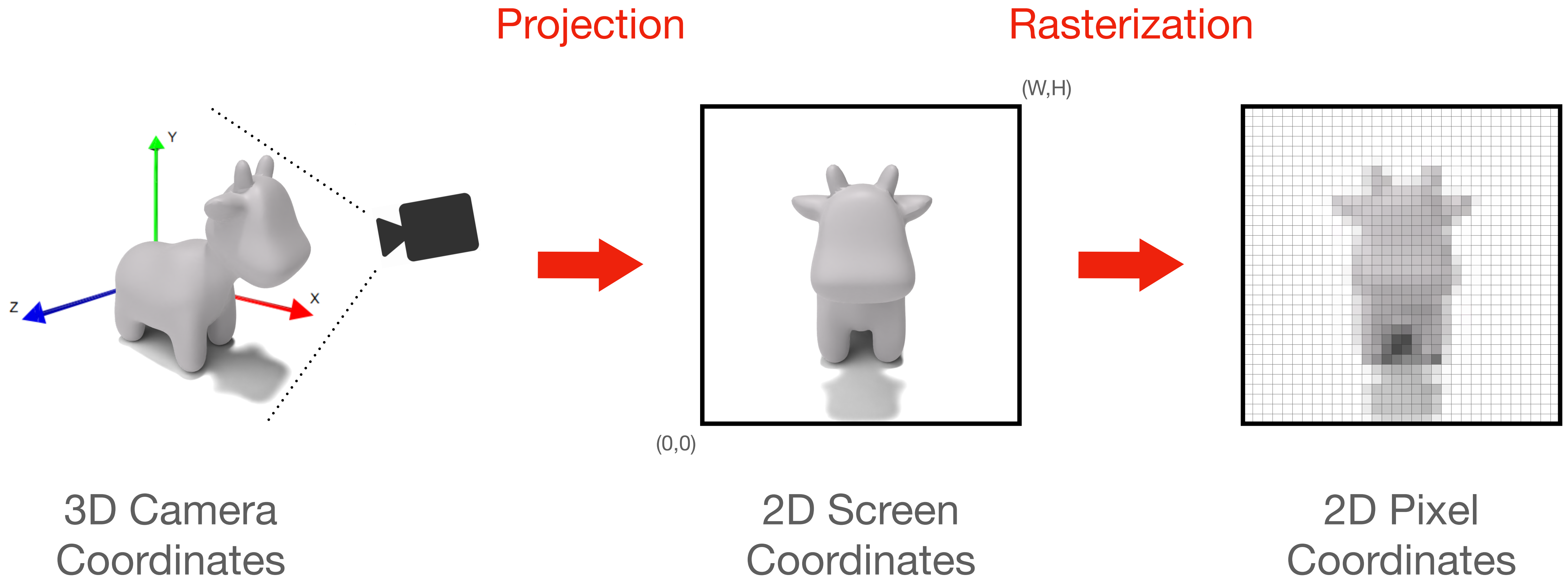
A quick look at the projection and rasterization pipeline





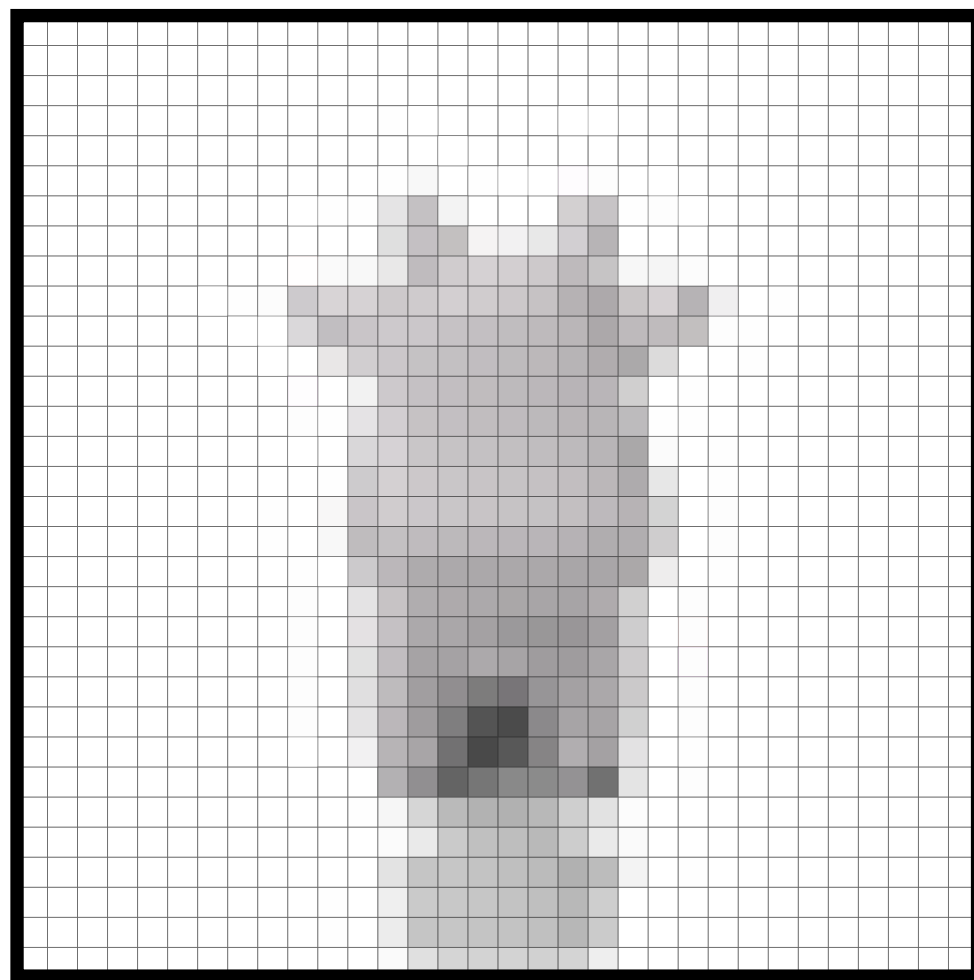
# Bridging the Gap Between 2D and 3D

A quick look at the projection and rasterization pipeline



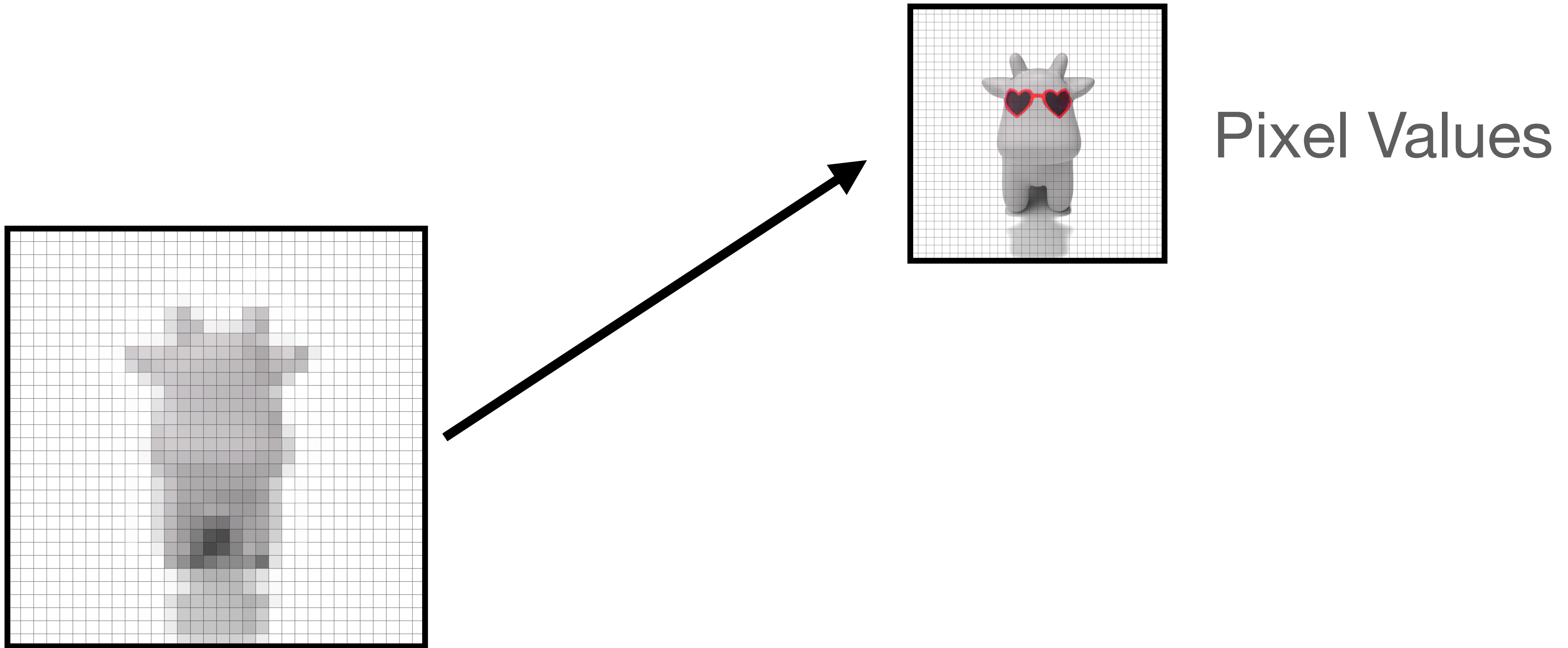
# Bridging the Gap Between 2D and 3D

Once mapped, we can use any 2D signal!



# Bridging the Gap Between 2D and 3D

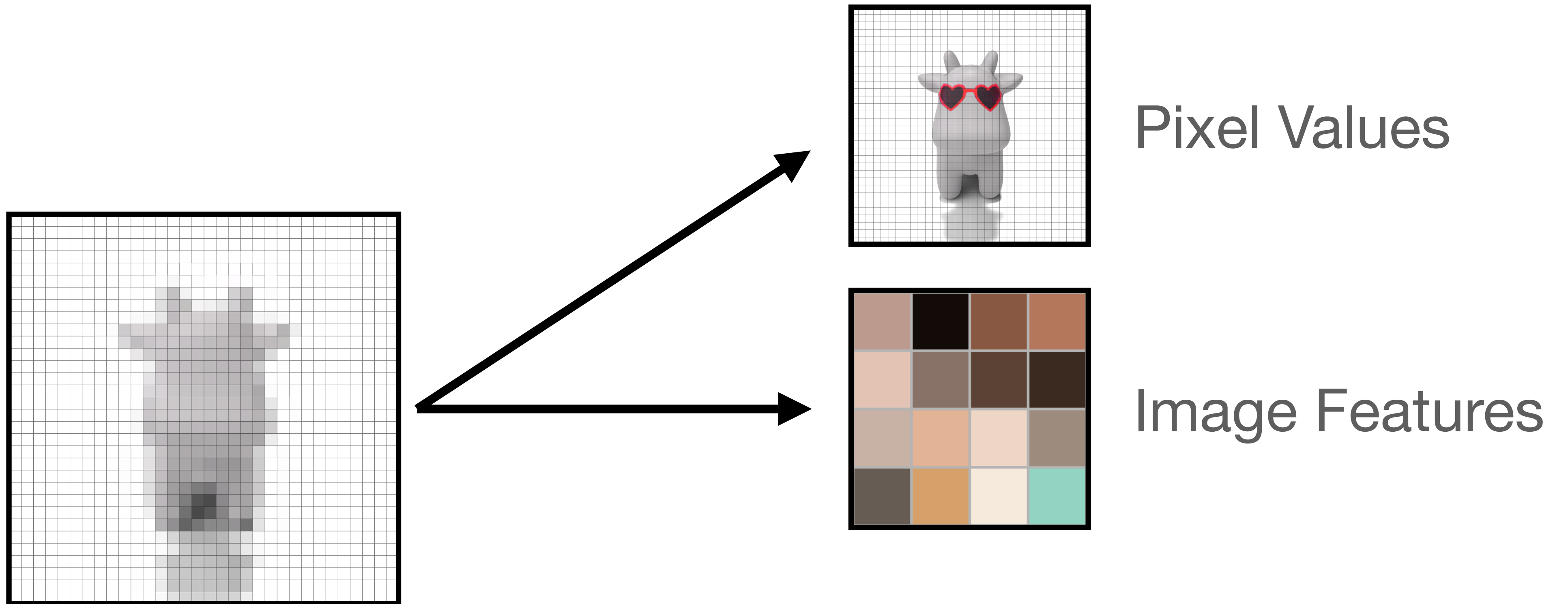
Once mapped, we can use any 2D signal!





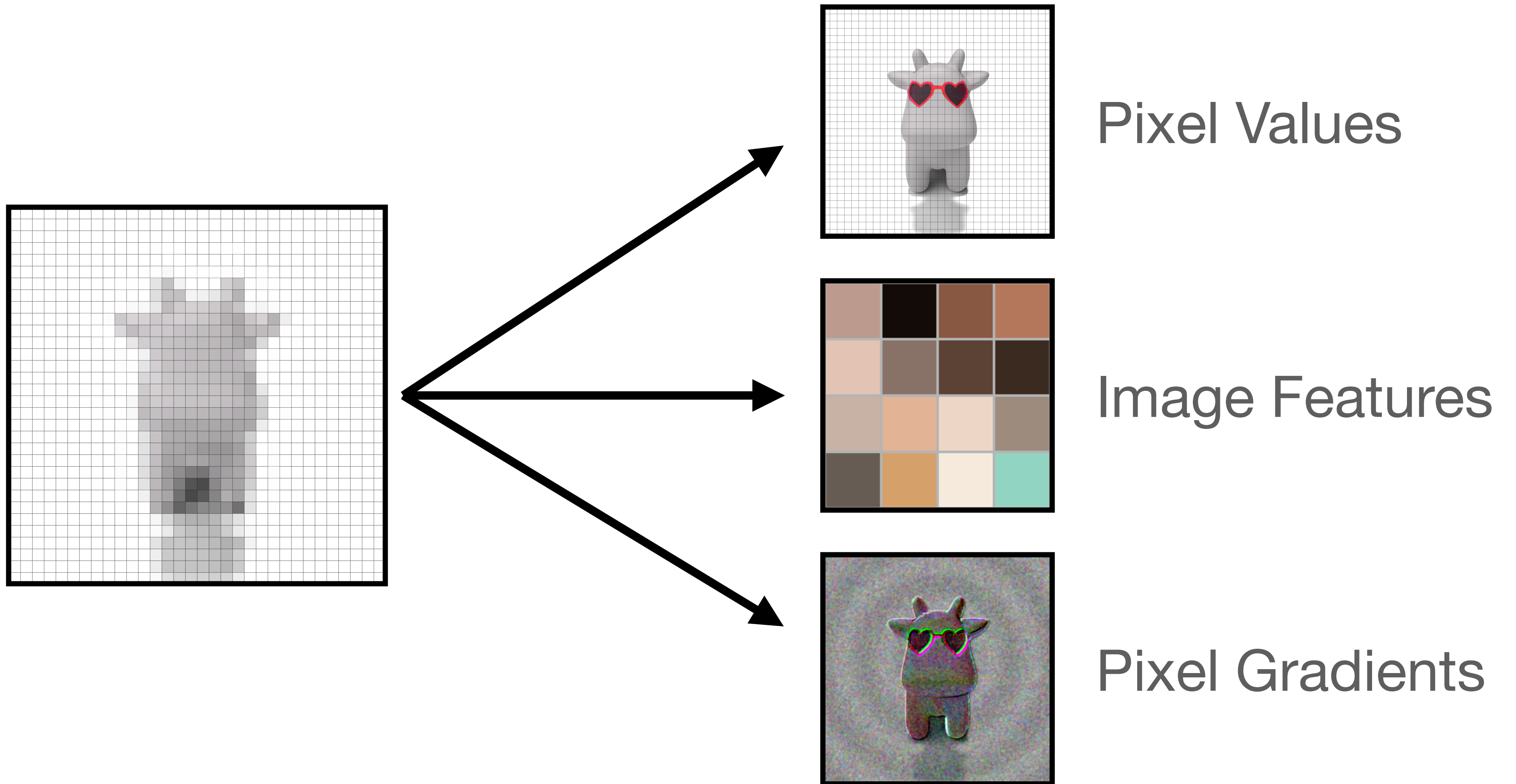
# Bridging the Gap Between 2D and 3D

Once mapped, we can use any 2D signal!



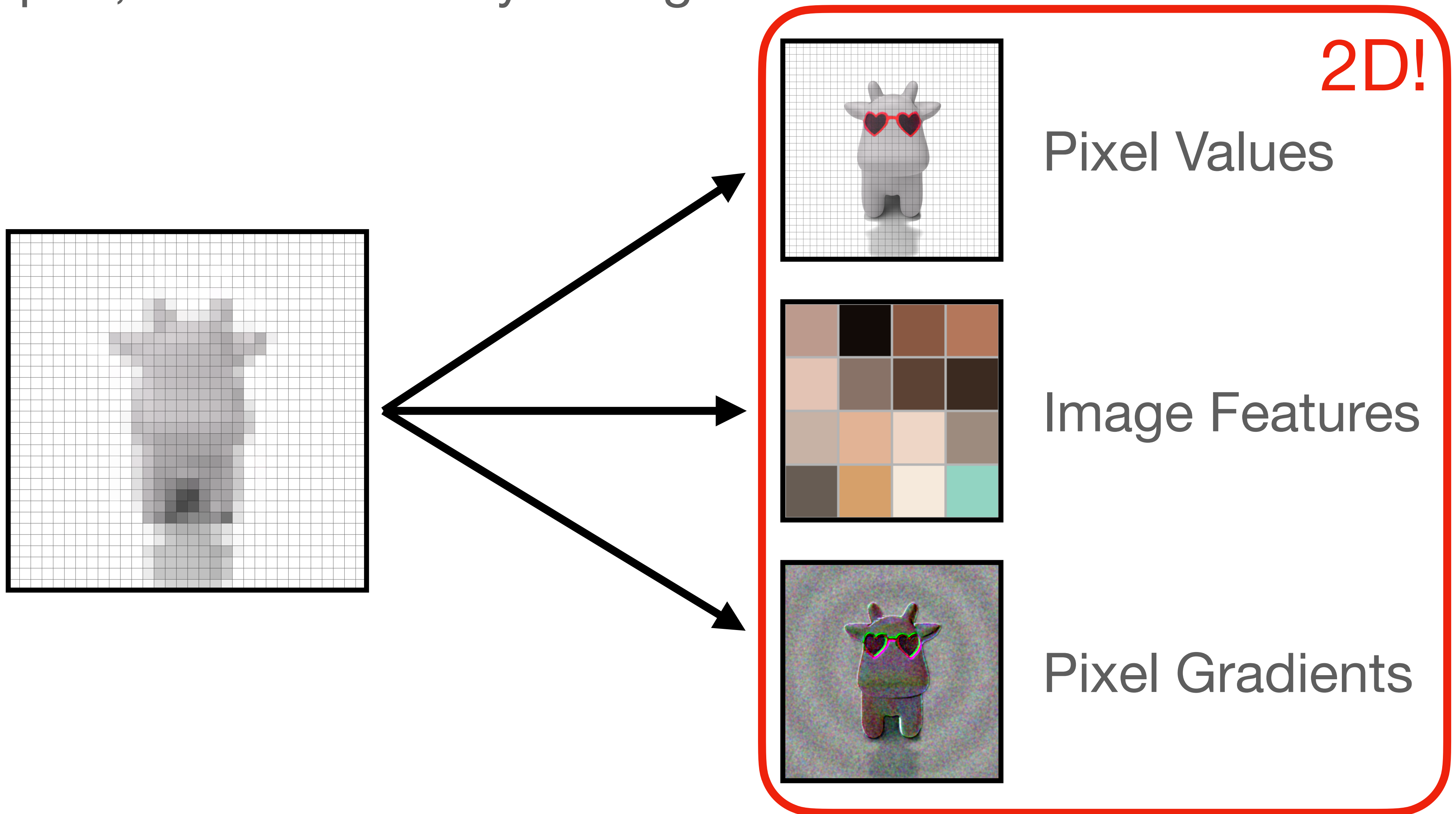
# Bridging the Gap Between 2D and 3D

Once mapped, we can use any 2D signal!



# Bridging the Gap Between 2D and 3D

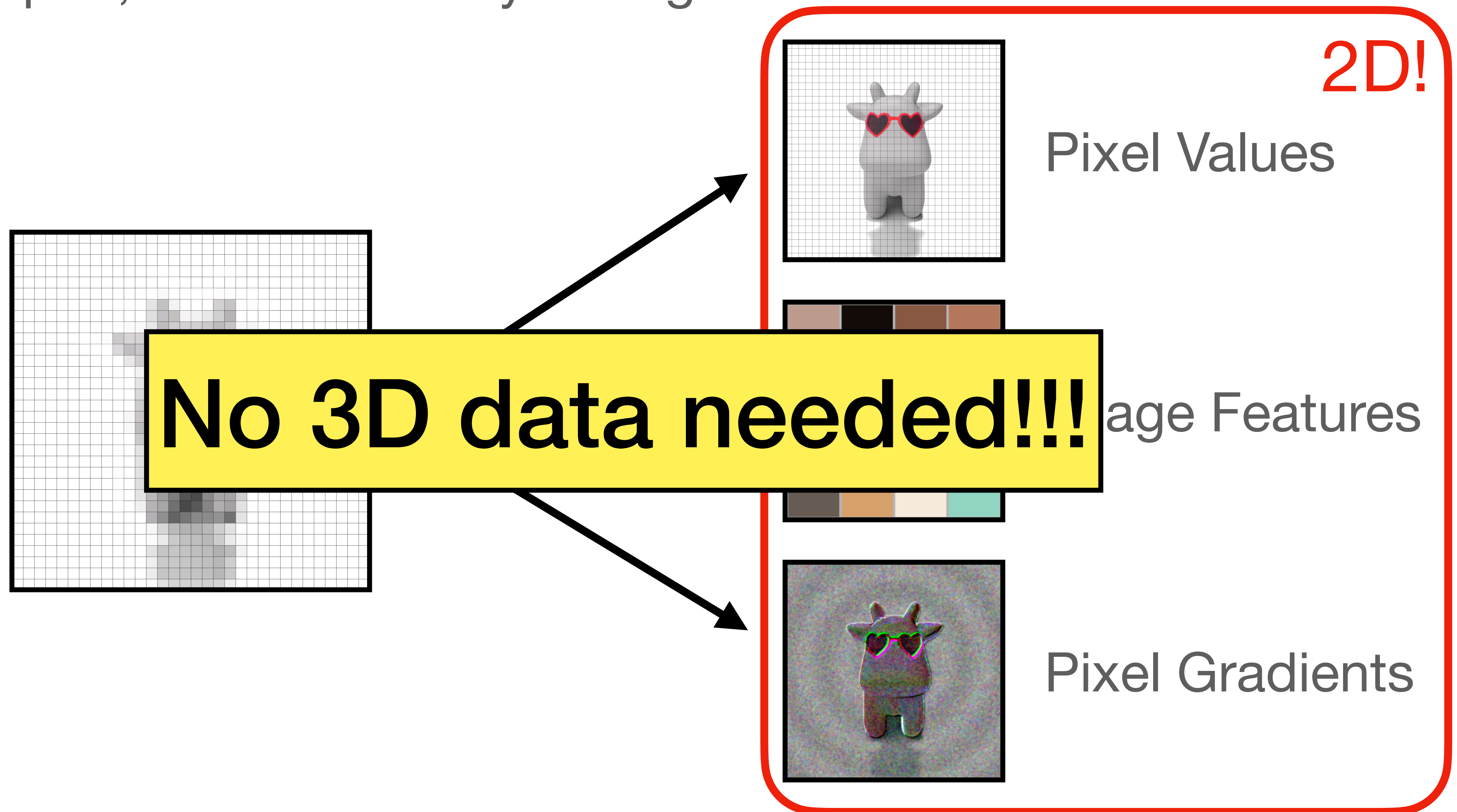
Once mapped, we can use any 2D signal!





# Bridging the Gap Between 2D and 3D

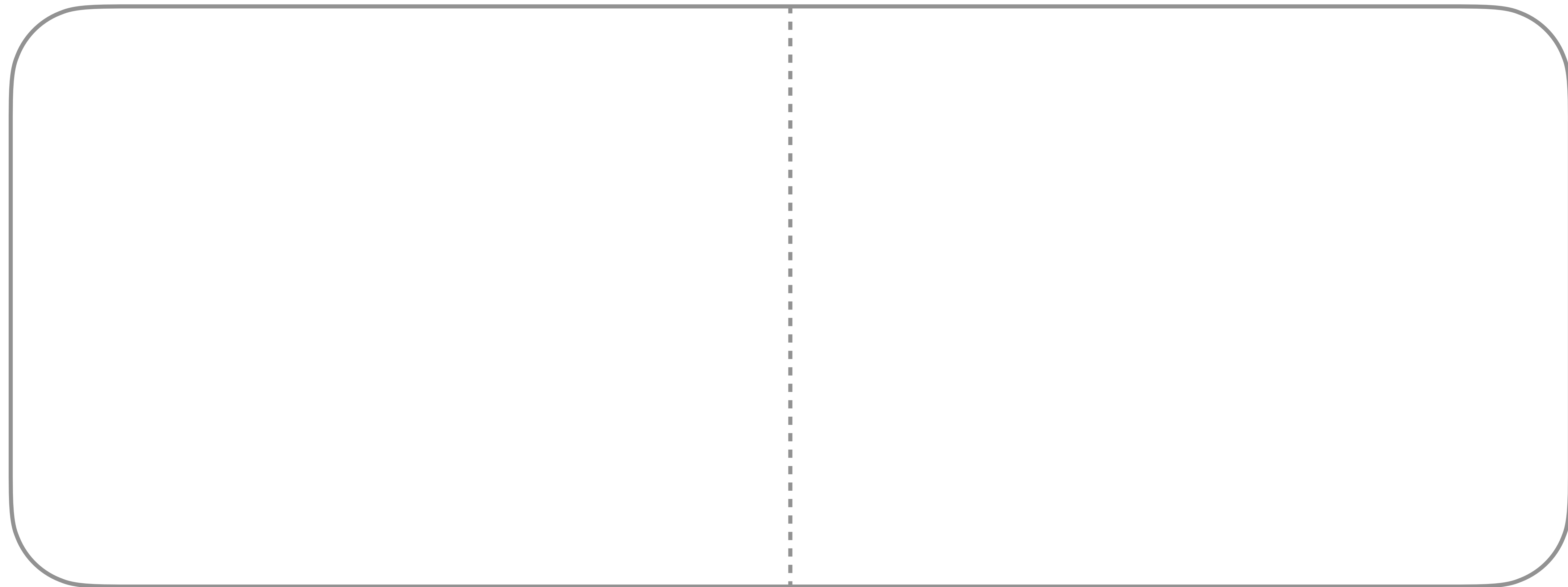
Once mapped, we can use any 2D signal!



# 3D From 2D Image Priors

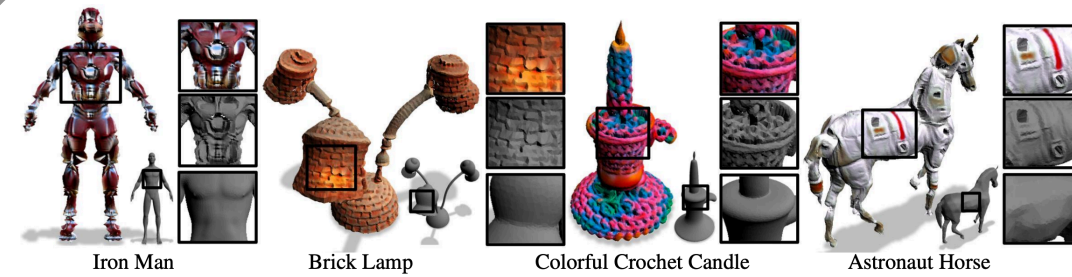
Optimization

Backprojection



# 3D From 2D Image Priors

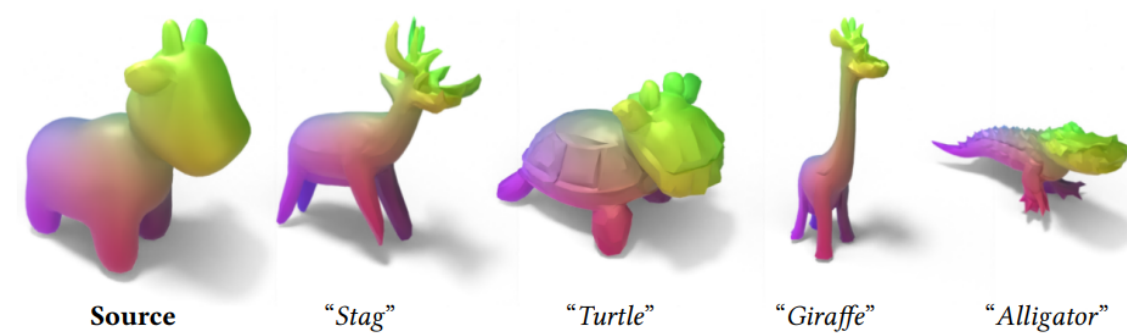
## Optimization



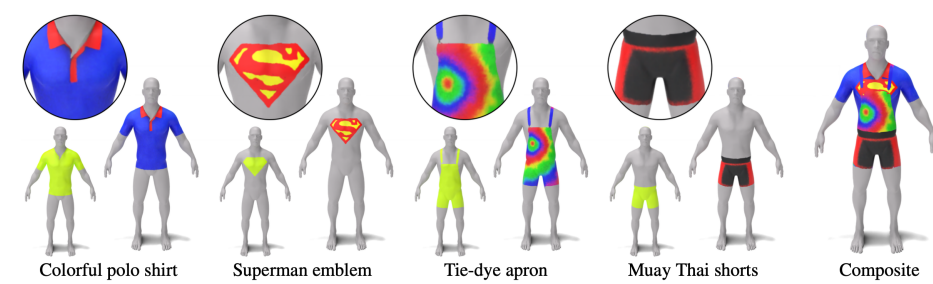
Text2Mesh, Mitchel et al., 2022



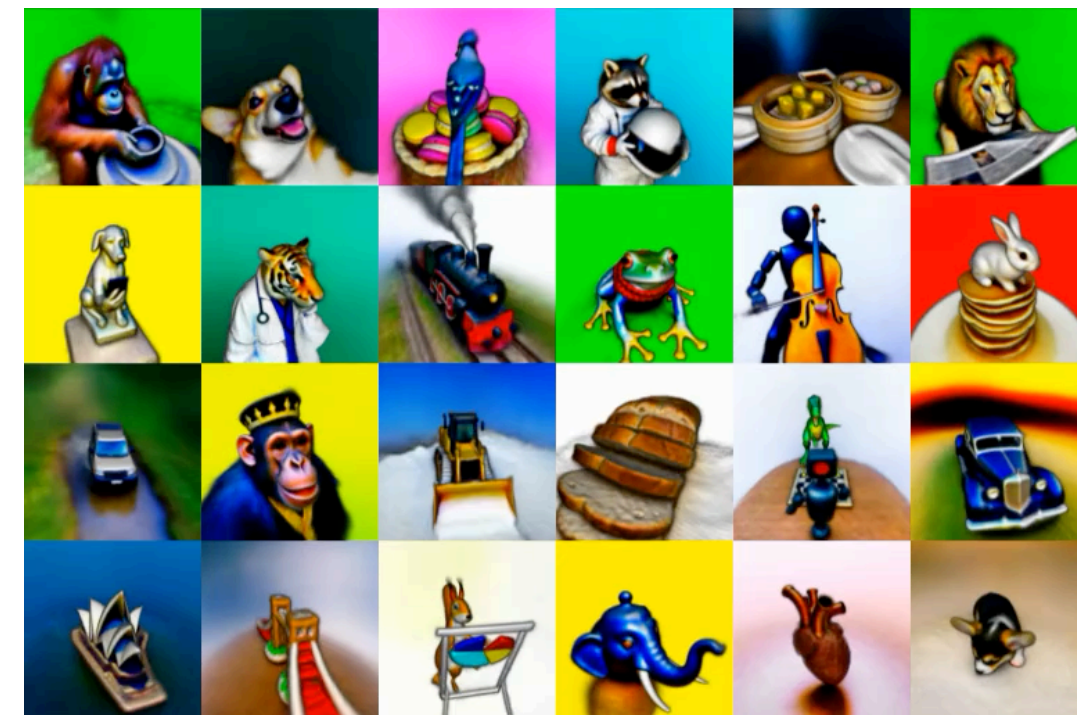
CLIP-Mesh, Khalid et al., 2022



TextDeformer, Gao et al., 2023



3D Paintbrush, Decatur et al., 2024



DreamFusion, Poole et al., 2023



Wir3d, Liu et al., 2025



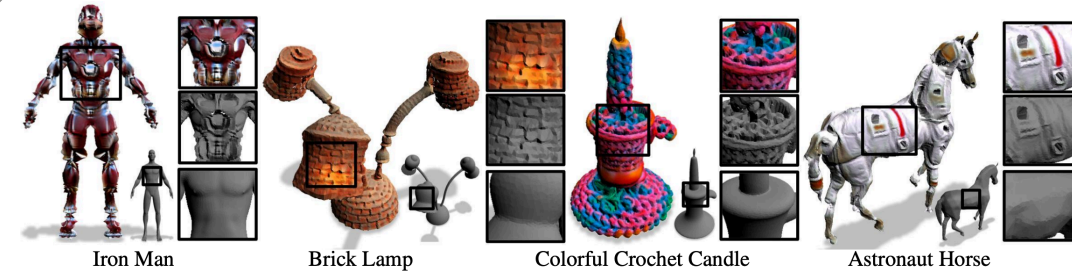
Geometry in Style, Dinh, et al., 2025

## Backprojection



# 3D From 2D Image Priors

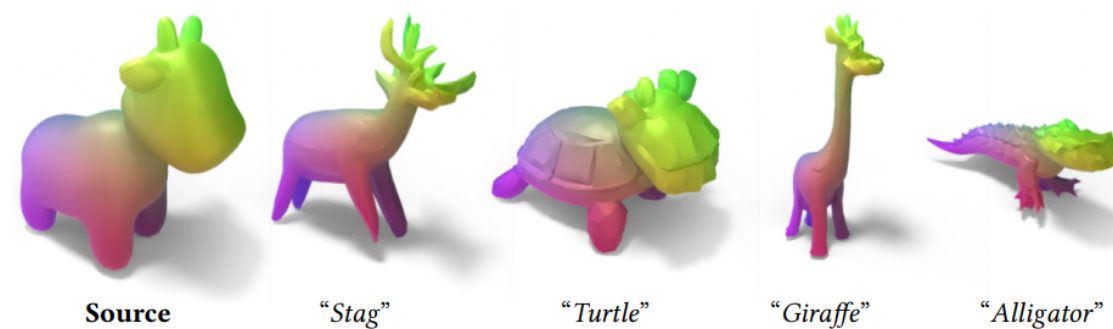
## Optimization



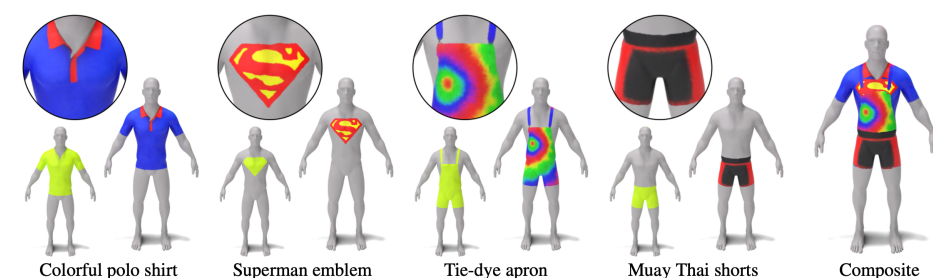
Text2Mesh, Mitchel et al., 2022



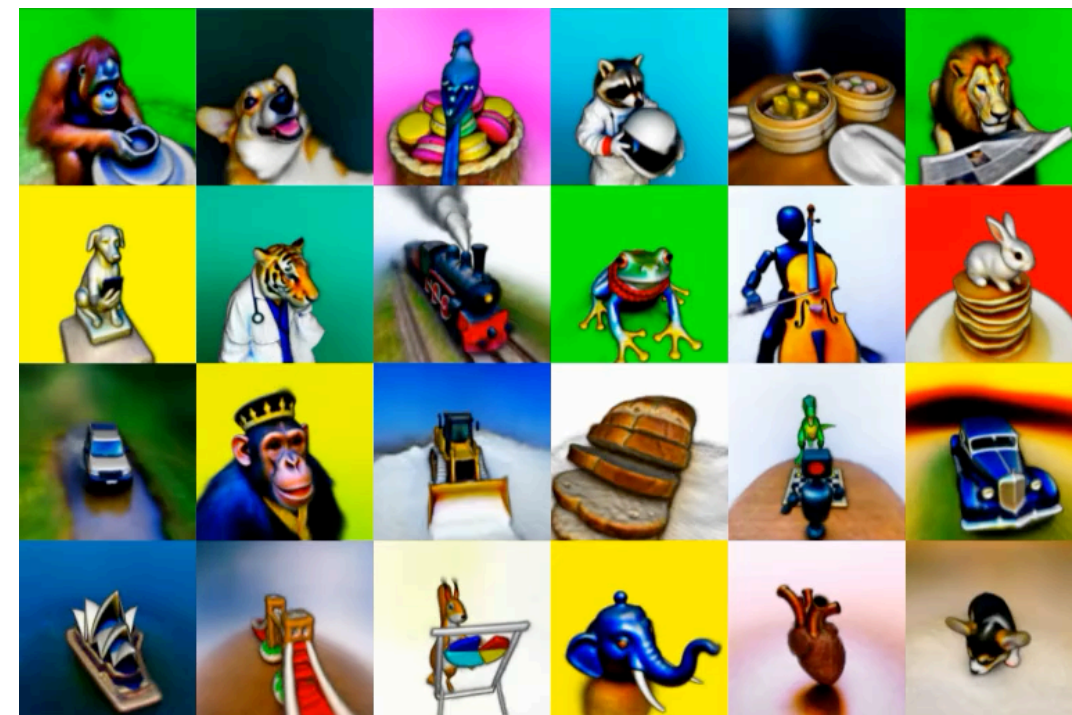
CLIP-Mesh, Khalid et al., 2022



TextDeformer, Gao et al., 2023



3D Paintbrush, Decatur et al., 2024



DreamFusion, Poole et al., 2023

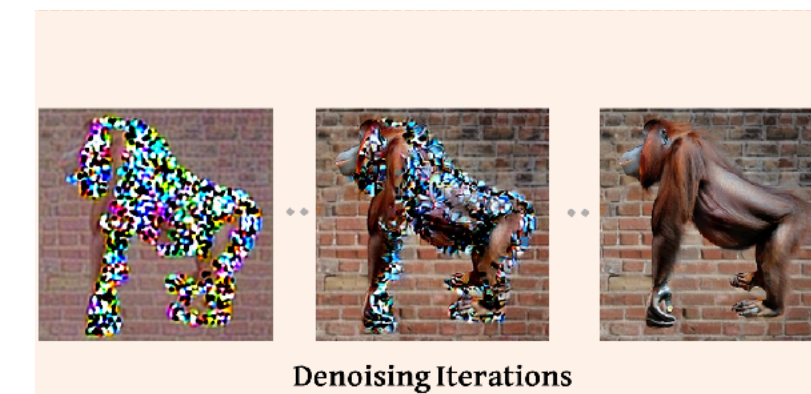


Wir3d, Liu et al., 2025

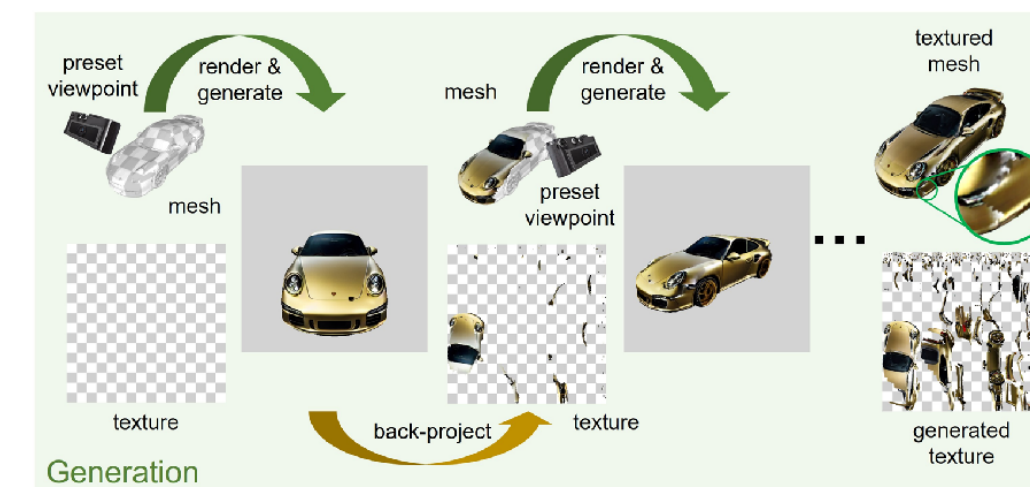


Geometry in Style, Dinh, et al., 2025

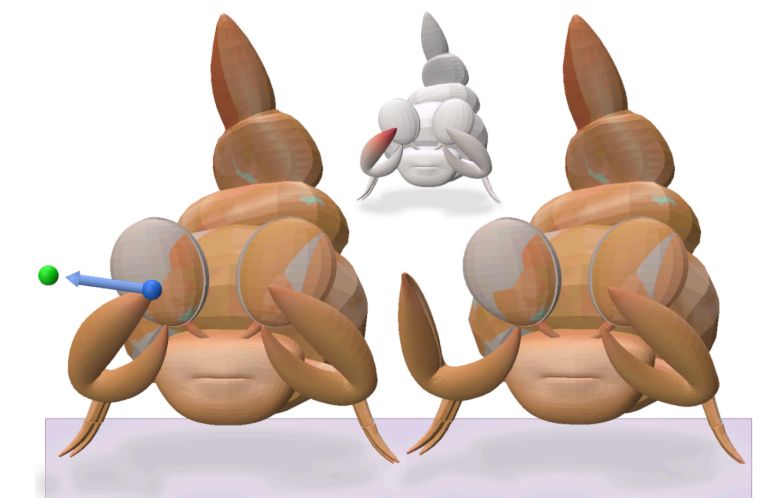
## Backprojection



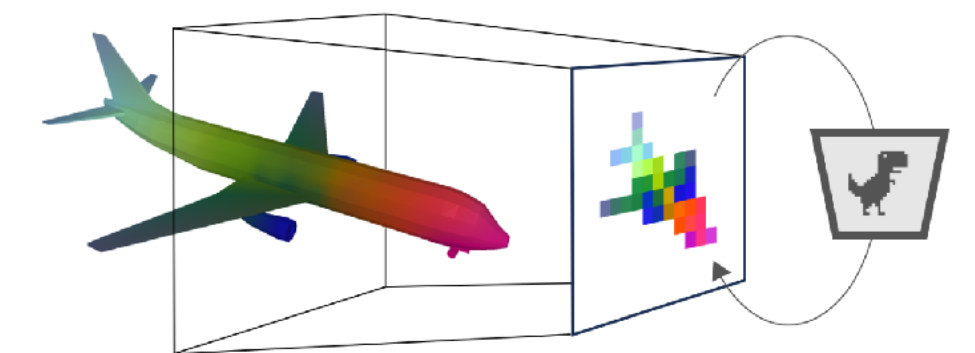
TEXTure, Richardson et al. 2023



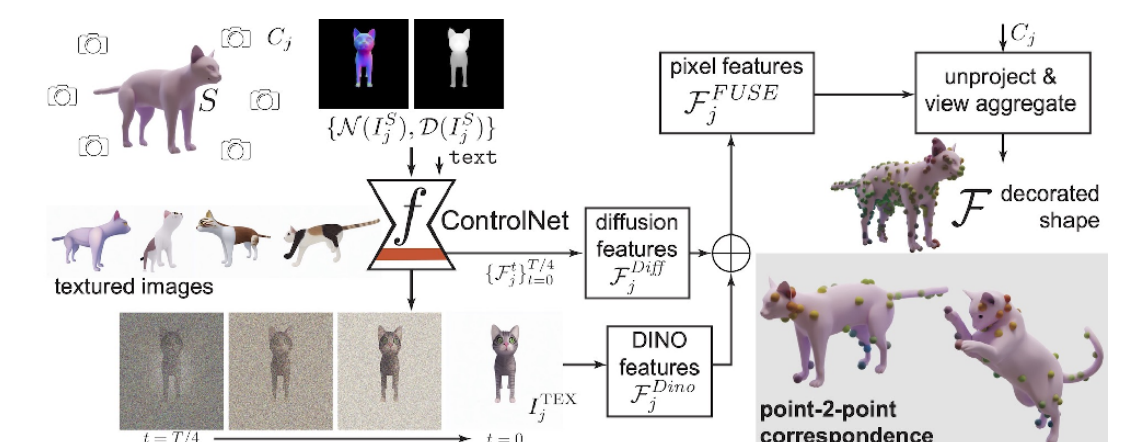
Text2Tex, Chen et al. 2023



DFD, Liu et al., 2026



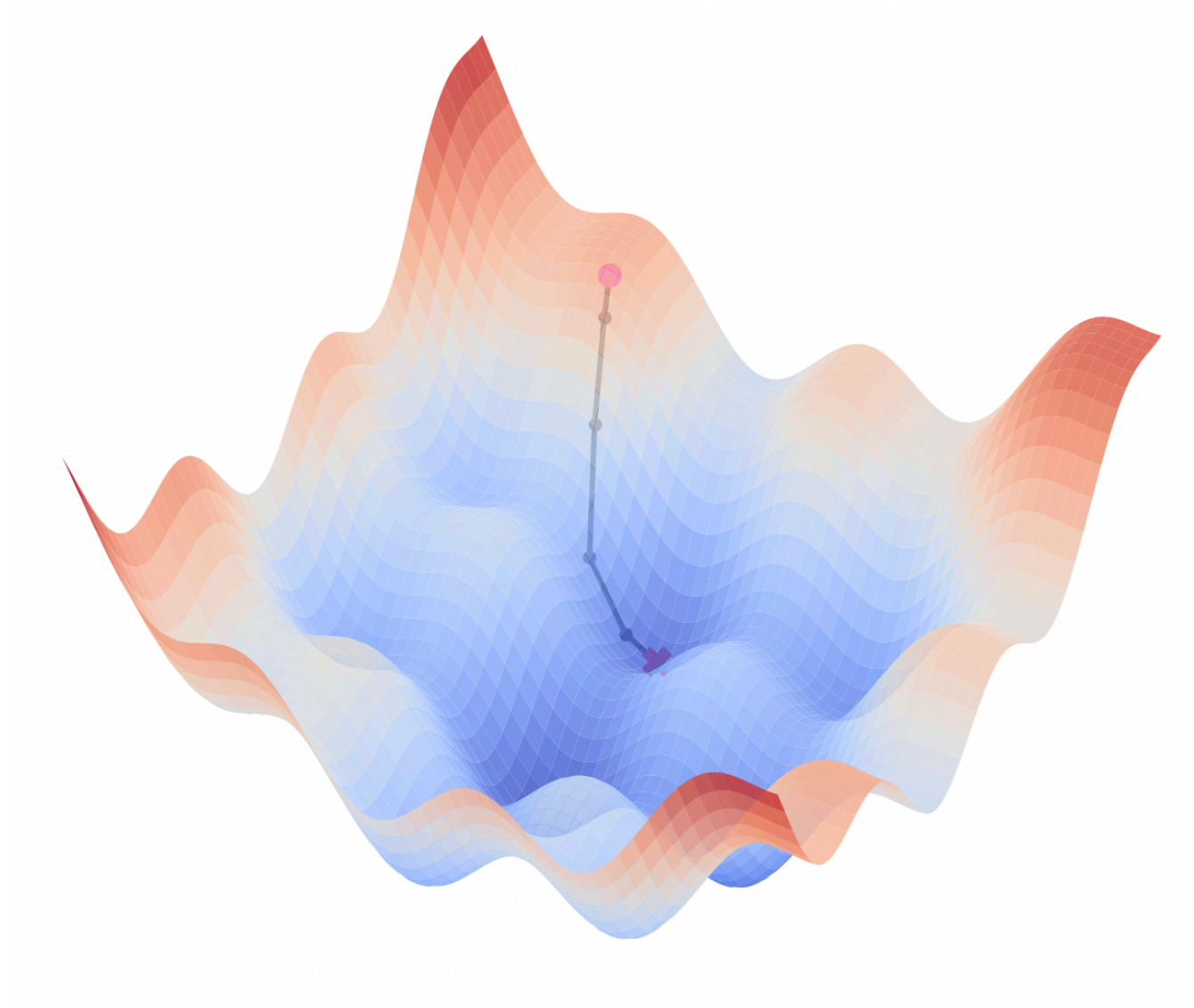
Back to 3D, Wimmer et al. 2023



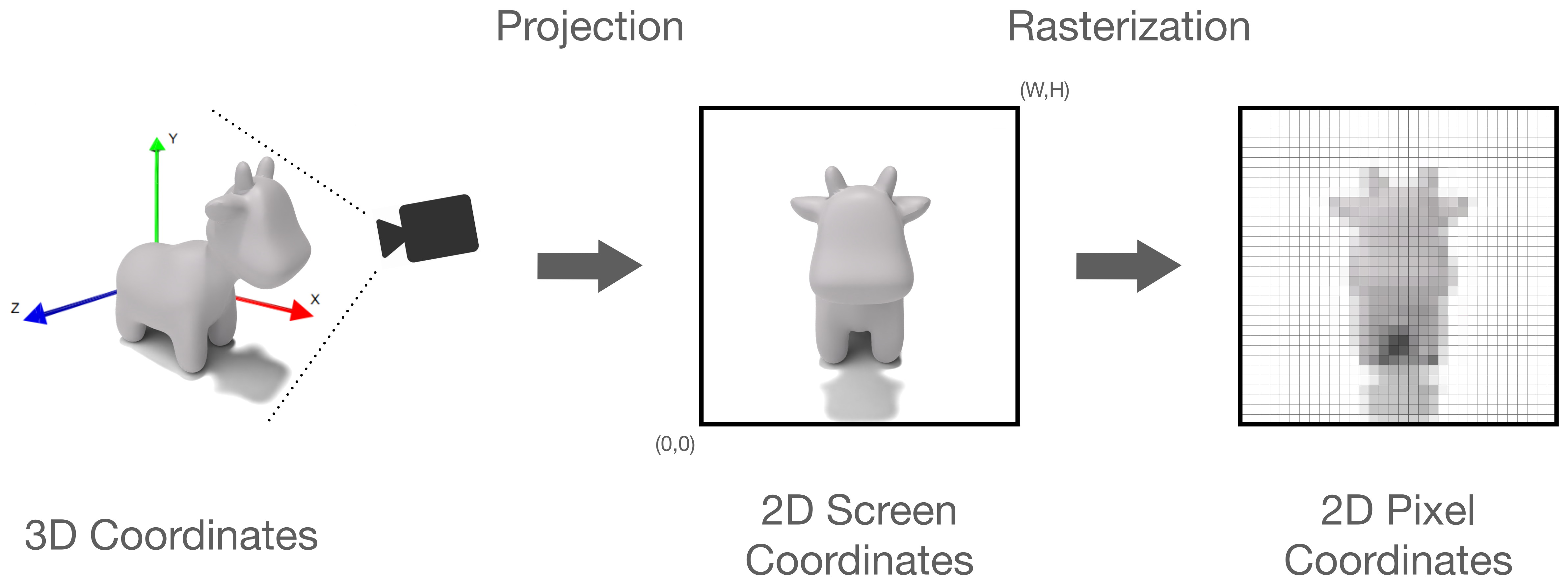
Diff3f, Dutt et al. 2024



# Optimization-Based Methods

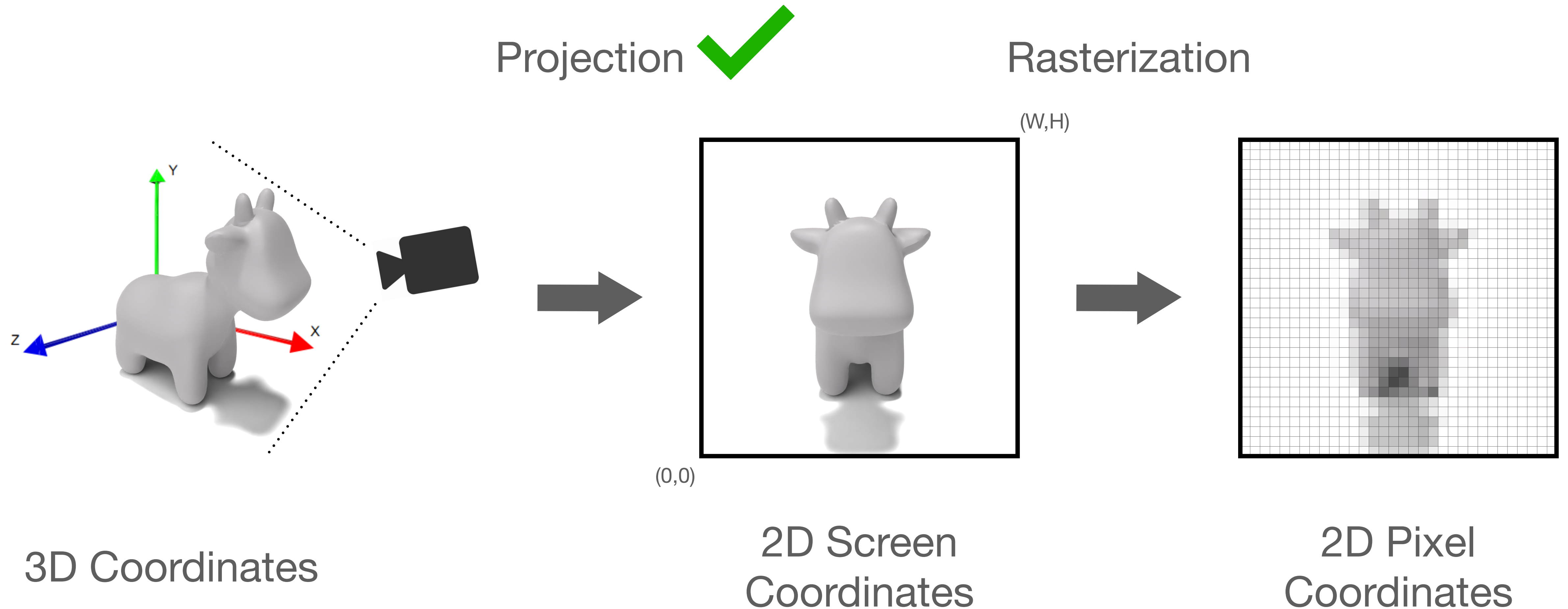


# Differentiable Mapping Between 3D and 2D

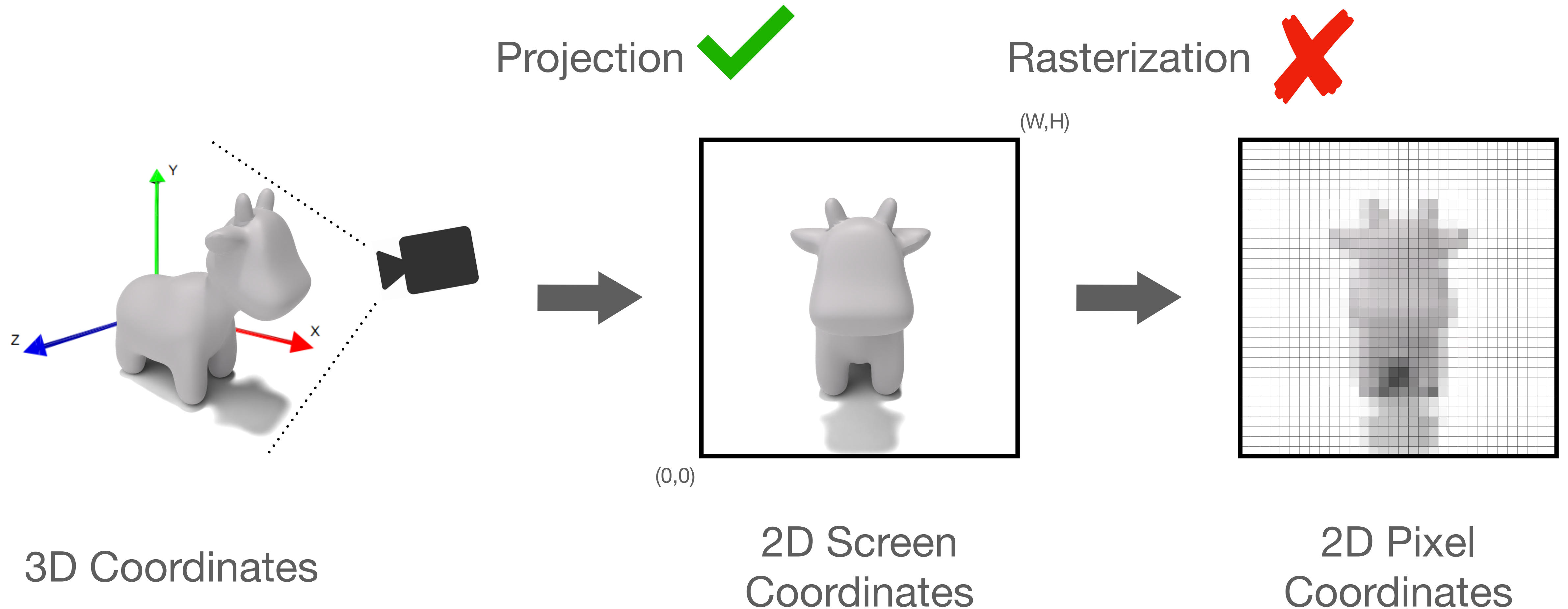




# Differentiable Mapping Between 3D and 2D

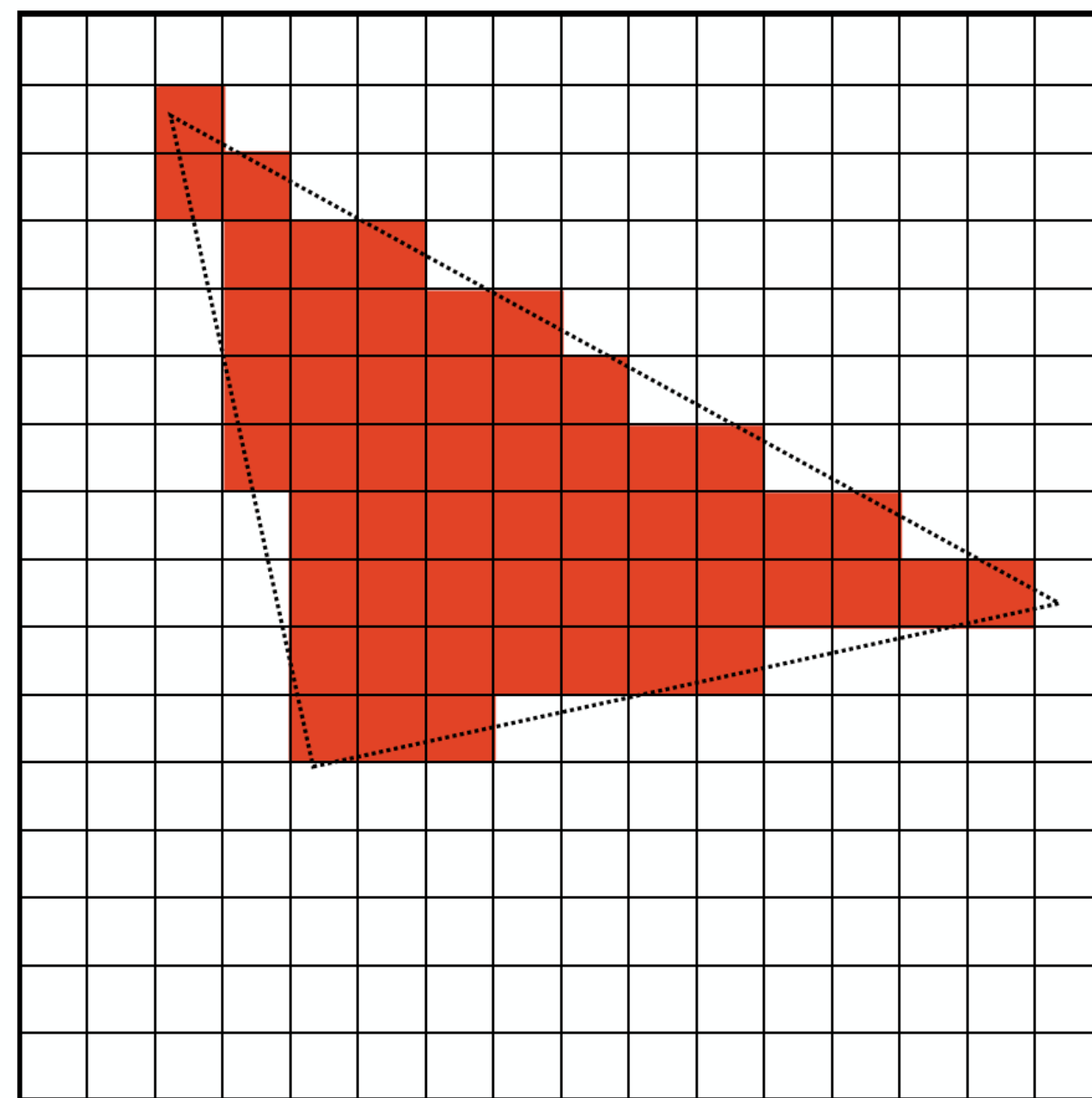
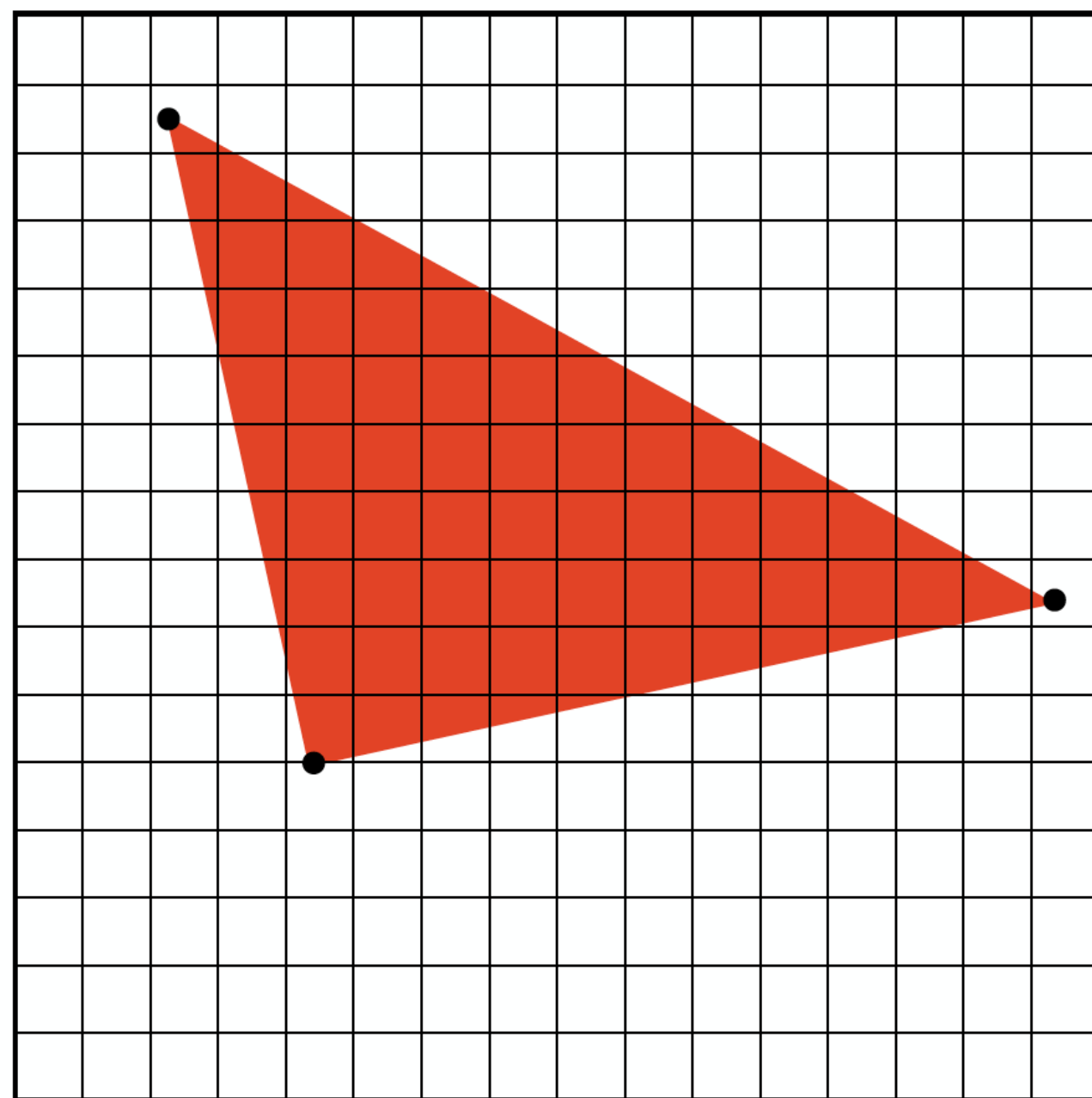


# Differentiable Mapping Between 3D and 2D



# Differentiable Rasterization & Rendering

“is this pixel covering the object”  $\rightarrow$  binary

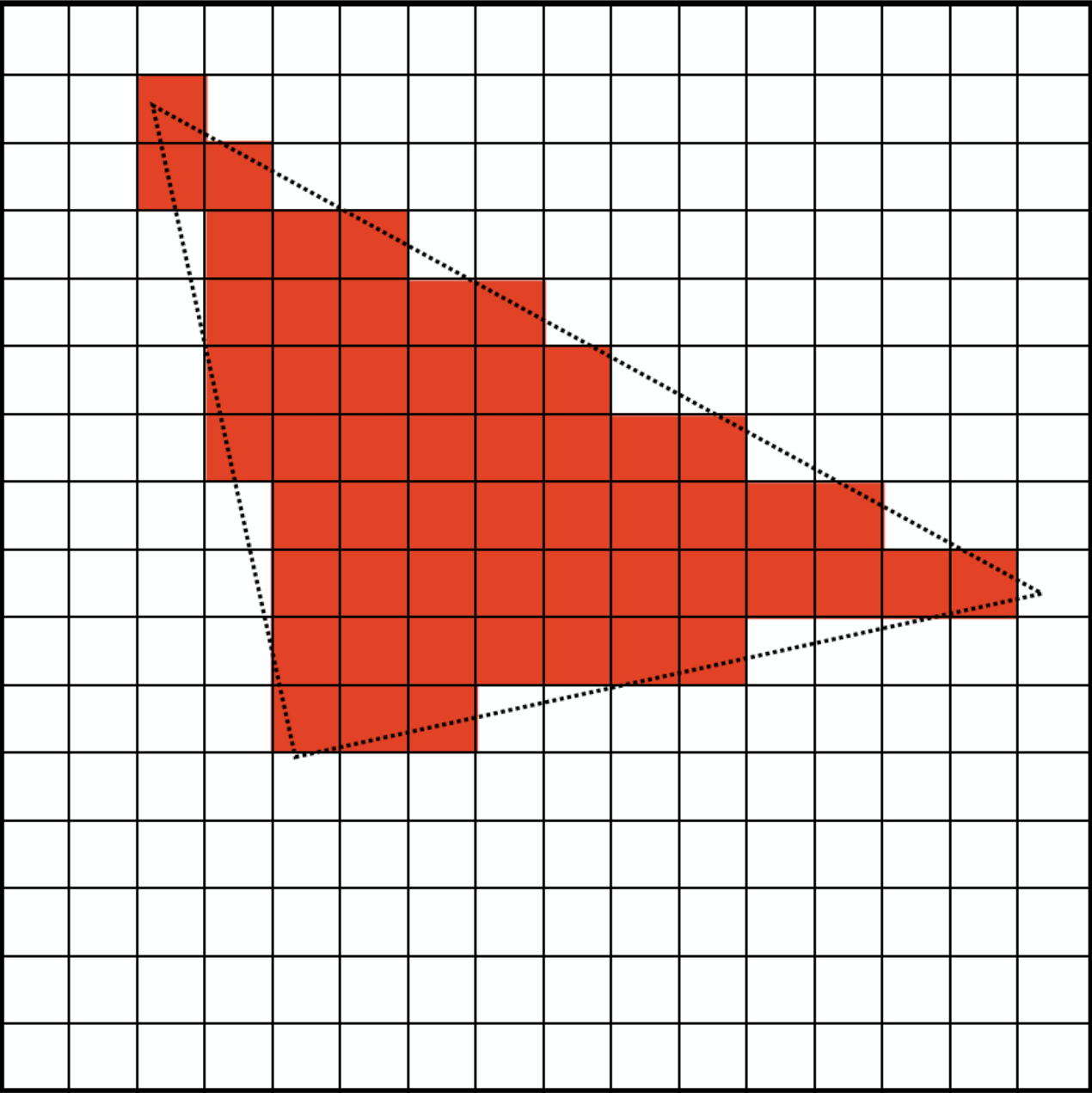
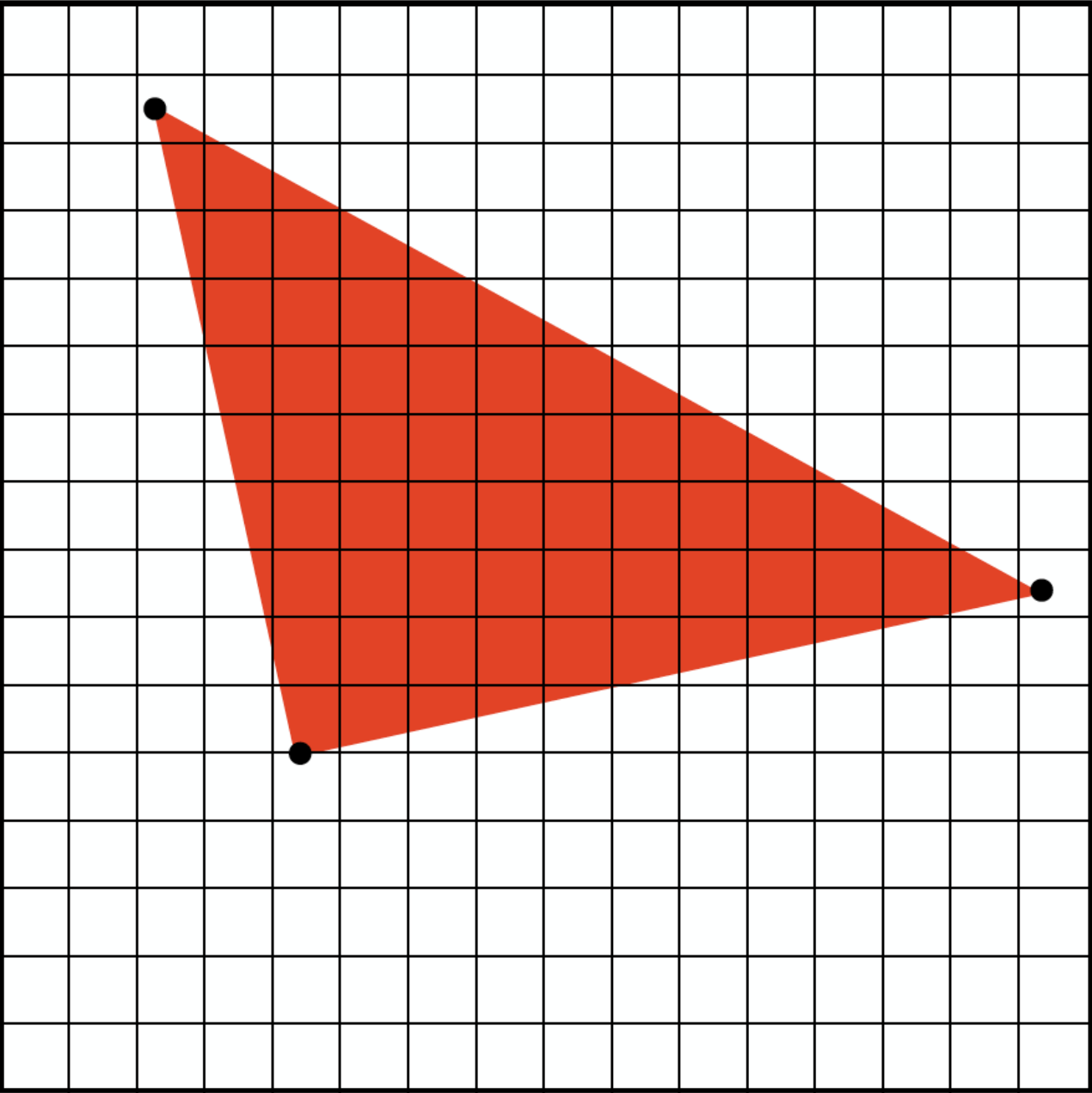




# Differentiable Rasterization & Rendering

Well studied problem!

“is this pixel covering the object” → binary



## Modular Primitives for High-Performance Differentiable Rendering

SAMULI LAINE, NVIDIA  
JANNE HELLSTEN, NVIDIA  
TERO KARRAS, NVIDIA  
YEONGHO SEOL, NVIDIA  
JAAKKO LEHTINEN, NVIDIA and Aalto University  
TIMO AILA, NVIDIA

We present a modular differentiable renderer design that yields performance superior to previous methods by leveraging existing, highly optimized hardware graphics pipelines. Our design supports all crucial operations in a modern graphics pipeline: rasterizing large numbers of triangles, attribute interpolation, filtered texture lookups, as well as user-programmable shading and geometry processing, all in high resolutions. Our modular primitives allow custom, high-performance graphics pipelines to be built directly within automatic differentiation frameworks such as PyTorch or TensorFlow. As a motivating application, we formulate facial performance capture as an inverse rendering problem and show that it can be solved efficiently using our tools. Our results indicate that this simple and straightforward approach achieves excellent geometric correspondence between rendered results and reference imagery.

CCS Concepts: • **Computing methodologies** → **Rasterization**; **Shape inference**.

Additional Key Words and Phrases: Differentiable rendering, rasterization, motion capture.

**ACM Reference Format:**  
Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular Primitives for High-Performance Differentiable Rendering. *ACM Trans. Graph.* 39, 6, Article 194 (December 2020), 14 pages. <https://doi.org/10.1145/3414685.3417861>

### 1 INTRODUCTION

Differentiable rendering is a fundamental building block in machine learning of 3D geometry. Typically training data is available only as images, and finding a corresponding 3D representation requires *analysis by synthesis*, i.e., rendering candidate images, computing the loss based on training and candidate images, and propagating the errors back to 3D positions and other scene attributes. Many classical computer graphics and vision problems including the estimation of reflectance, geometry, lighting, and camera parameters can be cast into this *inverse rendering* framework [Patow and Pueyo 2003].

Authors' addresses: Samuli Laine, NVIDIA, [slaine@nvidia.com](mailto:slaine@nvidia.com); Janne Hellsten, NVIDIA, [jhellsten@nvidia.com](mailto:jhellsten@nvidia.com); Tero Karras, NVIDIA, [tkarras@nvidia.com](mailto:tkarras@nvidia.com); Yeongho Seol, NVIDIA, [yseol@nvidia.com](mailto:yseol@nvidia.com); Jaakko Lehtinen, NVIDIA, Aalto University, [jlehtinen@nvidia.com](mailto:jlehtinen@nvidia.com); Timo Aila, NVIDIA, [aila@nvidia.com](mailto:aila@nvidia.com).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).  
© 2020 Association for Computing Machinery.  
0730-0301/2020/12-ART194 \$15.00  
<https://doi.org/10.1145/3414685.3417861>

Much of modern machine learning makes use of first order (gradient-based) optimization techniques implemented using backpropagation. From a computational point of view, the explosive growth in model sizes and capabilities has, for a large part, relied on the availability of primitive operations that allow massively parallel and coherent execution of both the forward and backward (gradient) passes, allowing highly complex computation graphs to be built out of them. However, the majority of effort in creating efficient primitive operations has focused on operating on densely-sampled data stored in multidimensional regular grids. These kind of operations alone are not sufficient for 3D rendering because the mapping from a scene representation to pixel values is highly irregular and dynamic. As such, specialized differentiable rendering algorithms have emerged for solving two problems:

- (1) *Forward pass*: Given the factors that affect the shape and appearance of a 3D scene, render a 2D image; and
- (2) *Backward pass*: Given the gradient of a loss function defined on the output image pixels, compute the gradient of the loss with respect to the input shape and appearance factors.

This interface is universally used by autodifferentiation frameworks such as PyTorch and TensorFlow, and it allows 3D rendering to be used as a building block in a complex model that is trained by modern stochastic first order optimization techniques.

Despite its long history, differentiable rendering can be considered a nascent field due to the recent proliferation of algorithms and applications. Most previous research is targeted towards a specific use case (e.g., pose or shape estimation), and is typically only evaluated on downstream tasks as part of a larger machine learning system [Chen et al. 2019; Kato et al. 2018; Liu et al. 2019]. These specific use cases and data sets allow optimizations and design choices that do not scale to other uses. For example, low geometric complexity may make it acceptable to not parallelize over triangles, but this quickly backfires on a larger scene: single objects viewed in a vacuum may enable one to disregard the effects of mutual occlusion between triangles when computing gradients, but this approximation is untenable in a scene with non-trivial depth complexity.

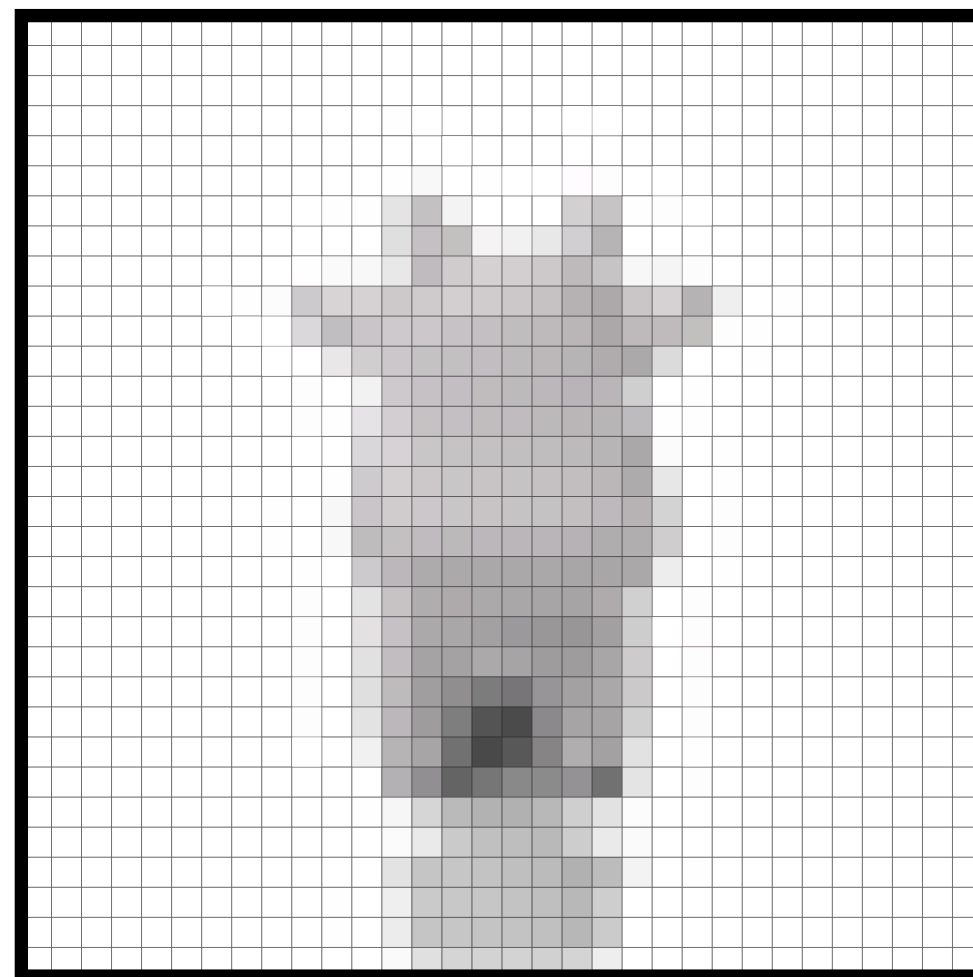
Another parallel line of research studies differentiable physically-based light transport simulation that models complex effects such as area light sources and indirect illumination [Li et al. 2018; Loubet et al. 2019; Nimier-David et al. 2019]. Built on Monte Carlo sampling, these methods seek the best attainable image quality and accuracy at the cost of longer rendering times.

ACM Trans. Graph., Vol. 39, No. 6, Article 194. Publication date: December 2020.

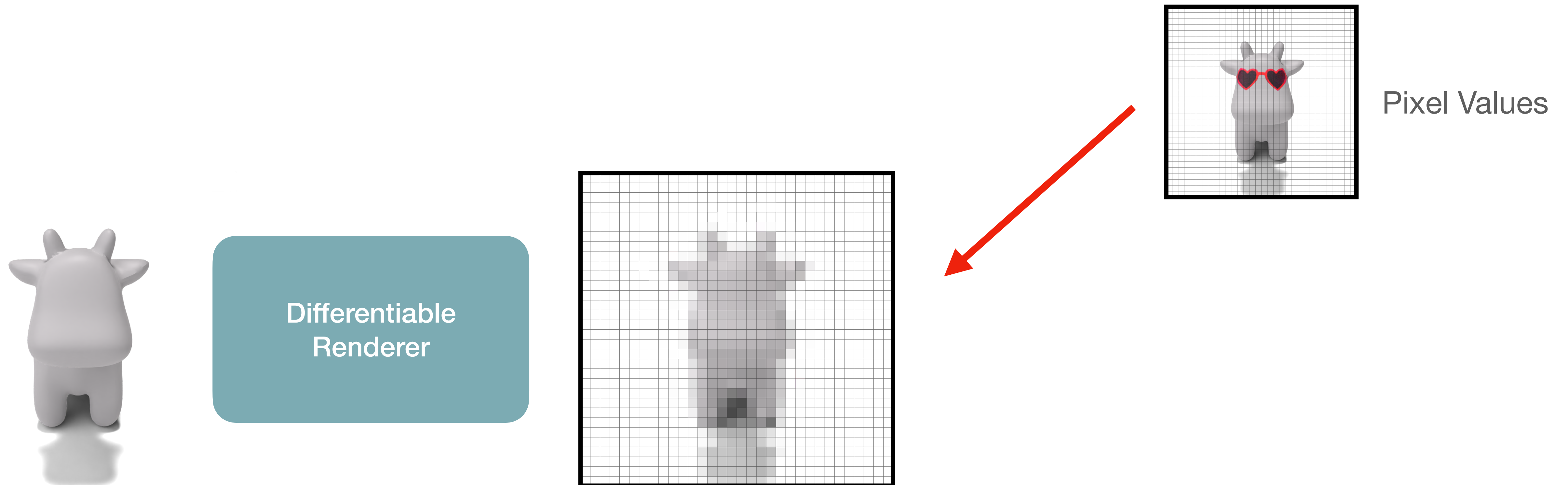
# 3D Optimization With a 2D Signal



Differentiable  
Renderer

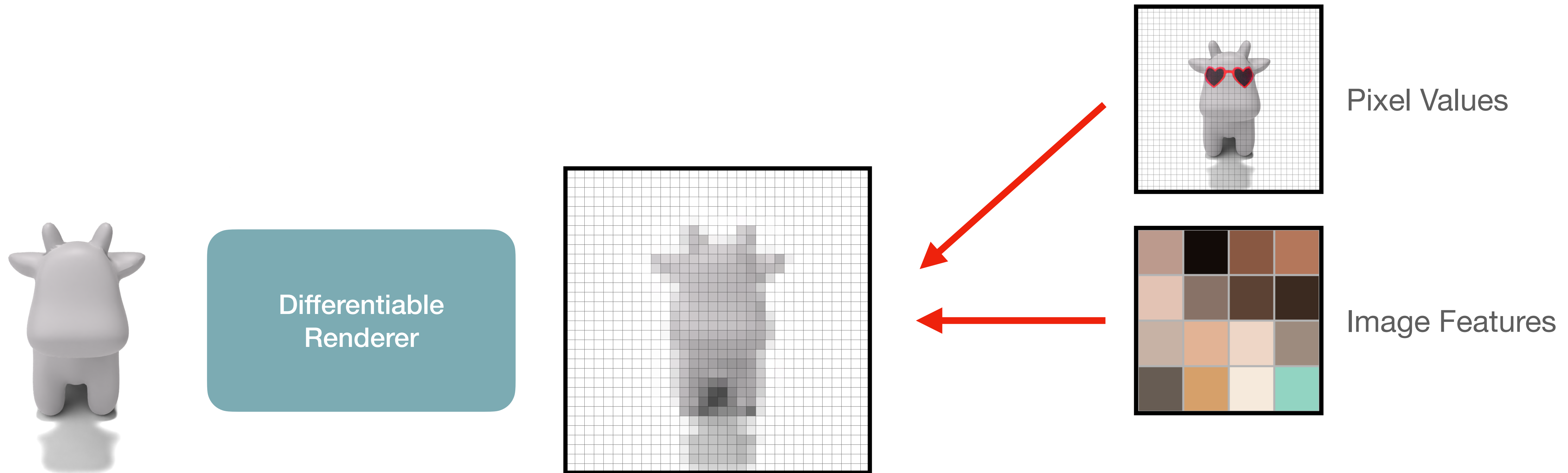


# 3D Optimization With a 2D Signal

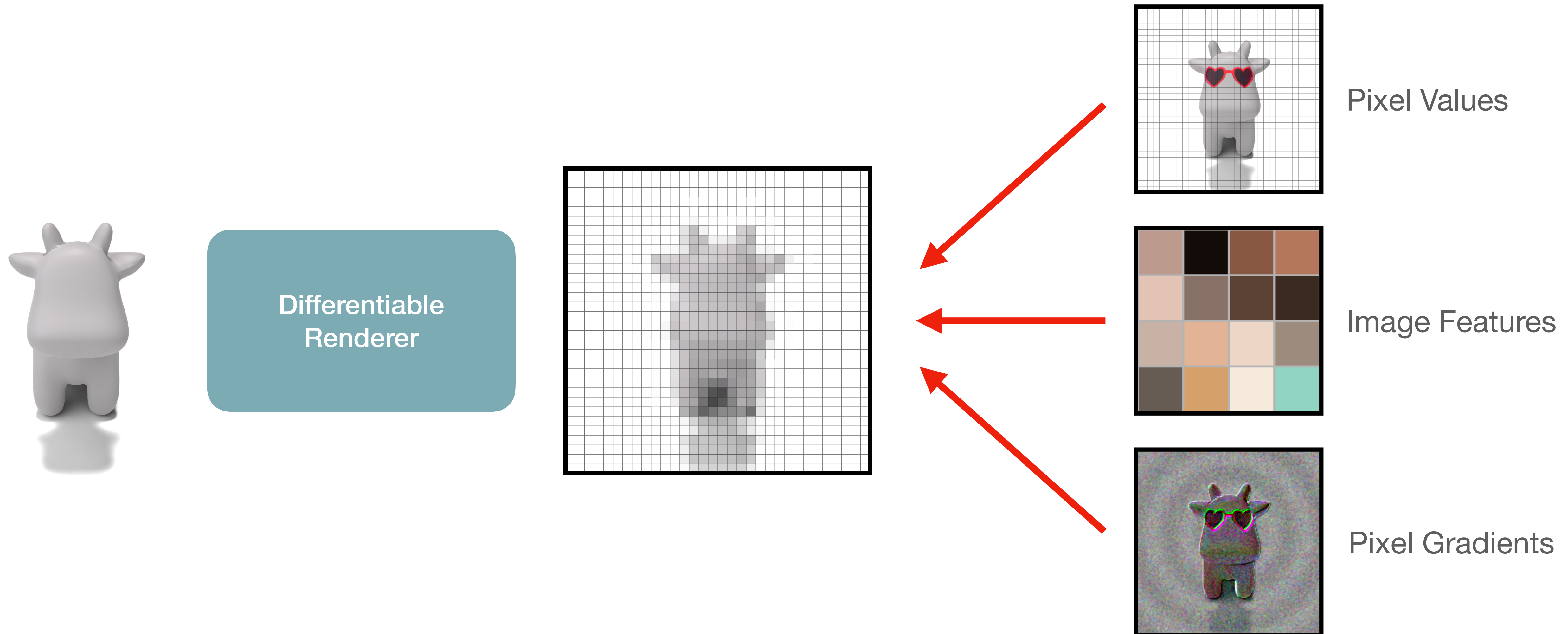




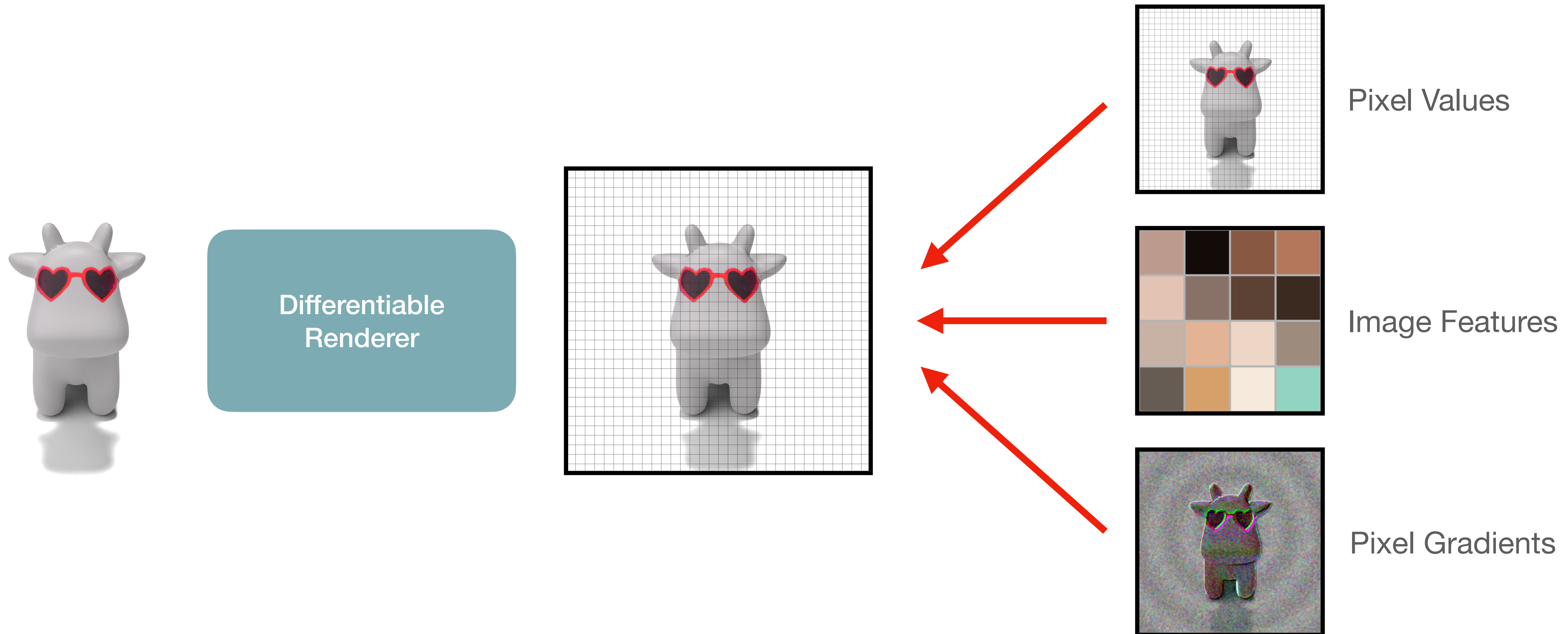
# 3D Optimization With a 2D Signal



# 3D Optimization With a 2D Signal

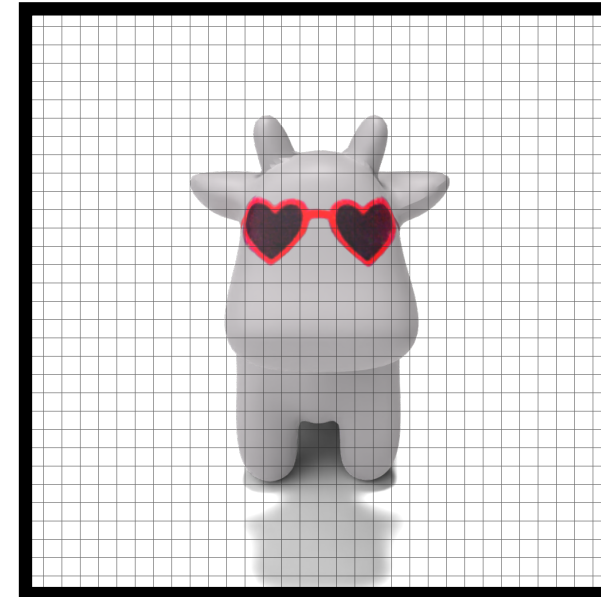


# 3D Optimization With a 2D Signal





# Types of 2D Signal



Pixel Values

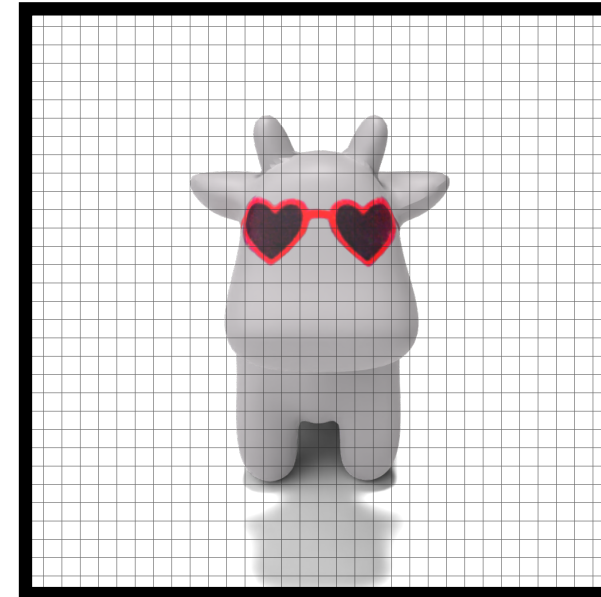


Image Features



Pixel Gradients

# Types of 2D Signal



Pixel Values

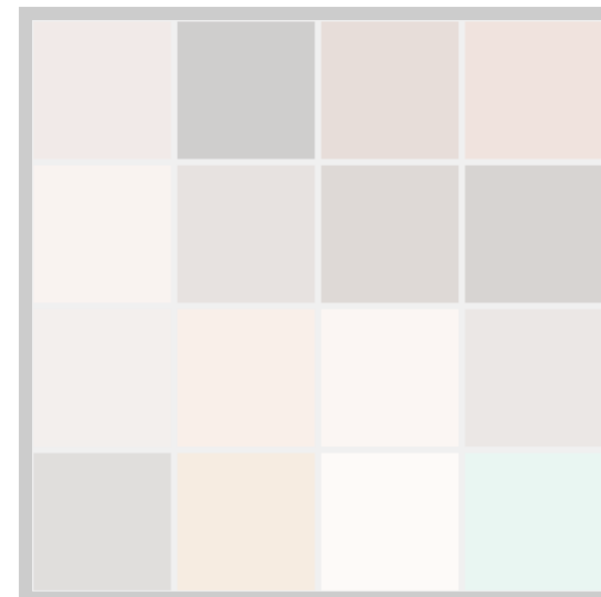


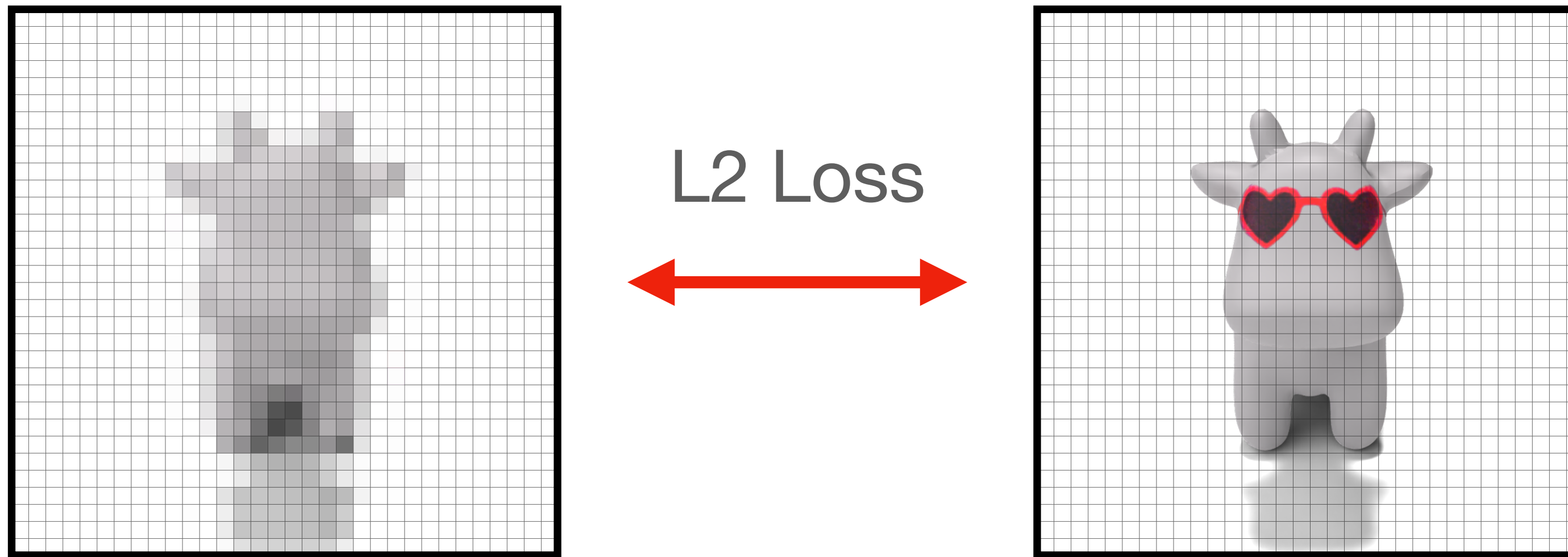
Image Features



Pixel Gradients

# Pixel Supervision

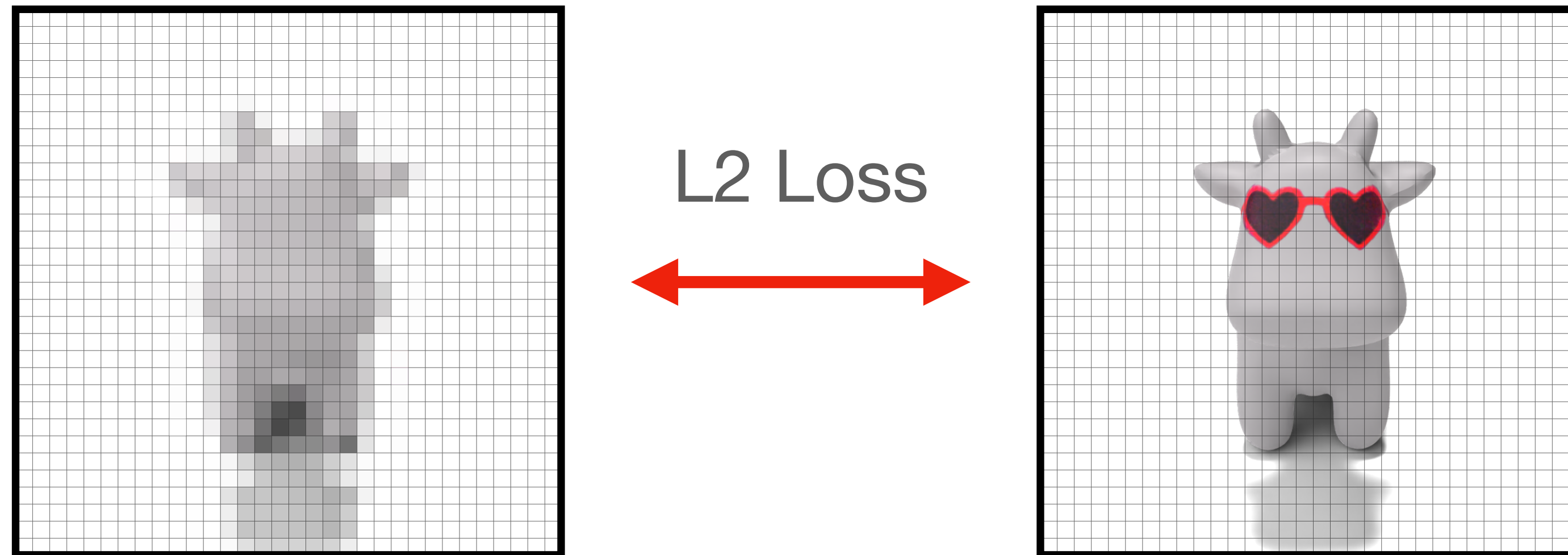
- Directly compute L2 loss on the pixels of each image





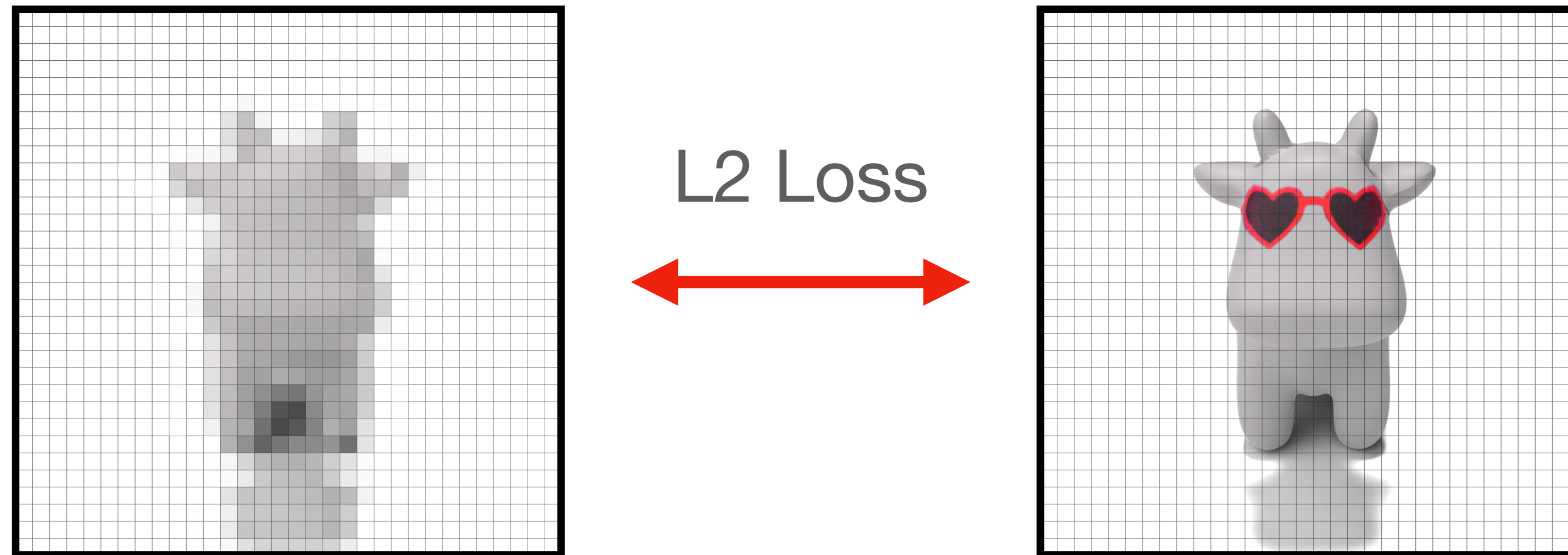
# Pixel Supervision

- Directly compute L2 loss on the pixels of each image
- Useful for when you have exact target images at the pixel level

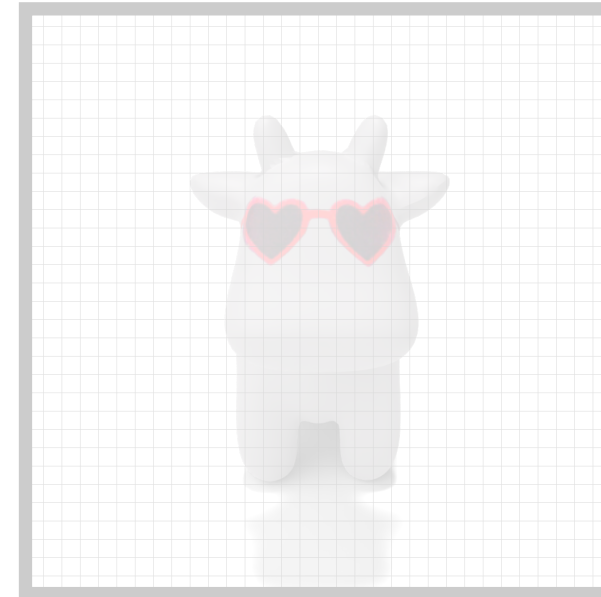


# Pixel Supervision

- Directly compute L2 loss on the pixels of each image
- Useful for when you have exact target images at the pixel level
- Often this is not the case...



# Types of 2D Signal



Pixel Values



Image Features



Pixel Gradients



# Feature Similarity Supervision

Useful for high level, semantic targets

# Feature Similarity Supervision

Useful for high level, semantic targets

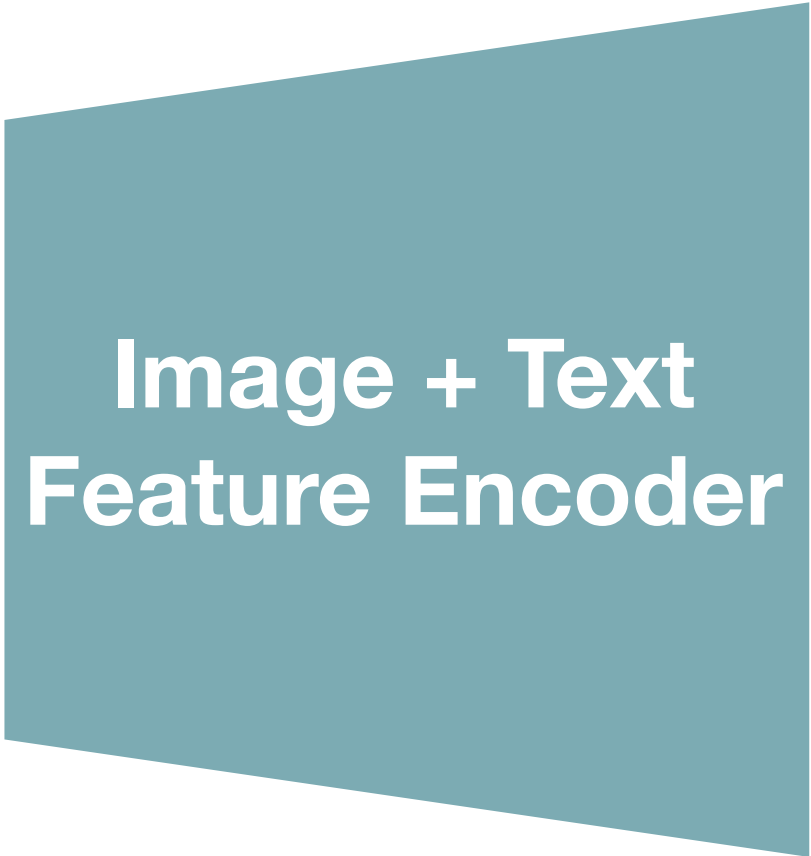


Image + Text  
Feature Encoder

# Feature Similarity Supervision

Useful for high level, semantic targets

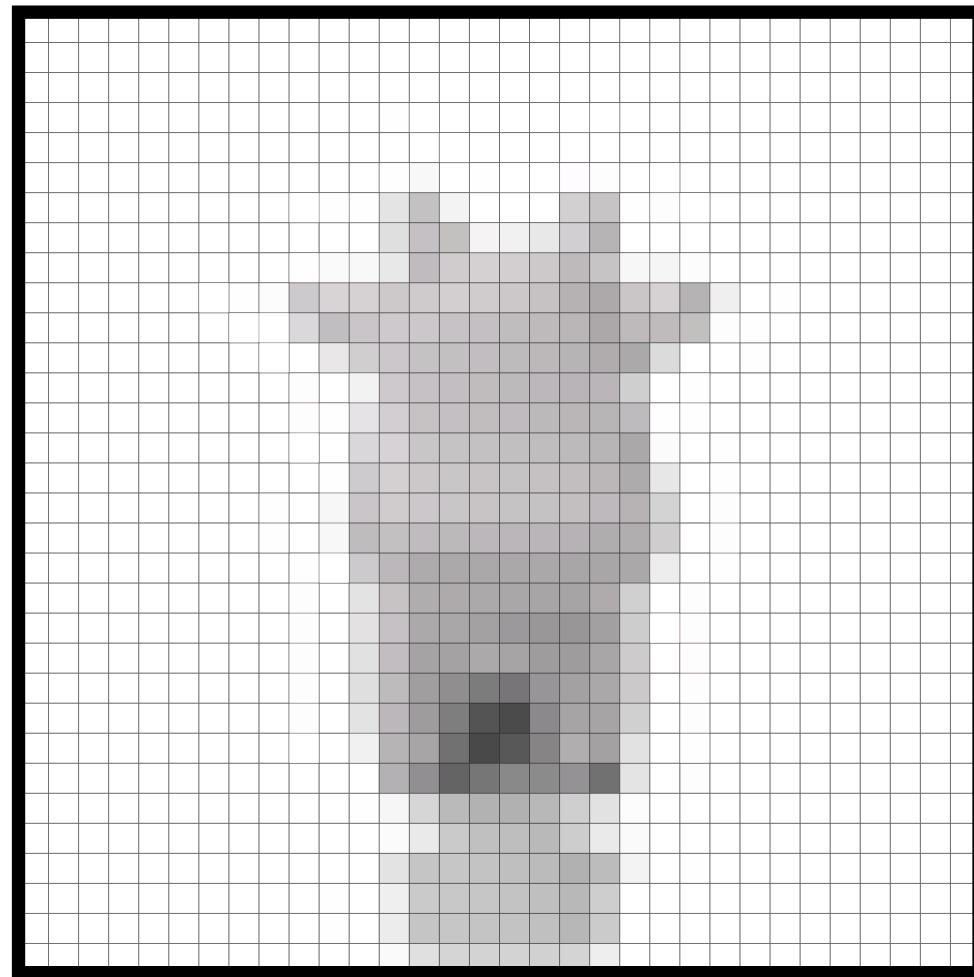


Image + Text  
Feature Encoder



# Feature Similarity Supervision

Useful for high level, semantic targets

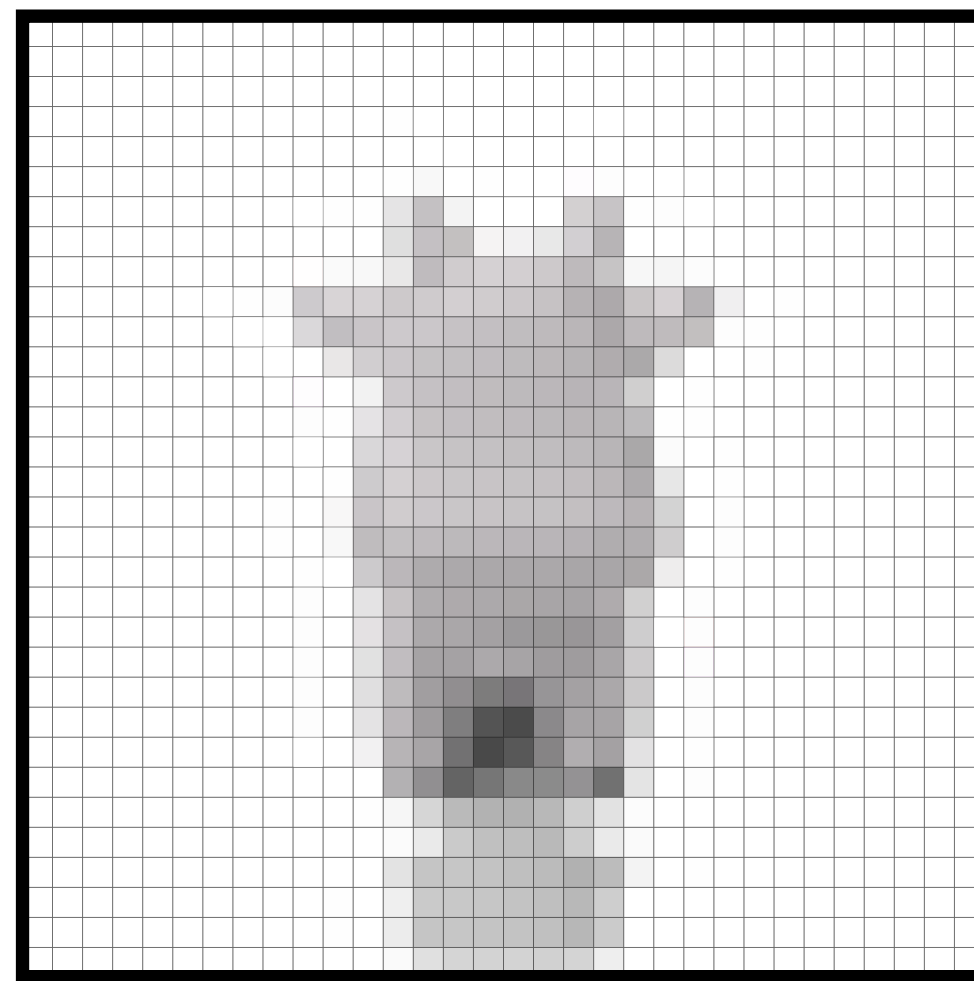


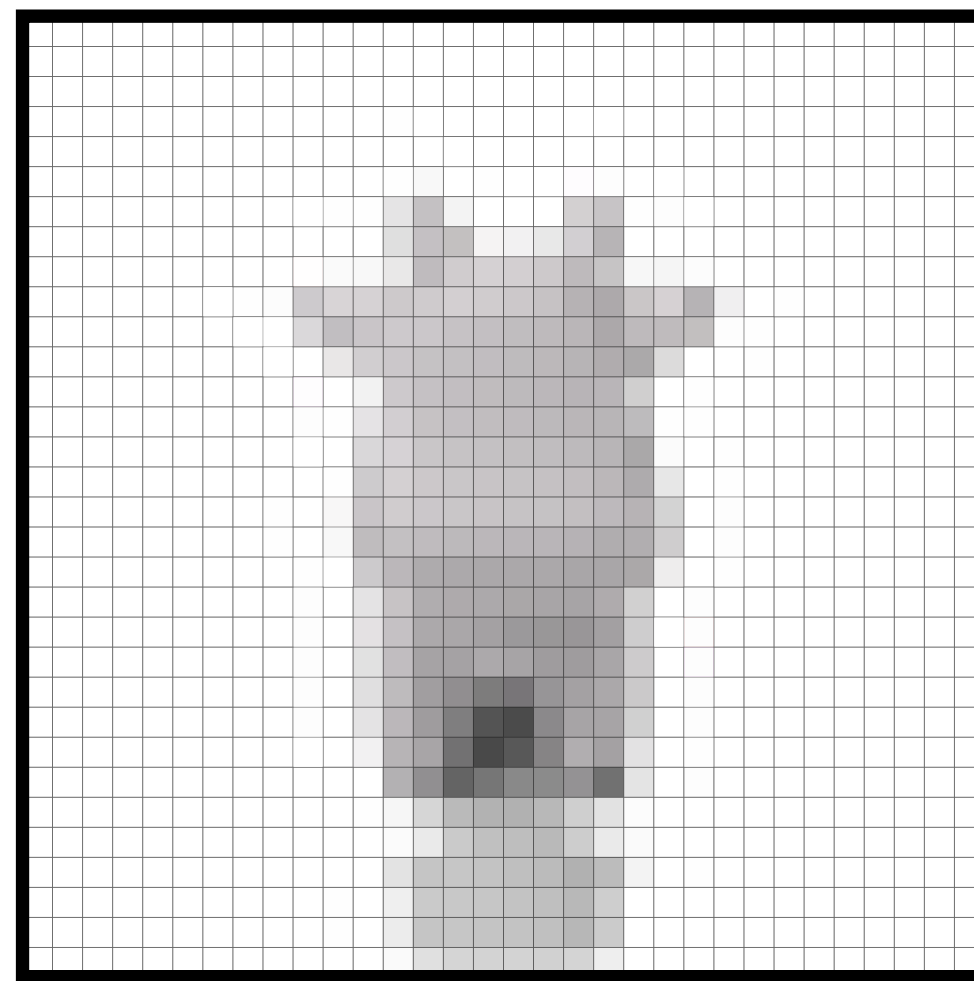
Image + Text  
Feature Encoder





# Feature Similarity Supervision

Useful for high level, semantic targets



*“red heart-shaped sunglasses”*

*or*

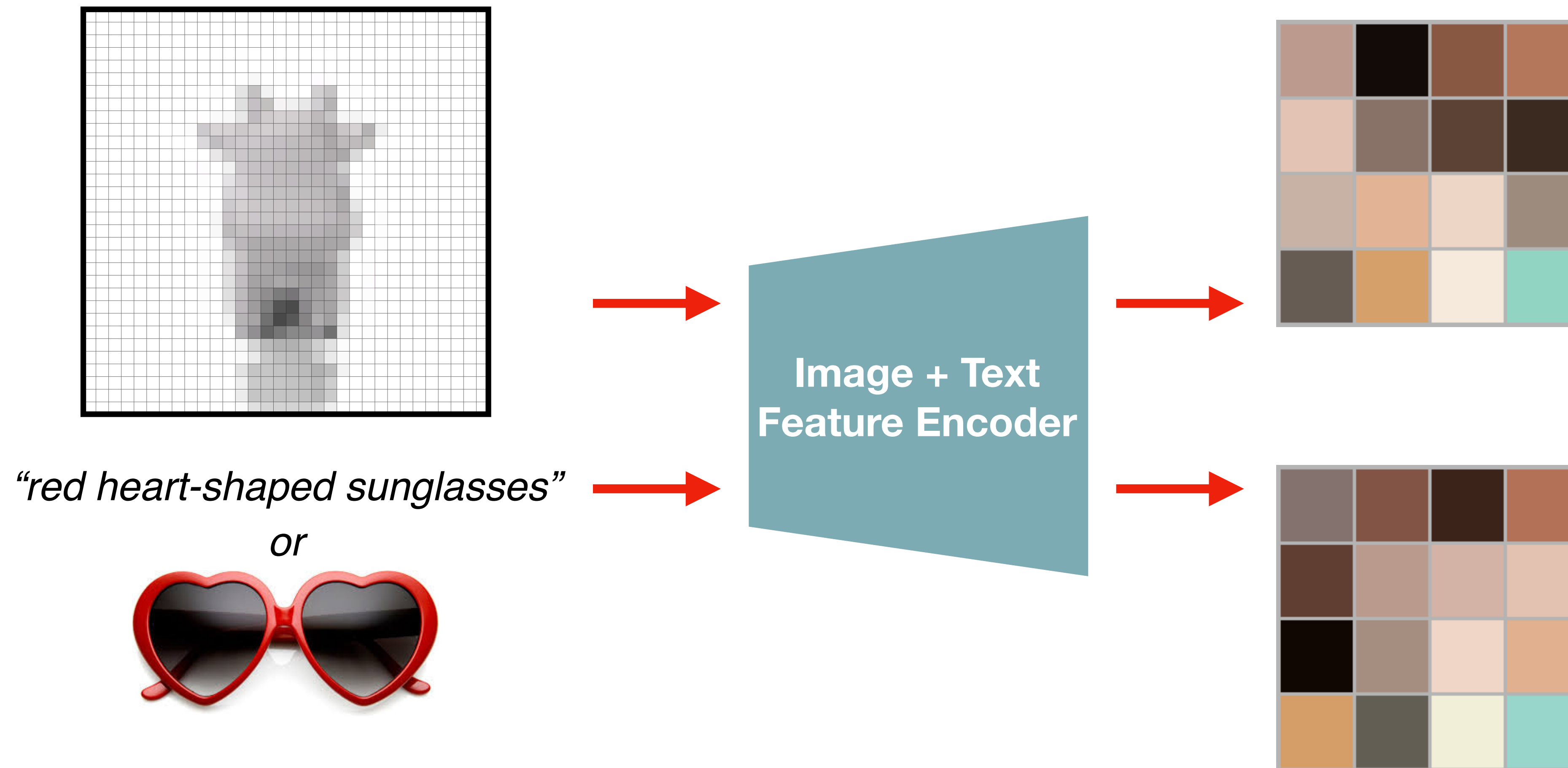


Image + Text  
Feature Encoder



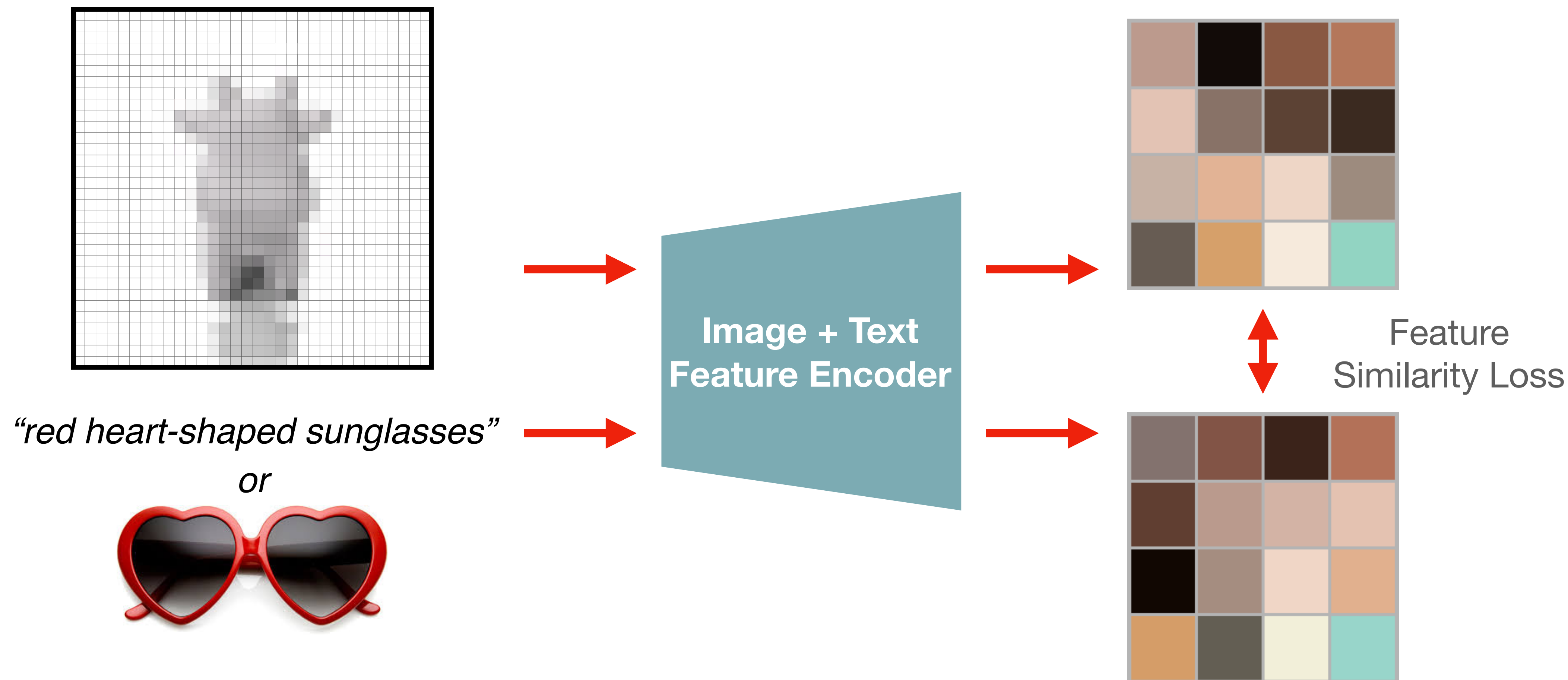
# Feature Similarity Supervision

Useful for high level, semantic targets



# Feature Similarity Supervision

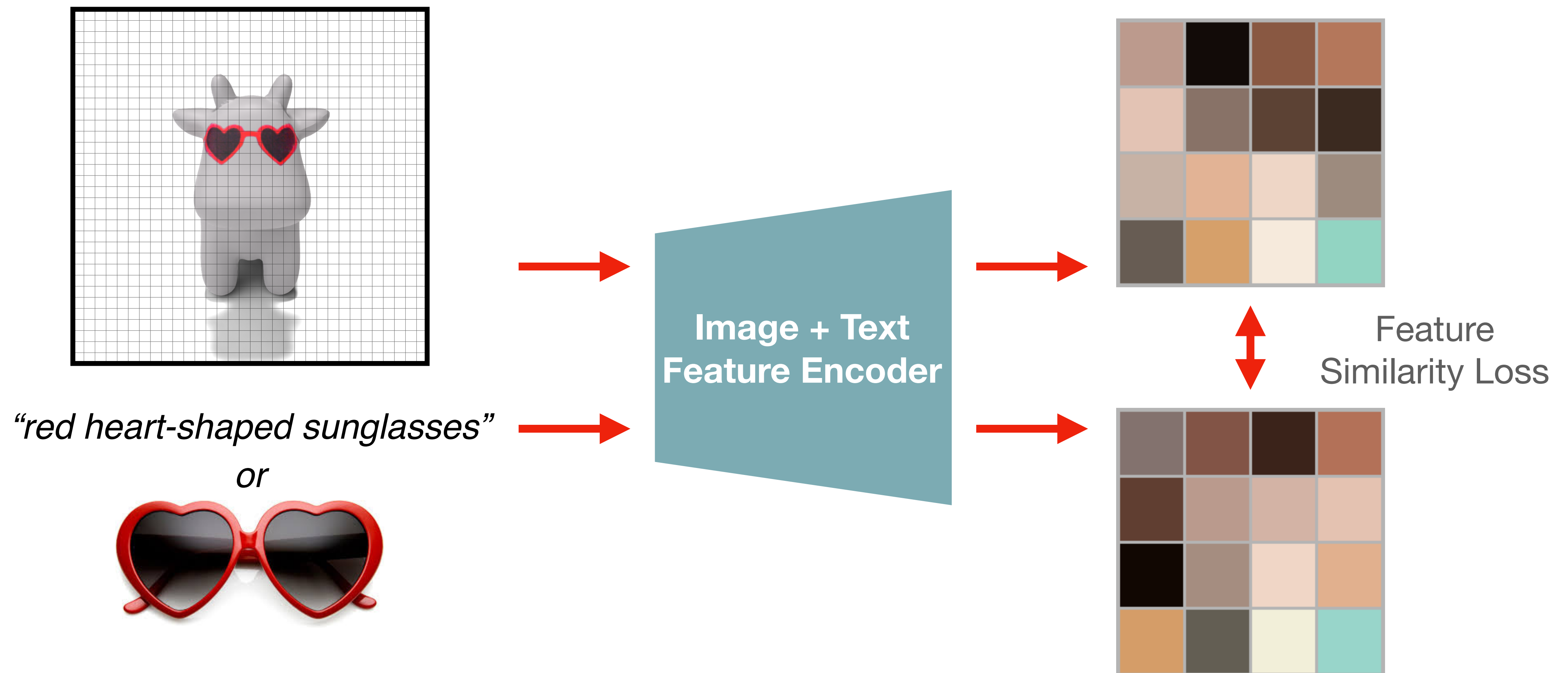
Useful for high level, semantic targets





# Feature Similarity Supervision

Useful for high level, semantic targets

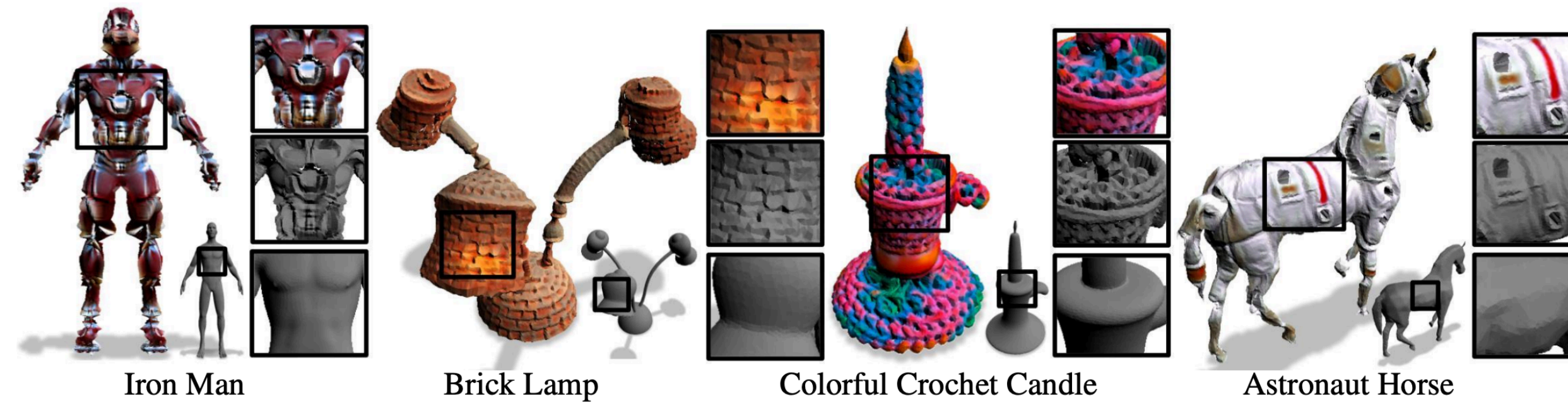




# Feature Similarity Applications — Text-Driven

# Feature Similarity Applications — Text-Driven

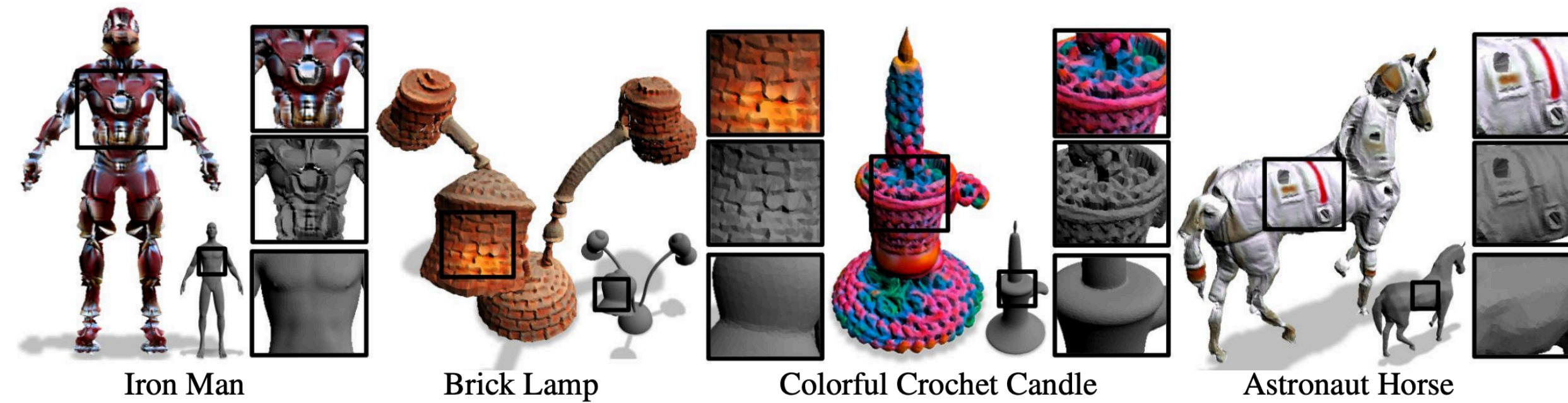
Stylization



Text2Mesh, Mitchel et al., 2022

# Feature Similarity Applications — Text-Driven

Stylization



Text2Mesh, Mitchel et al., 2022

Generation

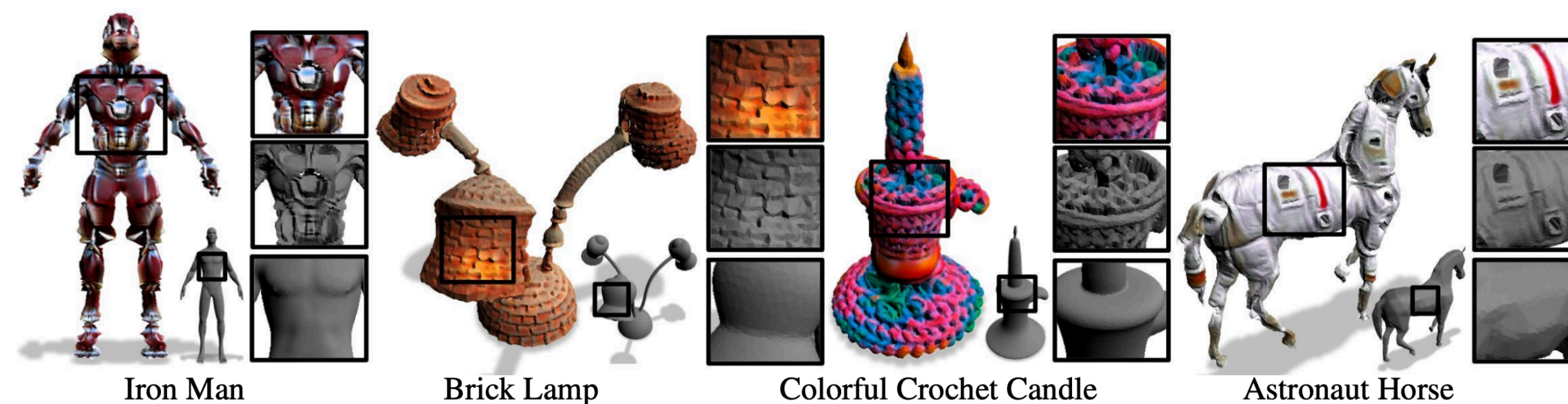


CLIP-Mesh, Khalid et al., 2022



# Feature Similarity Applications — Text-Driven

## Stylization



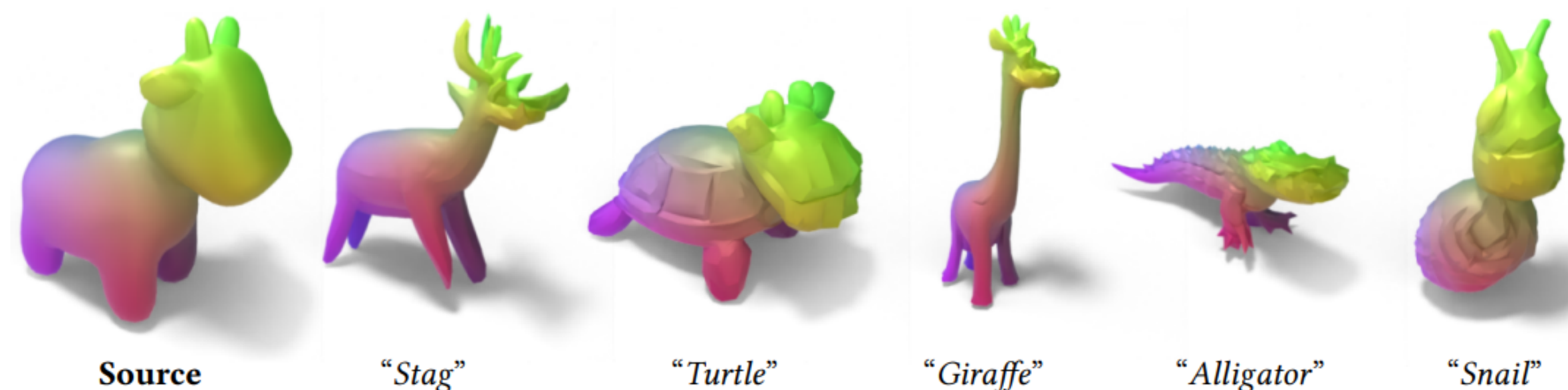
Text2Mesh, Mitchel et al., 2022

## Generation



CLIP-Mesh, Khalid et al., 2022

## Deformation



TextDeformer, Gao et al., 2023



# Feature Similarity Applications — Image Driven

Goal: extract curve abstraction from images of a 3D shape

# Feature Similarity Applications — Image Driven

Goal: extract curve abstraction from images of a 3D shape

- Curves and rendered images to not align at the pixel level



# Feature Similarity Applications — Image Driven

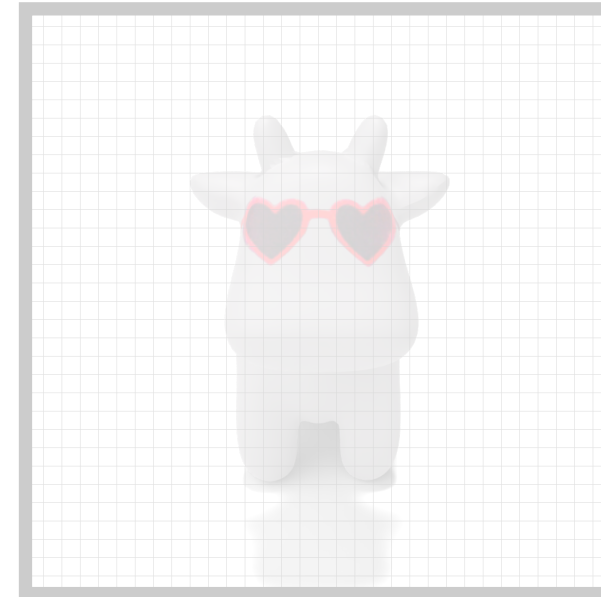
Goal: extract curve abstraction from images of a 3D shape

- Curves and rendered images do not align at the pixel level
- However, they are aligned in semantic feature space!





# Types of 2D Signal



Pixel Values

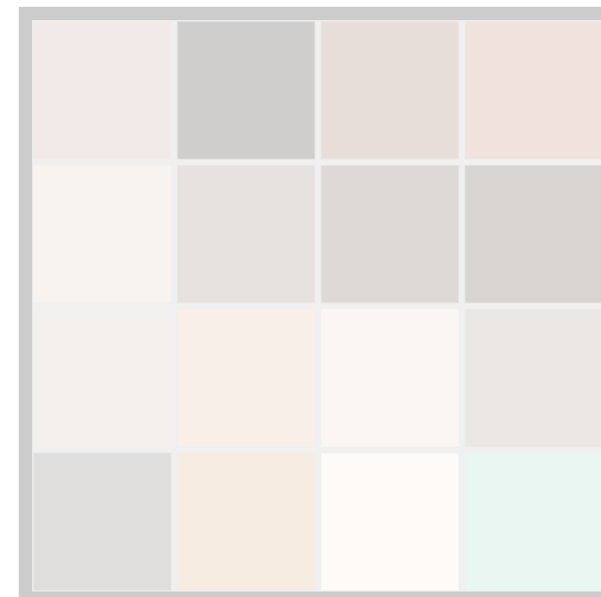


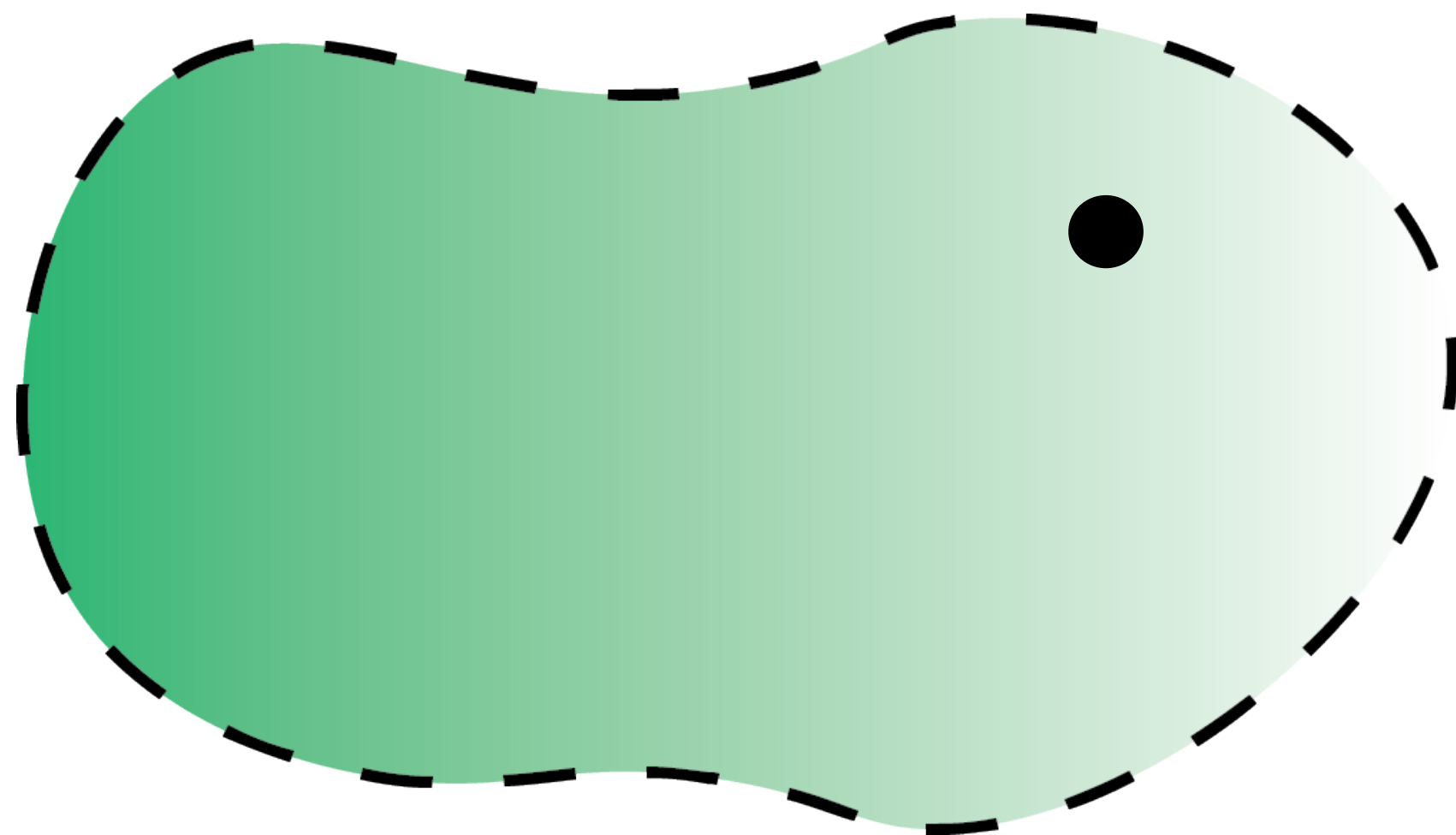
Image Features



Pixel Gradients



# Score Distillation — Diffusion Models



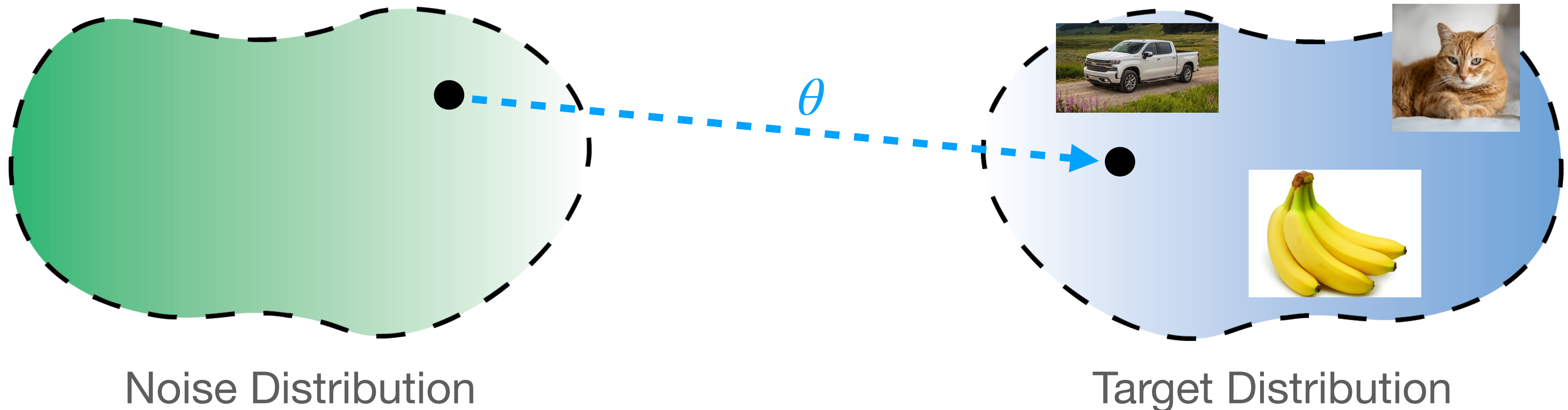
Noise Distribution



Target Distribution

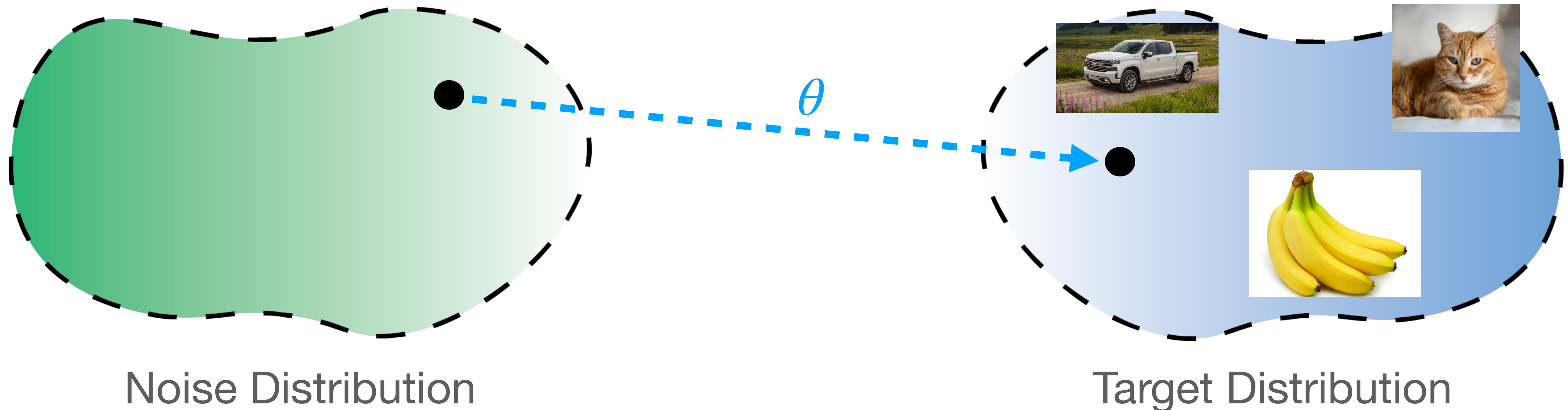
# Score Distillation — Diffusion Models

- Diffusion models learn a mapping between distributions



# Score Distillation — Diffusion Models

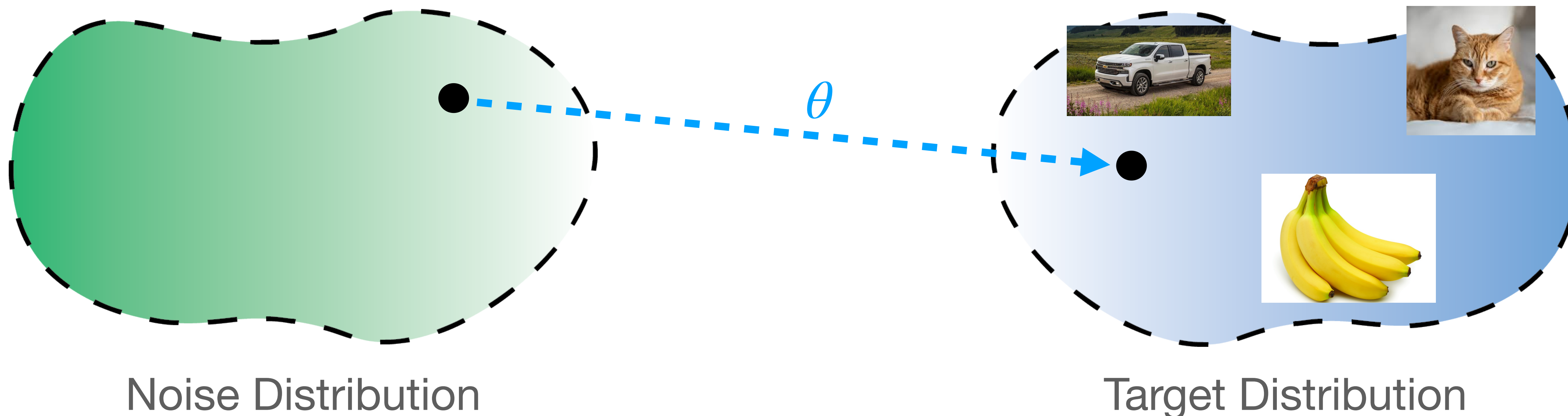
- Diffusion models learn a mapping between distributions
- During training: add noise to images, learn to de-noise





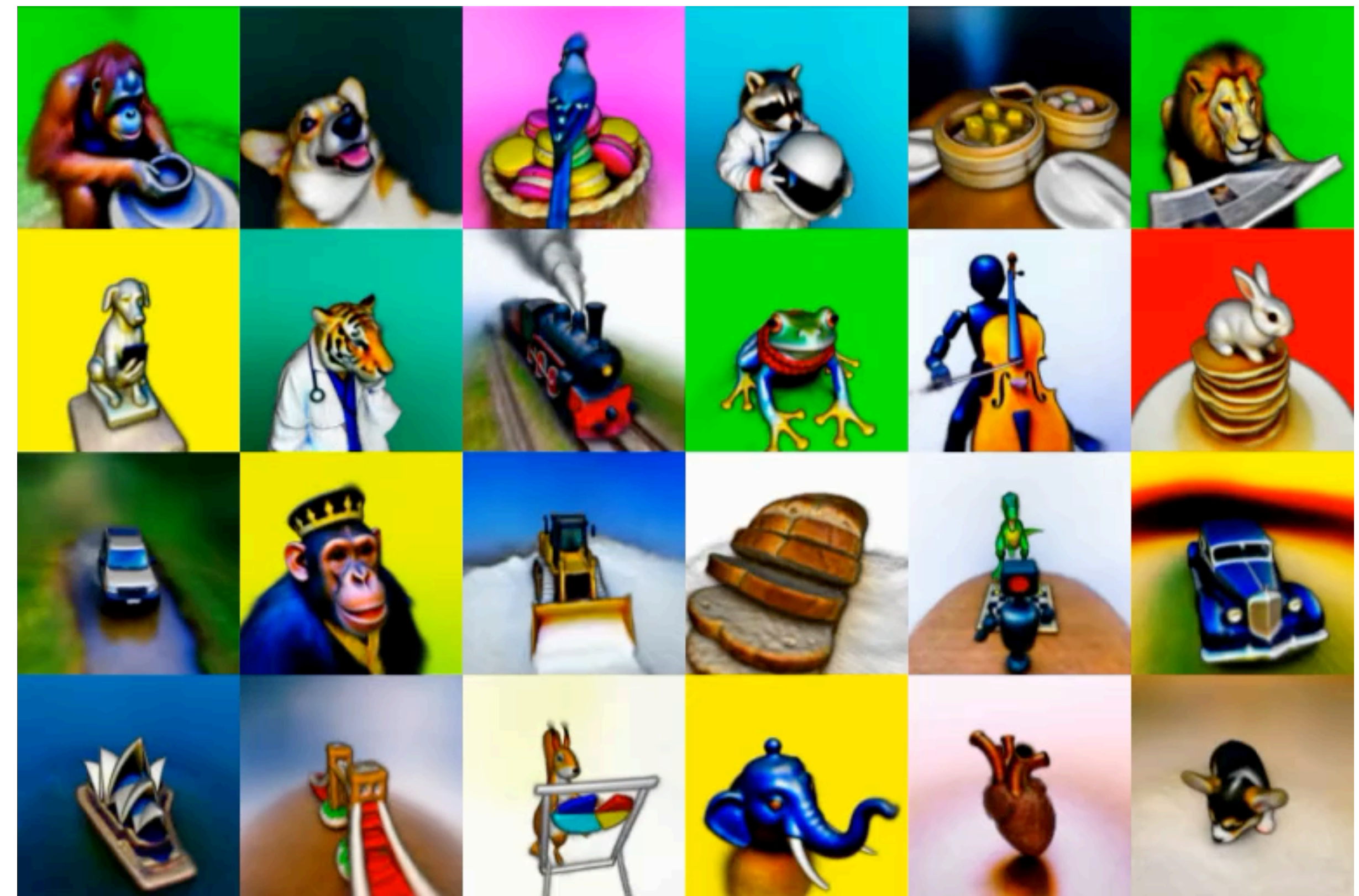
# Score Distillation — Diffusion Models

- Diffusion models learn a mapping between distributions
- During training: add noise to images, learn to de-noise
- Can we use this to supervise our renders?





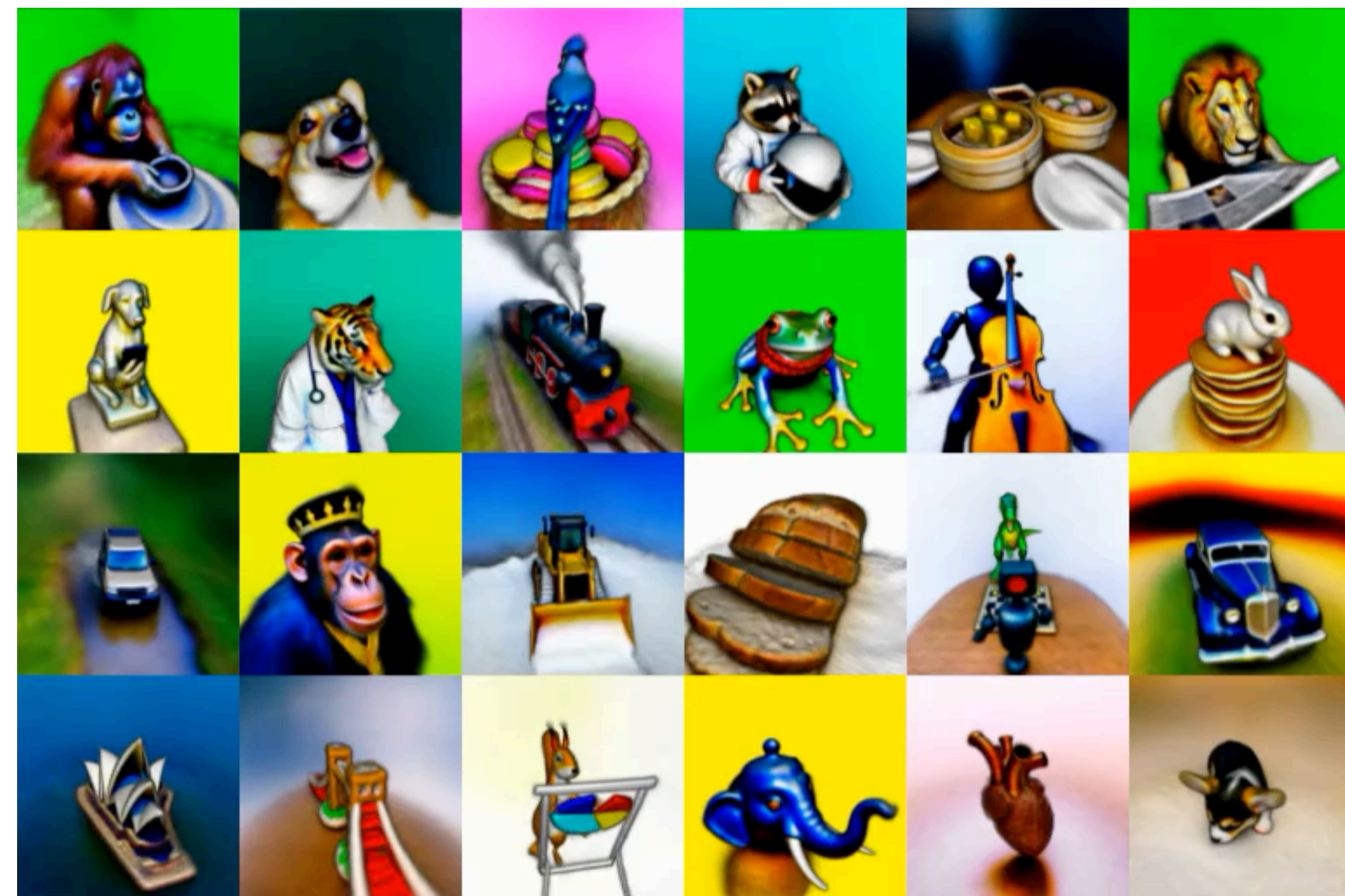
# Score Distillation Sampling (SDS)



DreamFusion, Poole et al., 2023



# Score Distillation Sampling (SDS)

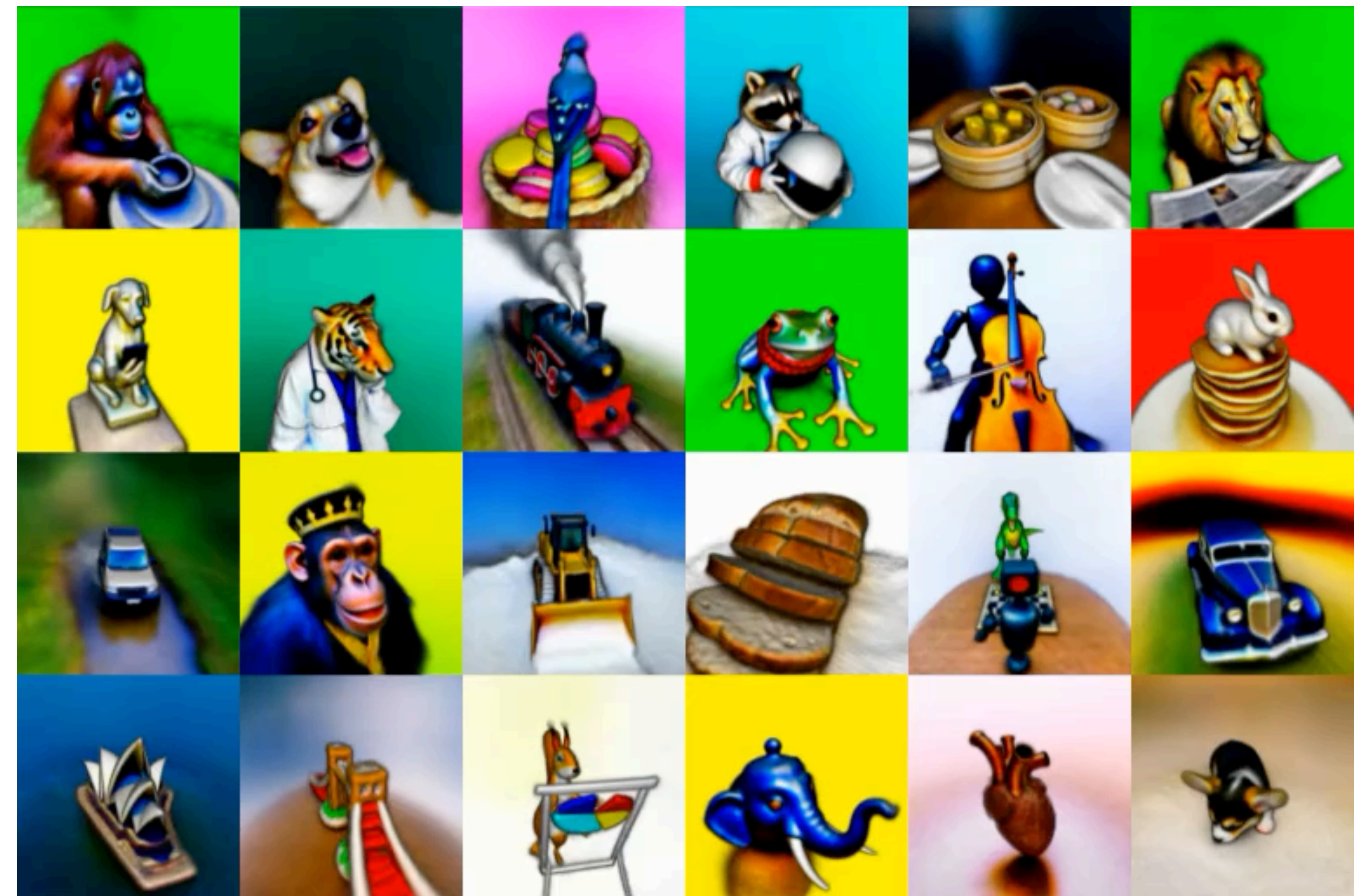


DreamFusion, Poole et al., 2023



# Score Distillation Sampling (SDS)

- Score Distillation Sampling does exactly this!

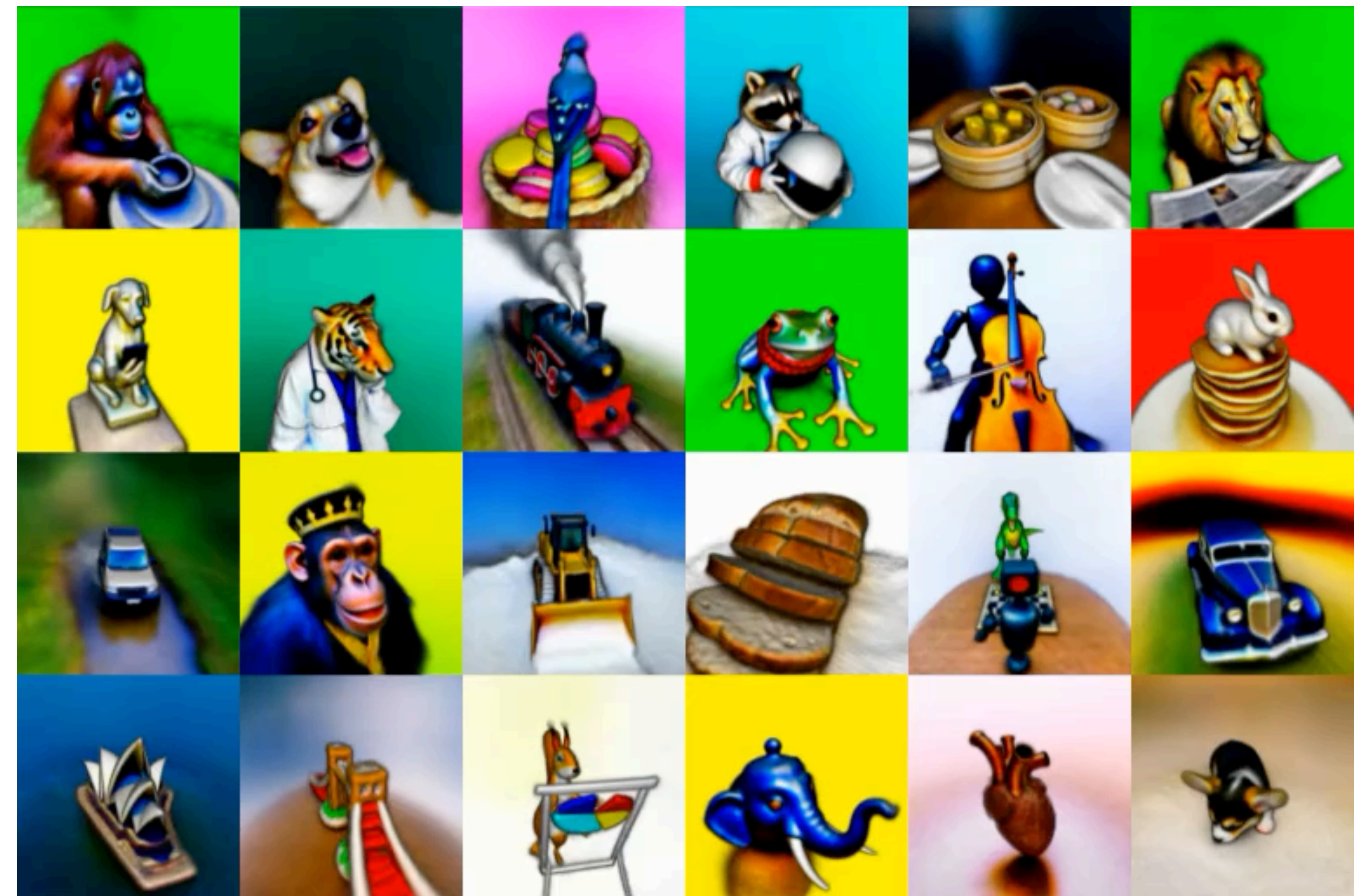


DreamFusion, Poole et al., 2023



# Score Distillation Sampling (SDS)

- Score Distillation Sampling does exactly this!
- Use the diffusion training objective

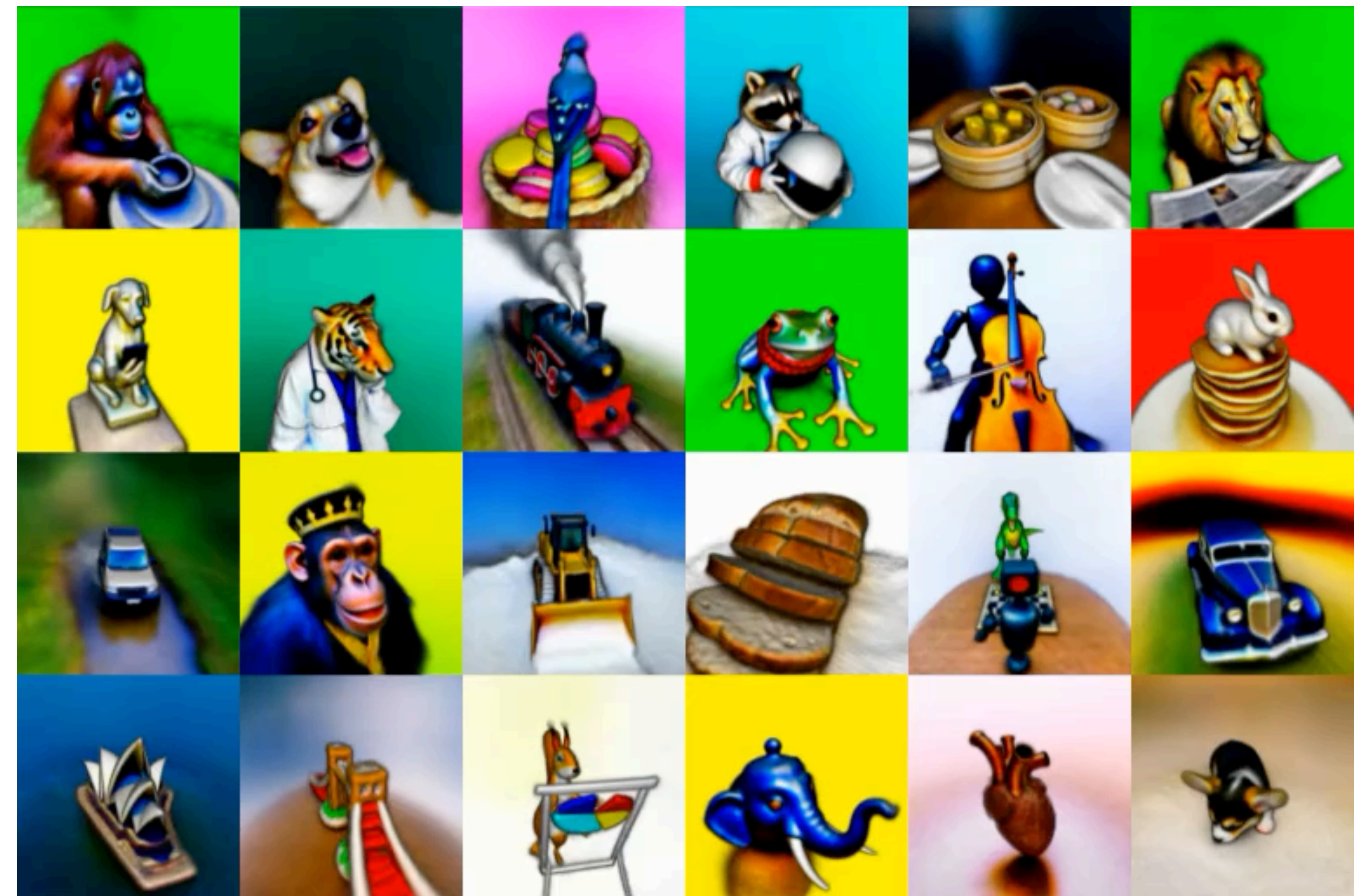


DreamFusion, Poole et al., 2023



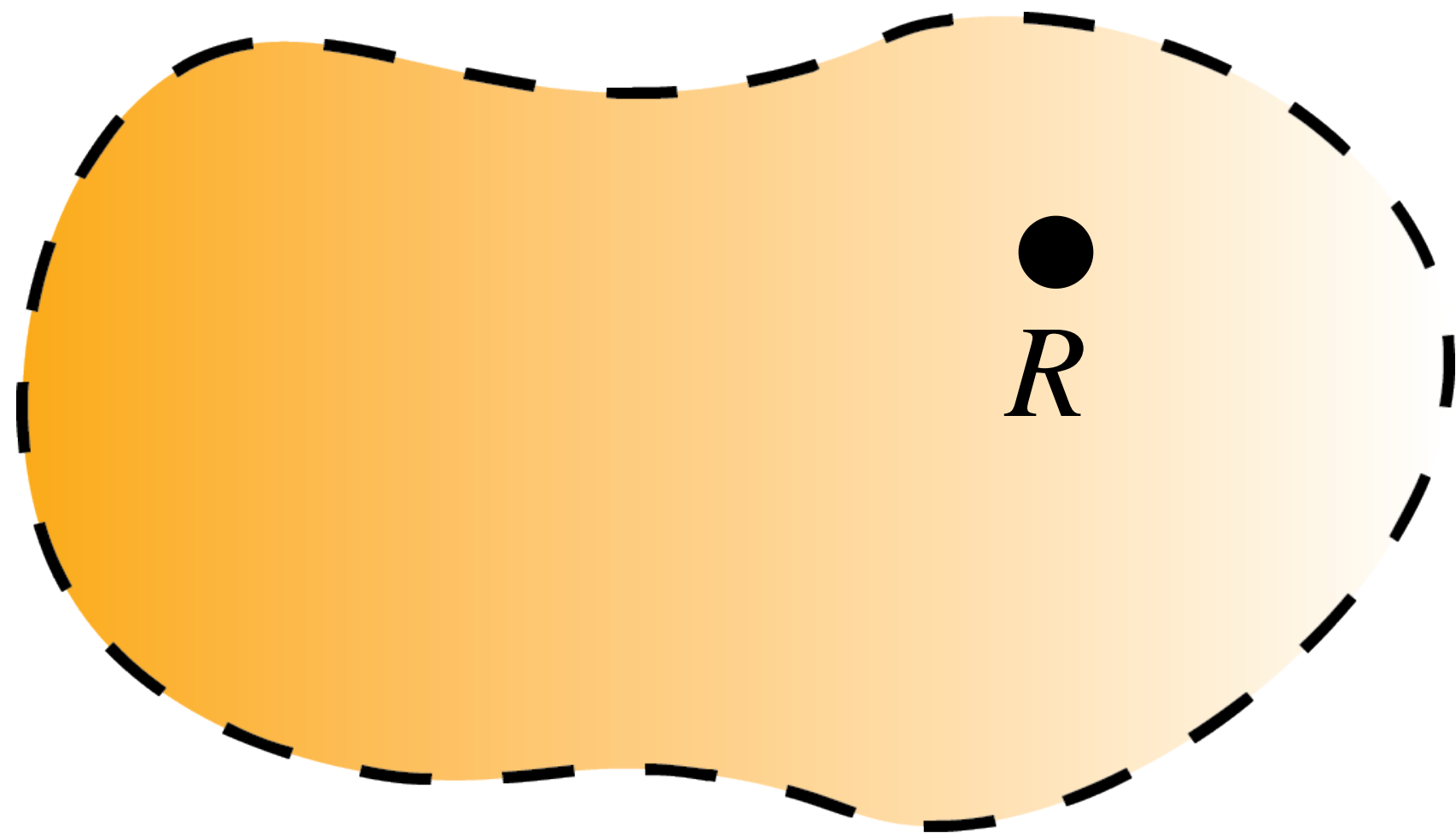
# Score Distillation Sampling (SDS)

- Score Distillation Sampling does exactly this!
- Use the diffusion training objective
- Instead of minimizing the loss w.r.t. the model parameters, do so w.r.t the 2D rendered image

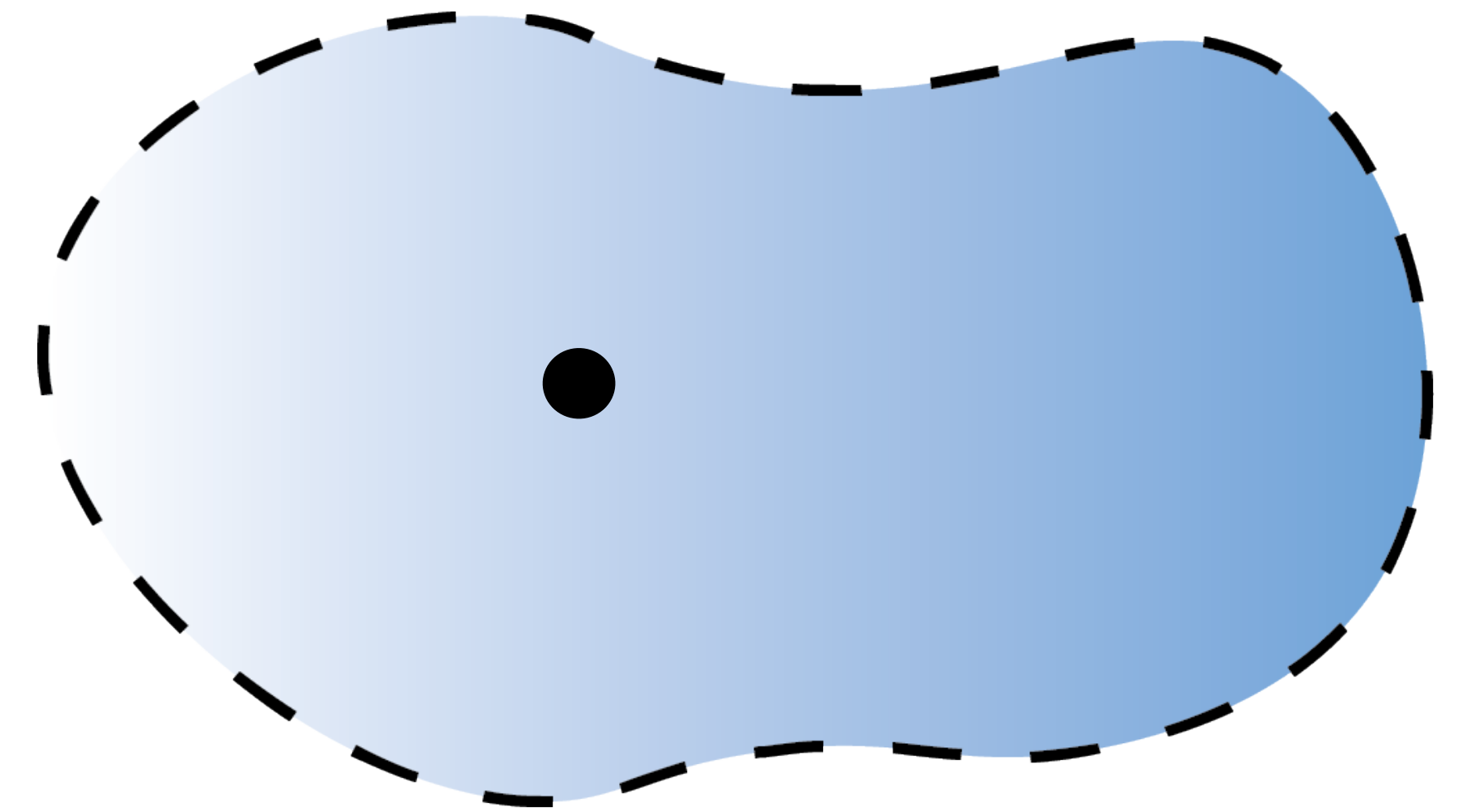


DreamFusion, Poole et al., 2023

# Score Distillation Sampling Visual Intuition

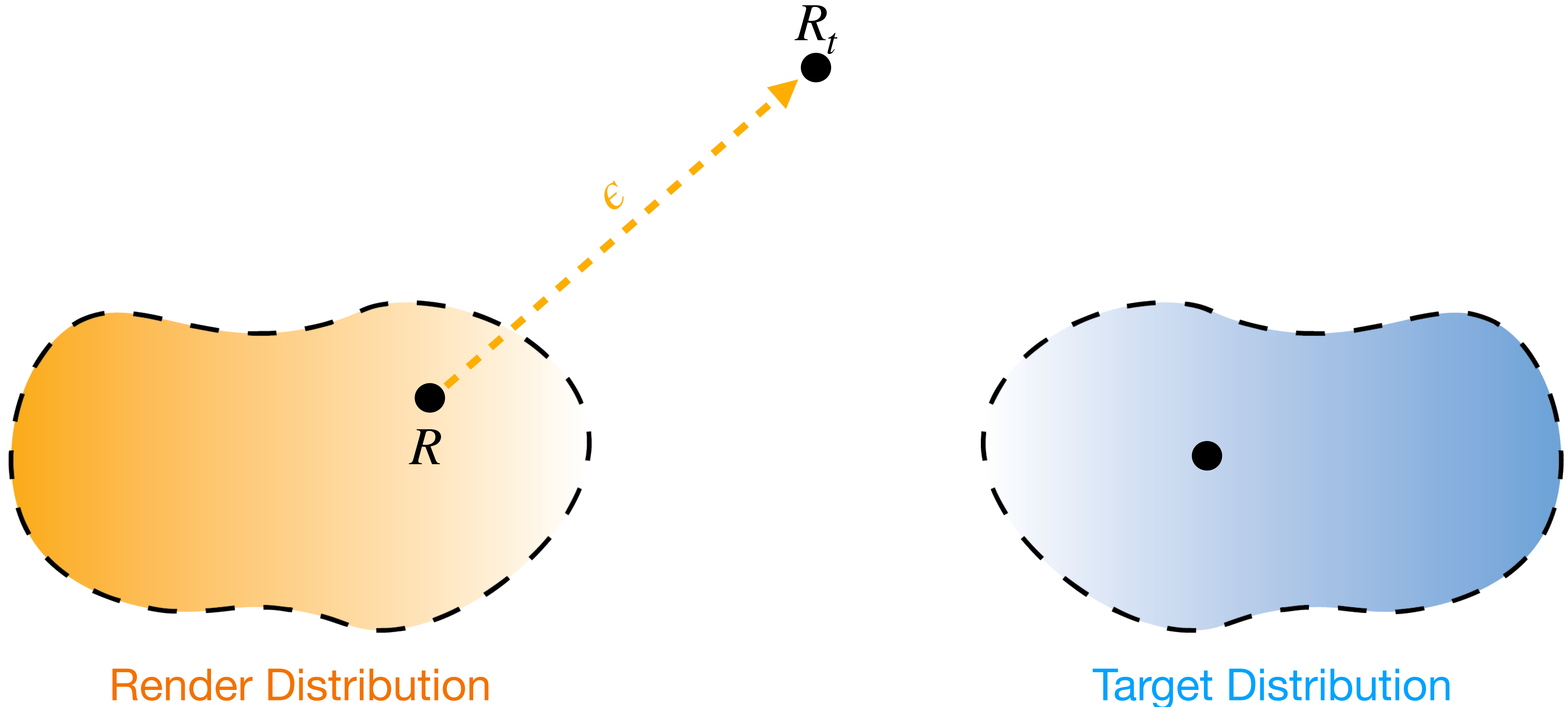


Render Distribution



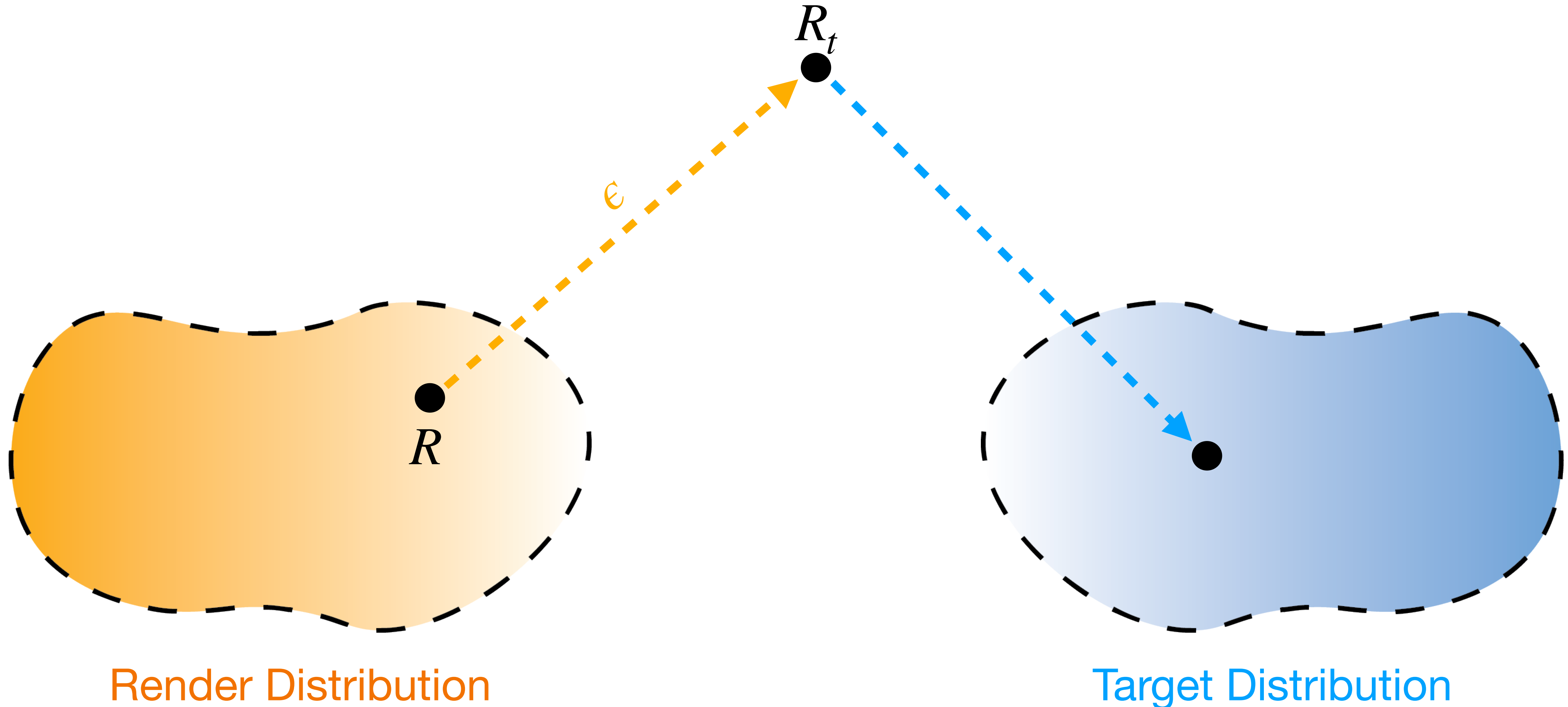
Target Distribution

# Score Distillation Sampling Visual Intuition

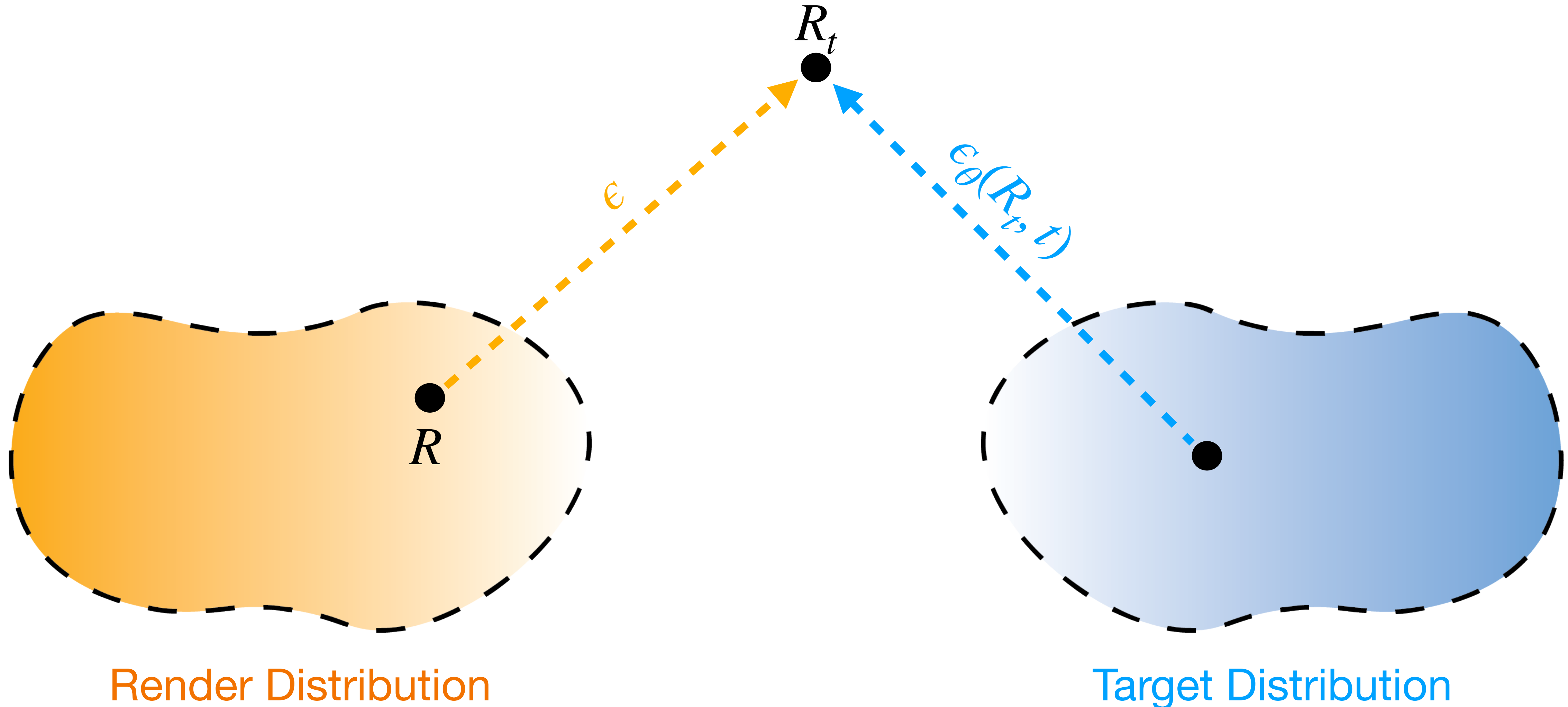




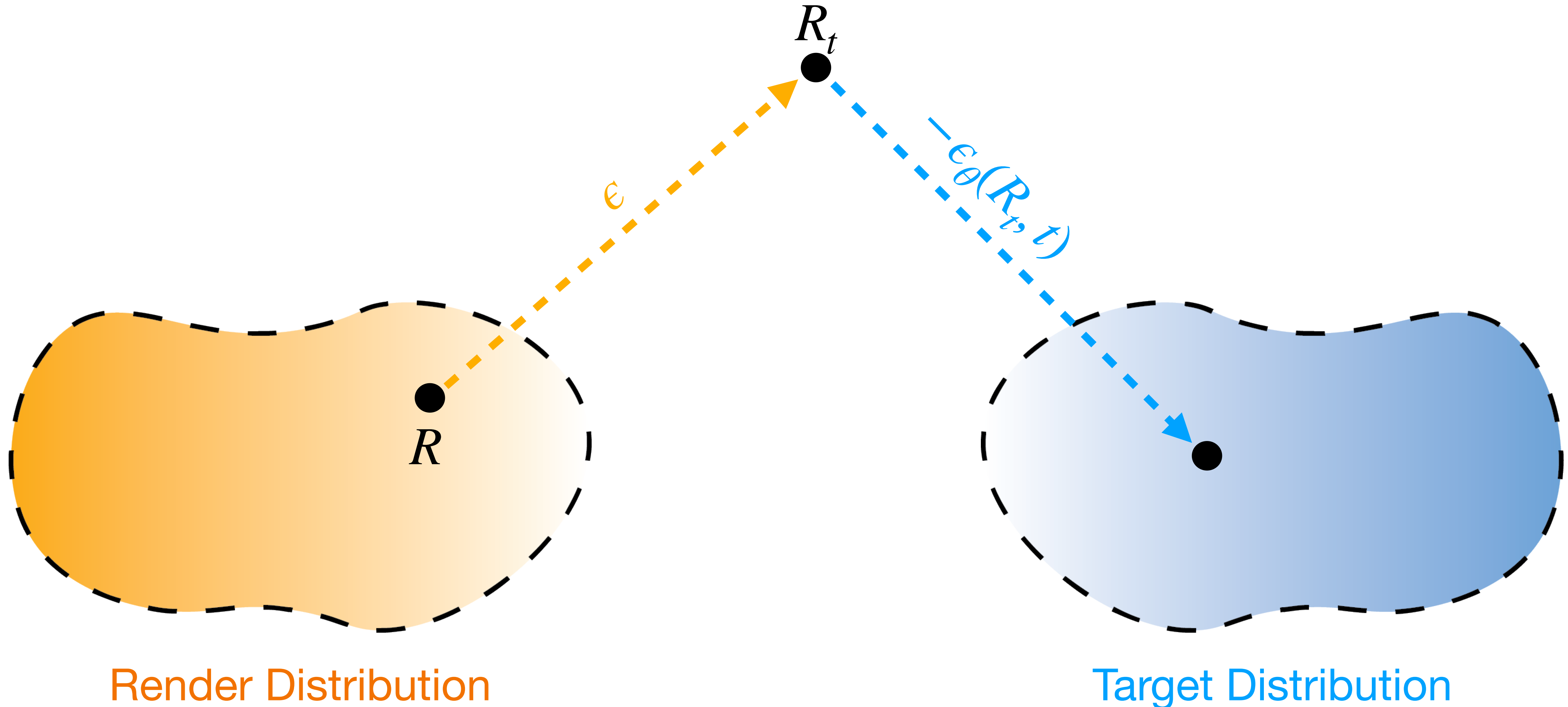
# Score Distillation Sampling Visual Intuition



# Score Distillation Sampling Visual Intuition

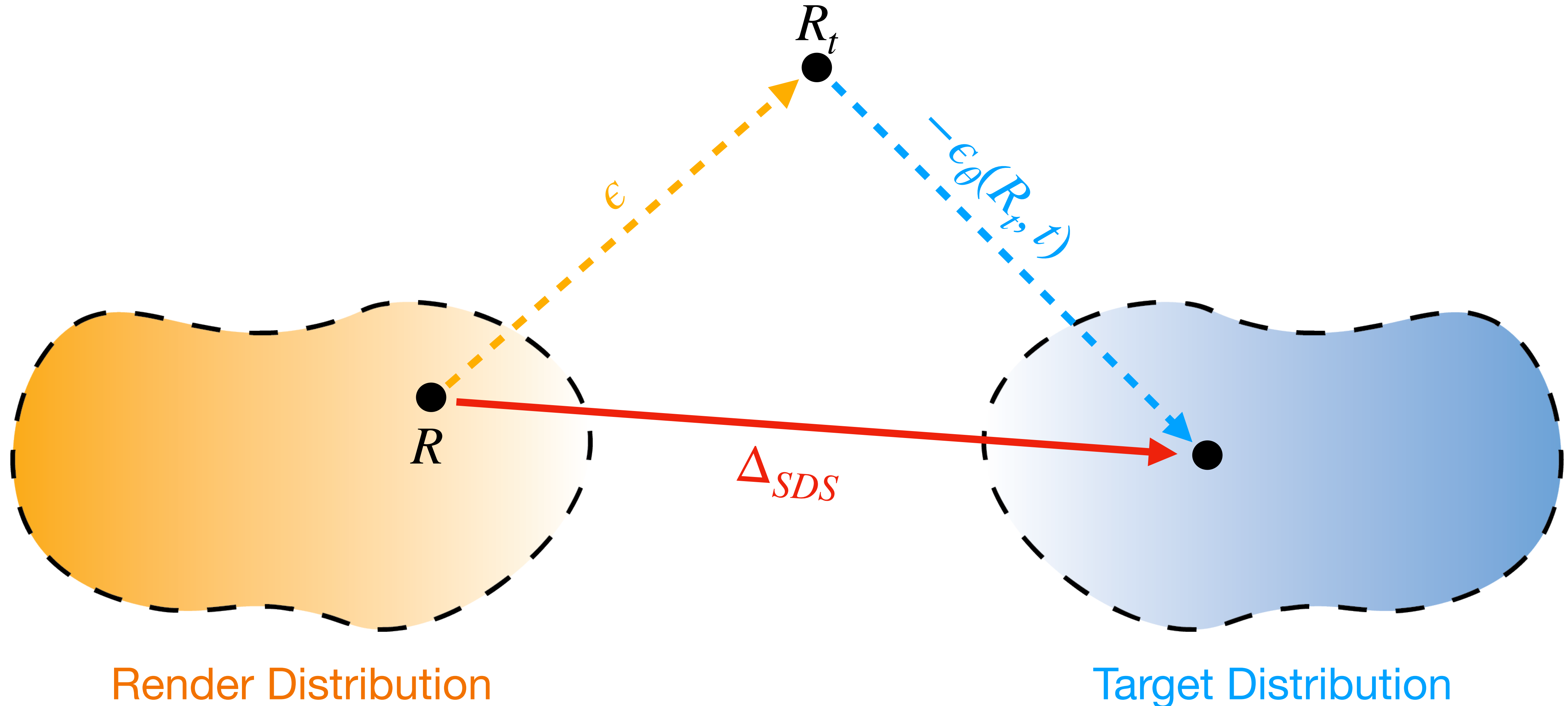


# Score Distillation Sampling Visual Intuition





# Score Distillation Sampling Visual Intuition



# SDS Extensions: Higher Resolution



# SDS Extensions: Higher Resolution

- Can we get higher resolution supervision than standard SDS?





# SDS Extensions: Higher Resolution

- Can we get higher resolution supervision than standard SDS?
- Standard pixel-based SDS supervises at 64x64 resolution



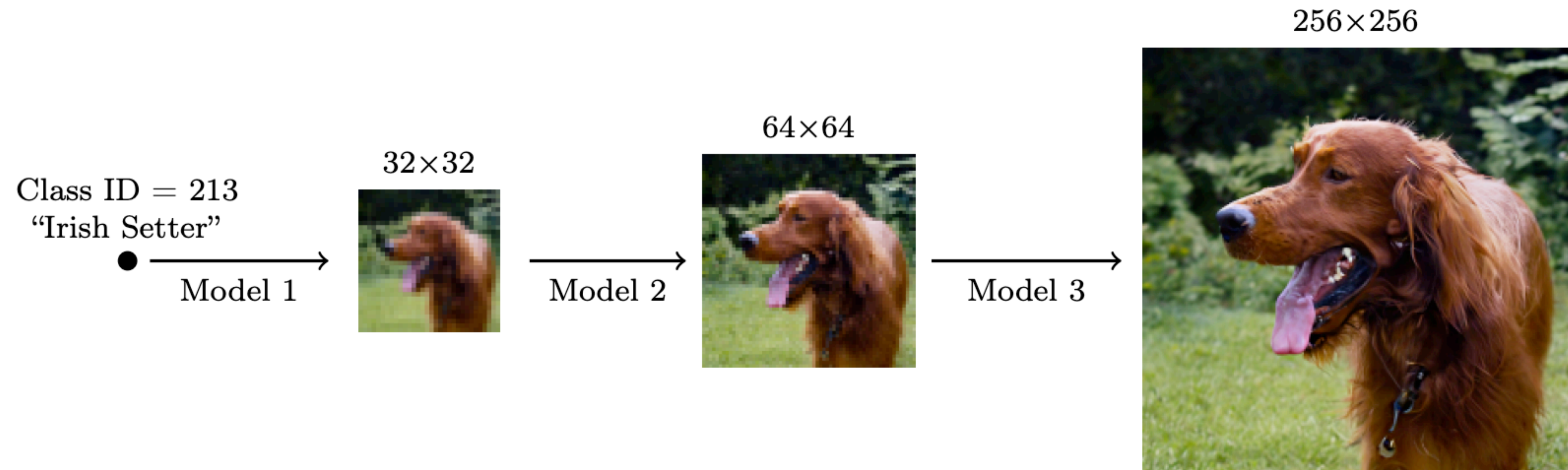
# SDS Extensions: Higher Resolution

- Can we get higher resolution supervision than standard SDS?
- Standard pixel-based SDS supervises at 64x64 resolution
- This is lower resolution than most image models generate at... Why?





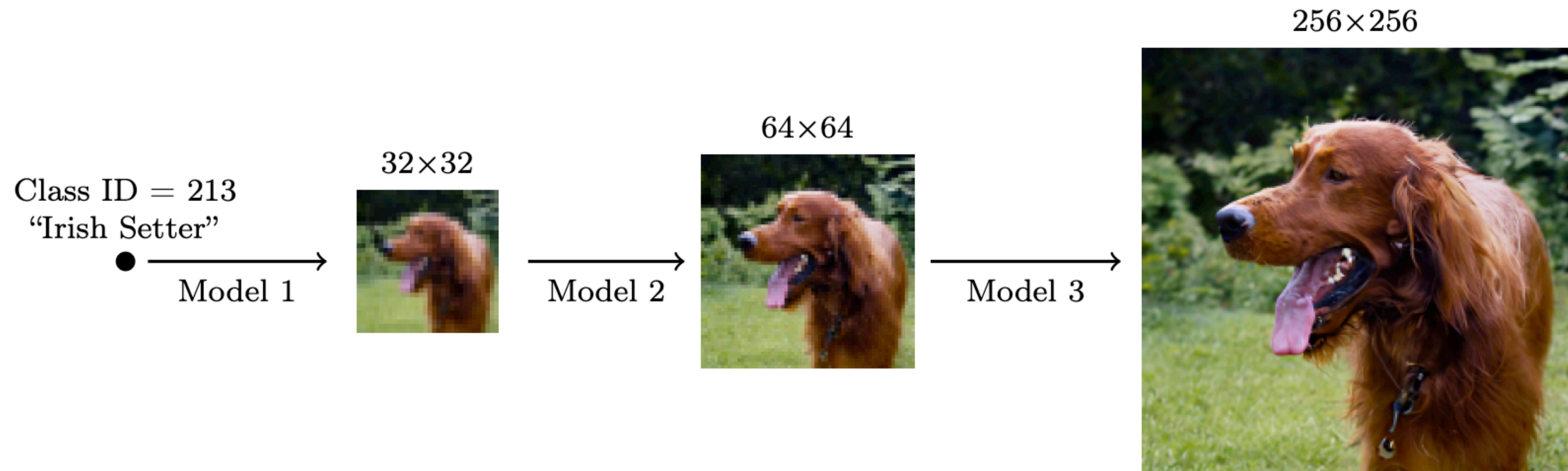
# Cascaded Diffusion Models (CDMs)





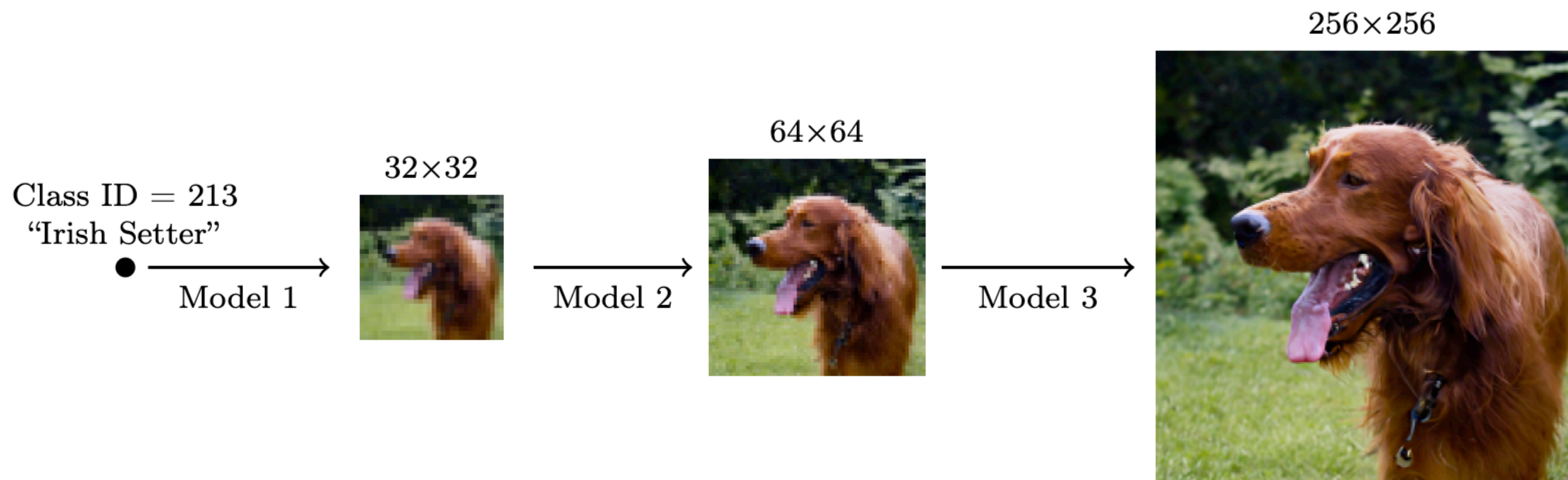
# Cascaded Diffusion Models (CDMs)

- CDMs used to increase resolution for 2D diffusion models



# Cascaded Diffusion Models (CDMs)

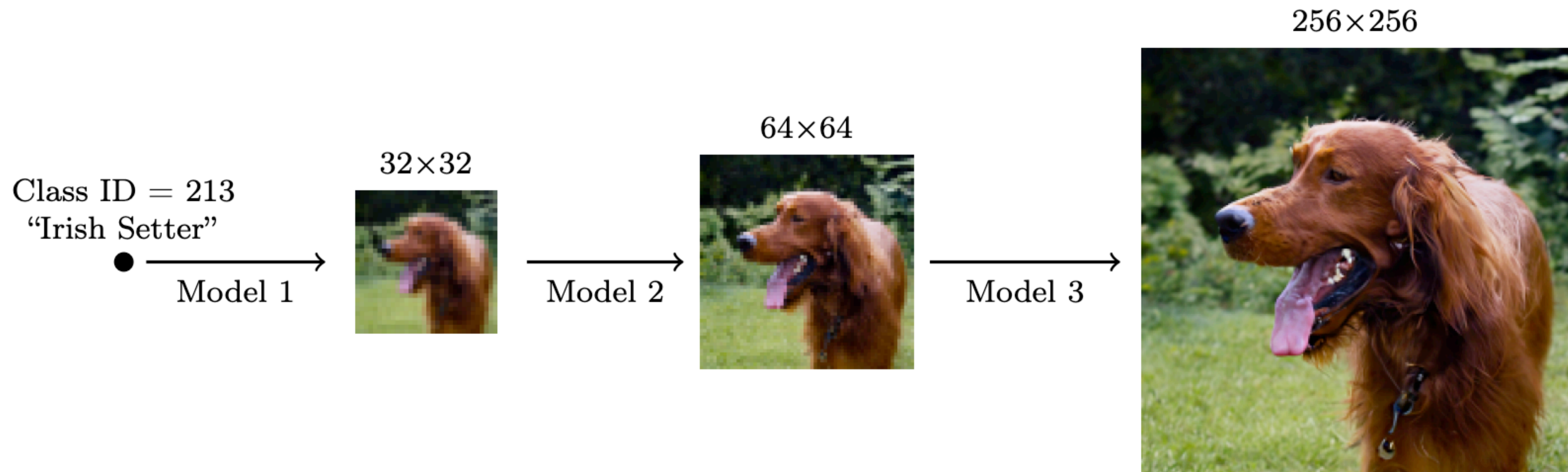
- CDMs used to increase resolution for 2D diffusion models
- Consist of multiple models (stages) stacked on top of each other





# Cascaded Diffusion Models (CDMs)

- CDMs used to increase resolution for 2D diffusion models
- Consist of multiple models (stages) stacked on top of each other
- These stages produce increasingly higher resolution outputs

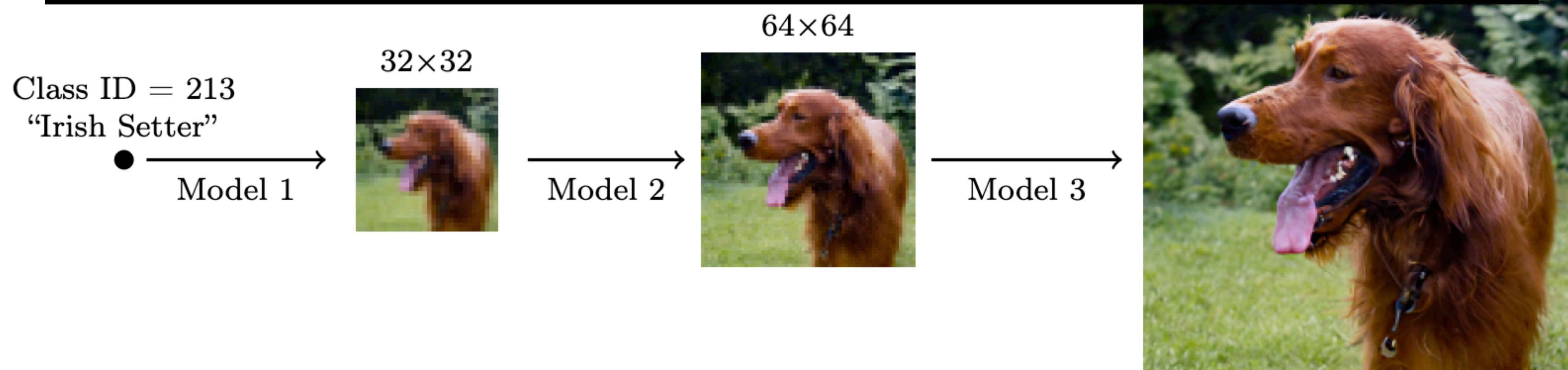




# Cascaded Diffusion Models (CDMs)

- CDMs used to increase resolution for 2D diffusion models
- Consist of multiple models (stages) stacked on top of each other
- These stages produce increasingly higher resolution outputs

**Higher res stages w/ distillation?**



# Multi-Resolution SDS

# Multi-Resolution SDS

- Standard SDS was built using CDMs, but only uses the first low resolution base stage!



# Multi-Resolution SDS

- Standard SDS was built using CDMs, but only uses the first low resolution base stage!
- Why not use the higher resolution stages as well?

# Multi-Resolution SDS

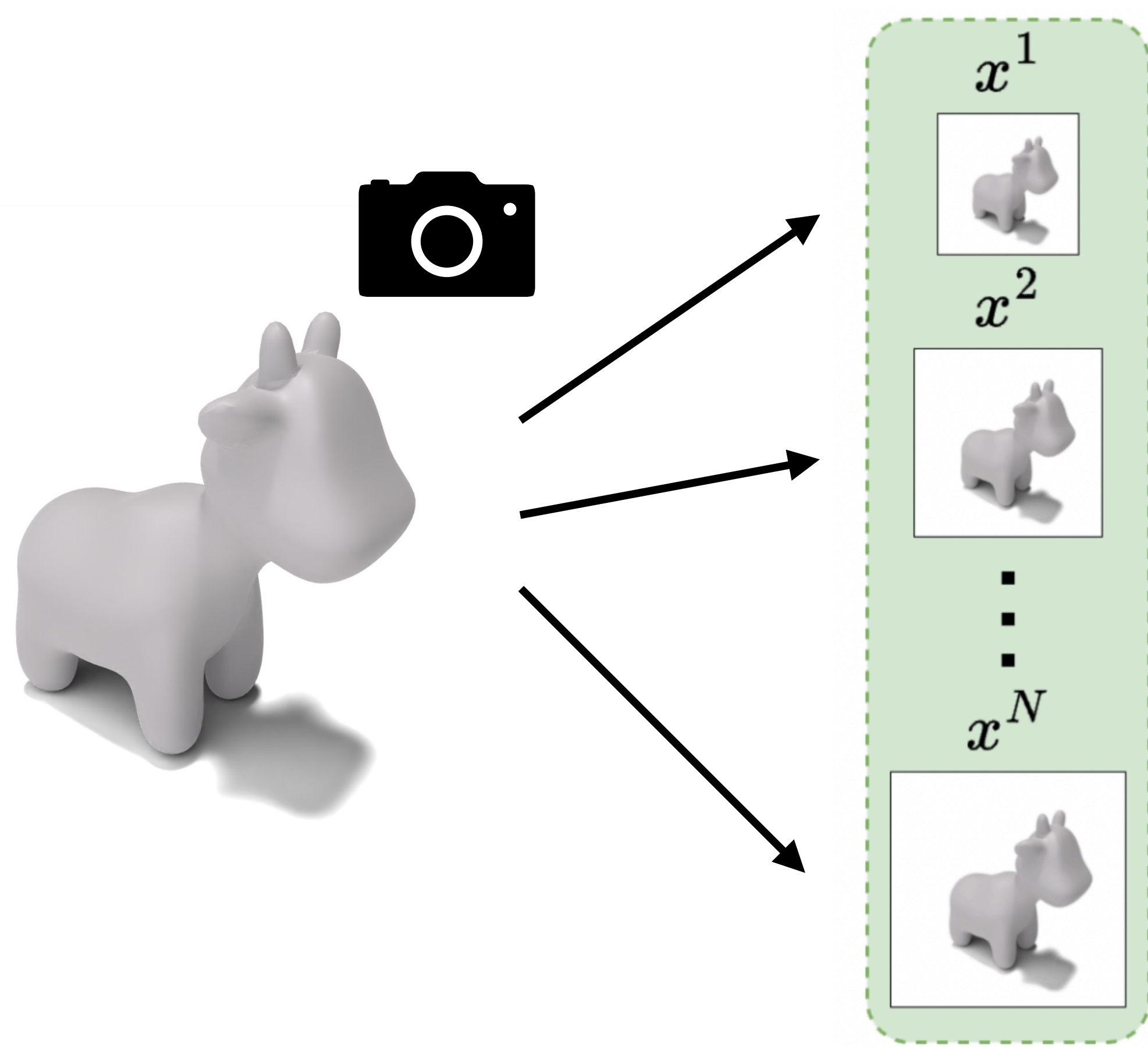
- Standard SDS was built using CDMs, but only uses the first low resolution base stage!
- Why not use the higher resolution stages as well?
- To do so, use Cascaded Score Distillation (CSD), **distill scores from the higher resolution stages** in addition to the base stage

# Cascaded Score Distillation

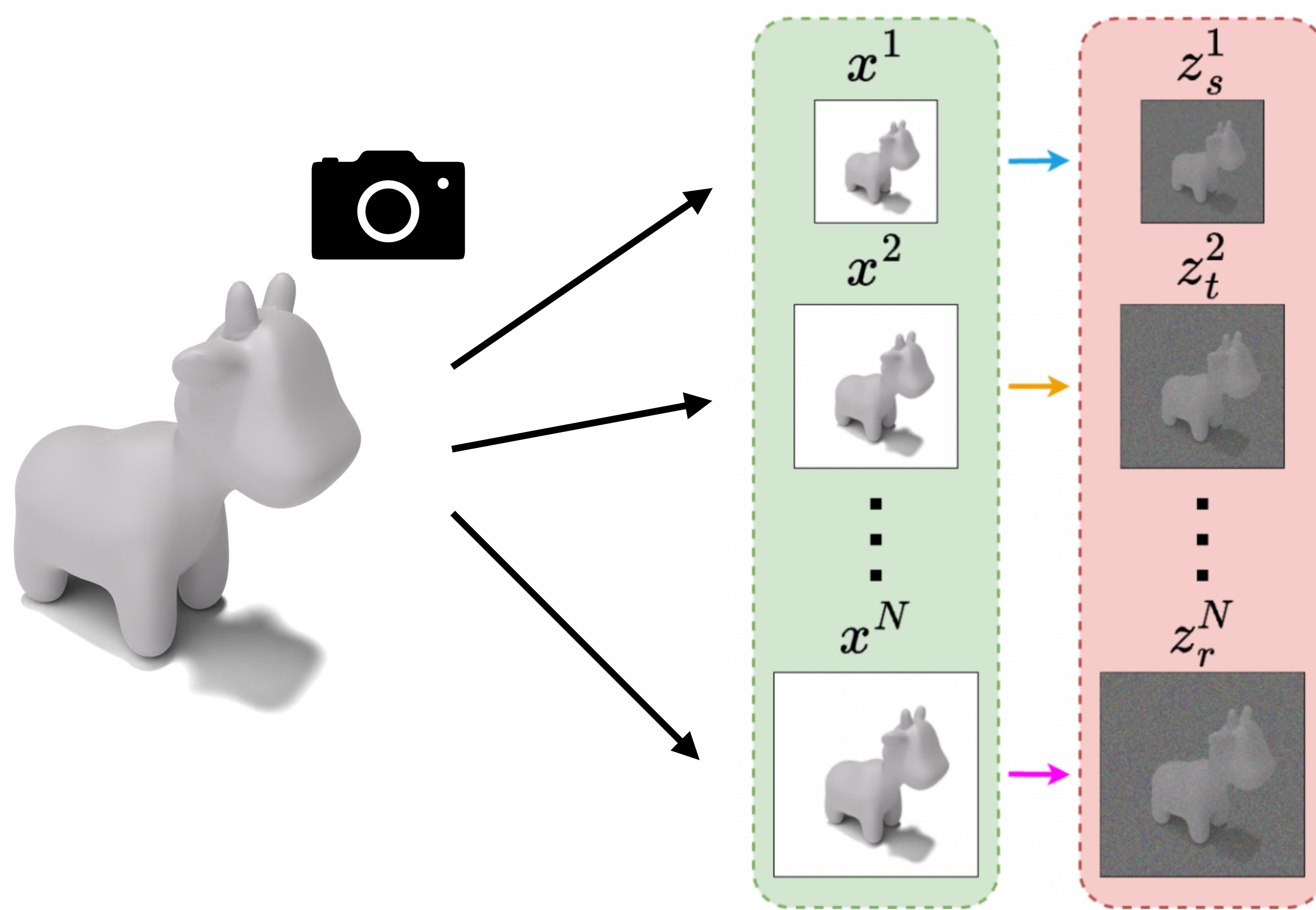




# Cascaded Score Distillation

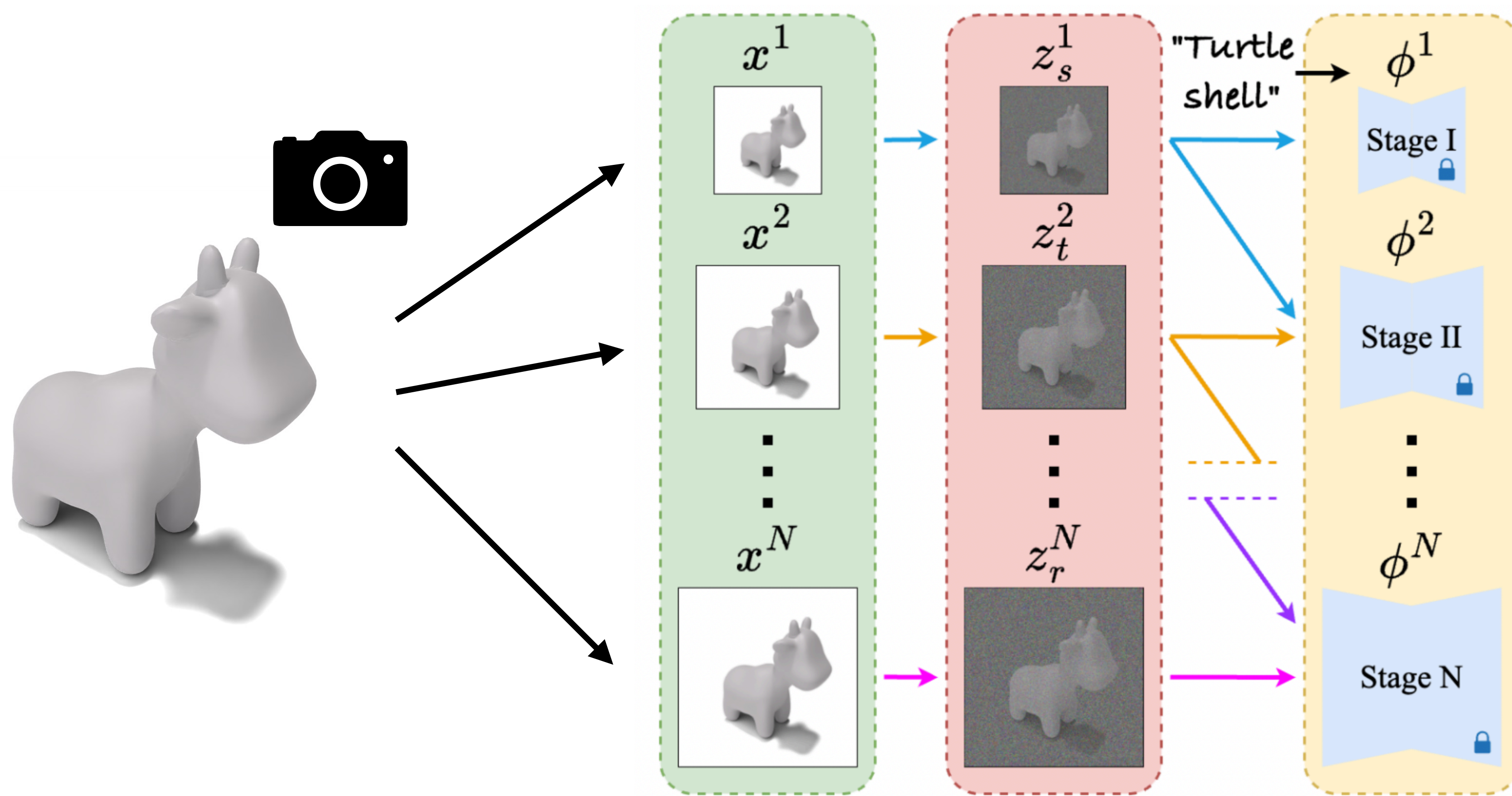


# Cascaded Score Distillation



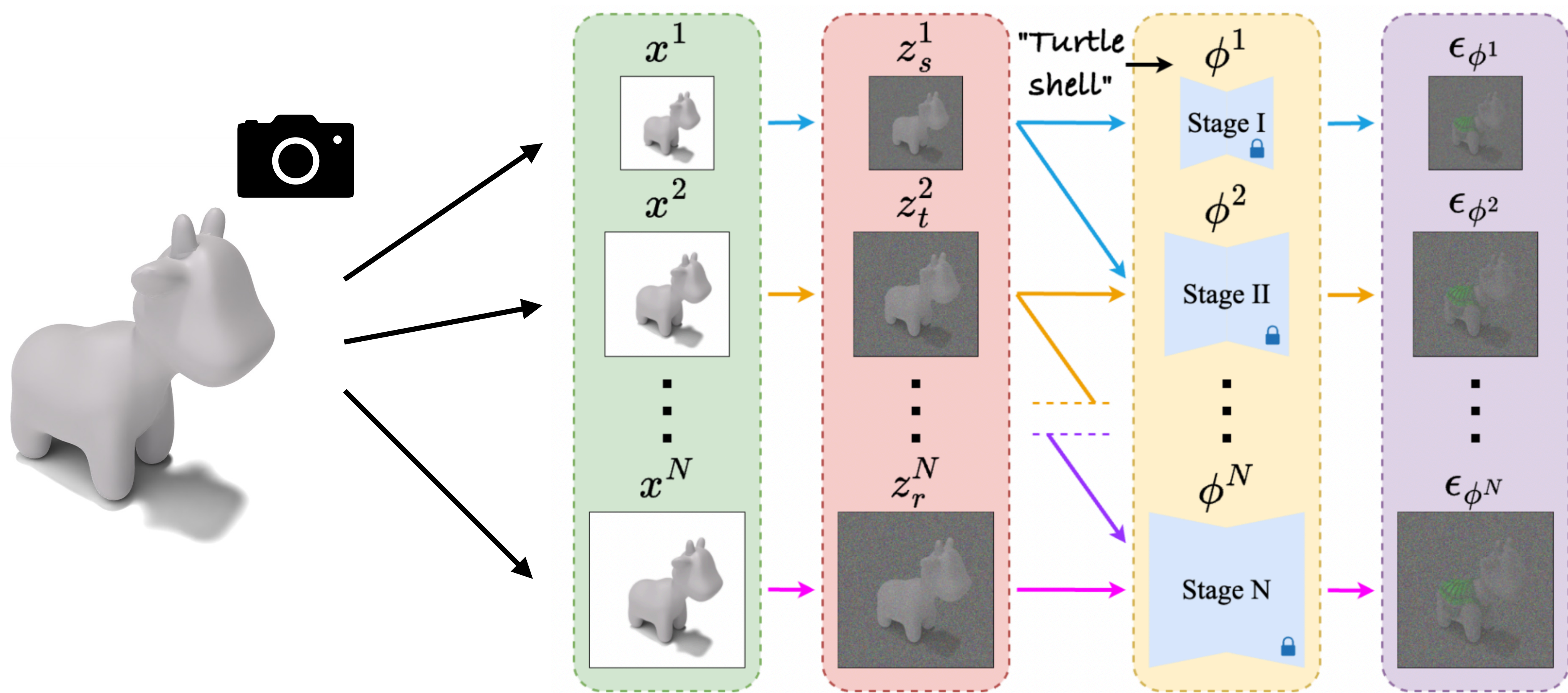


# Cascaded Score Distillation





# Cascaded Score Distillation



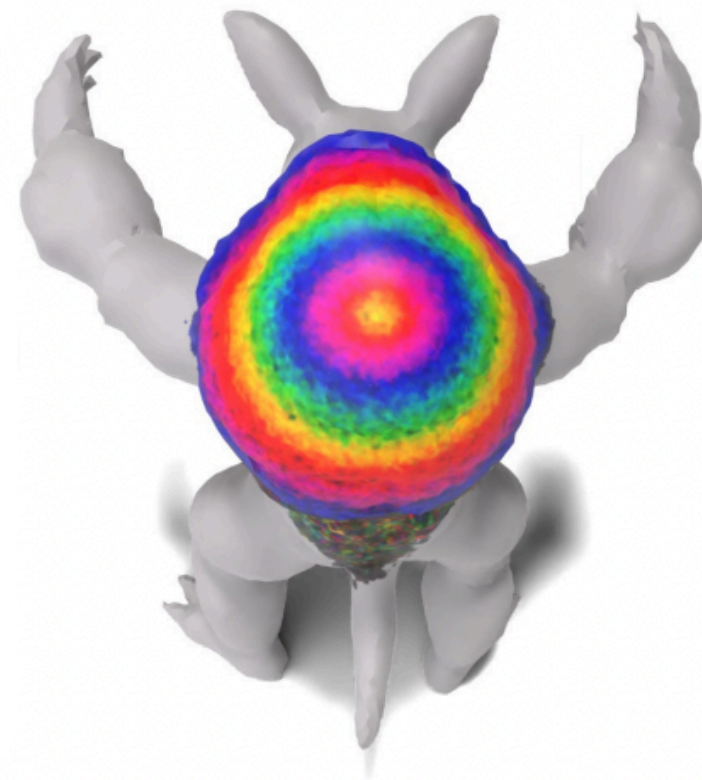
# CSD *vs* SDS



# CSD vs SDS

SDS

*“colorful crochet  
turtle shell”*



*“cactus base”*



*“tiger stripe shirt”*

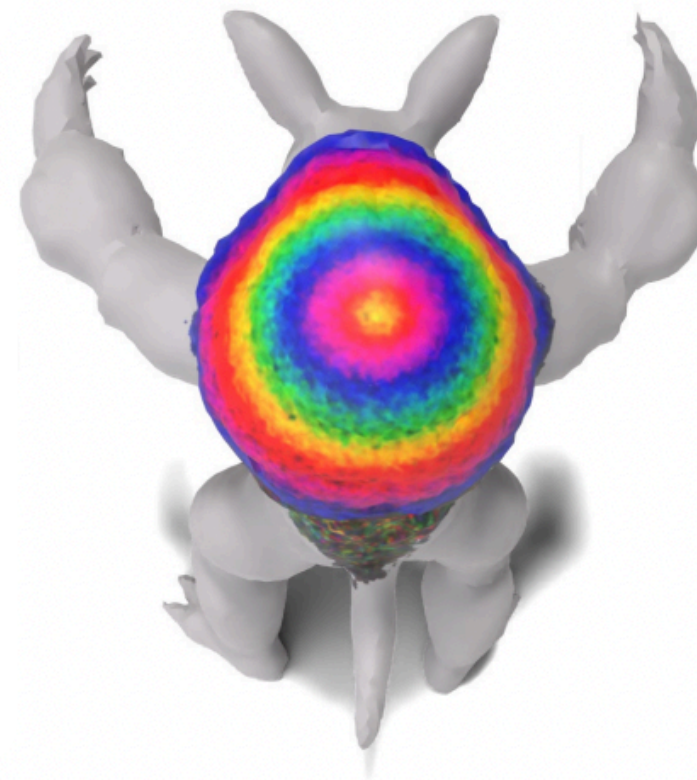




# CSD vs SDS

SDS

*“colorful crochet  
turtle shell”*



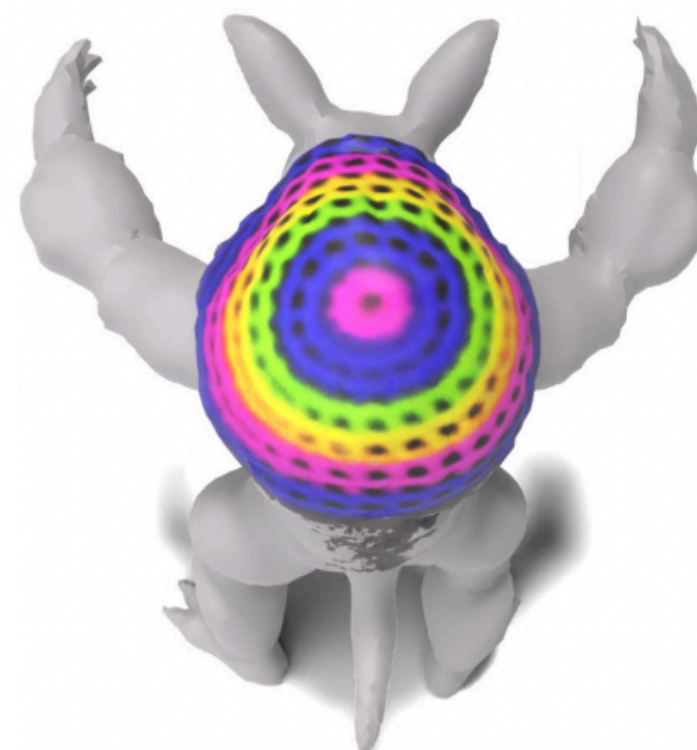
*“cactus base”*



*“tiger stripe shirt”*



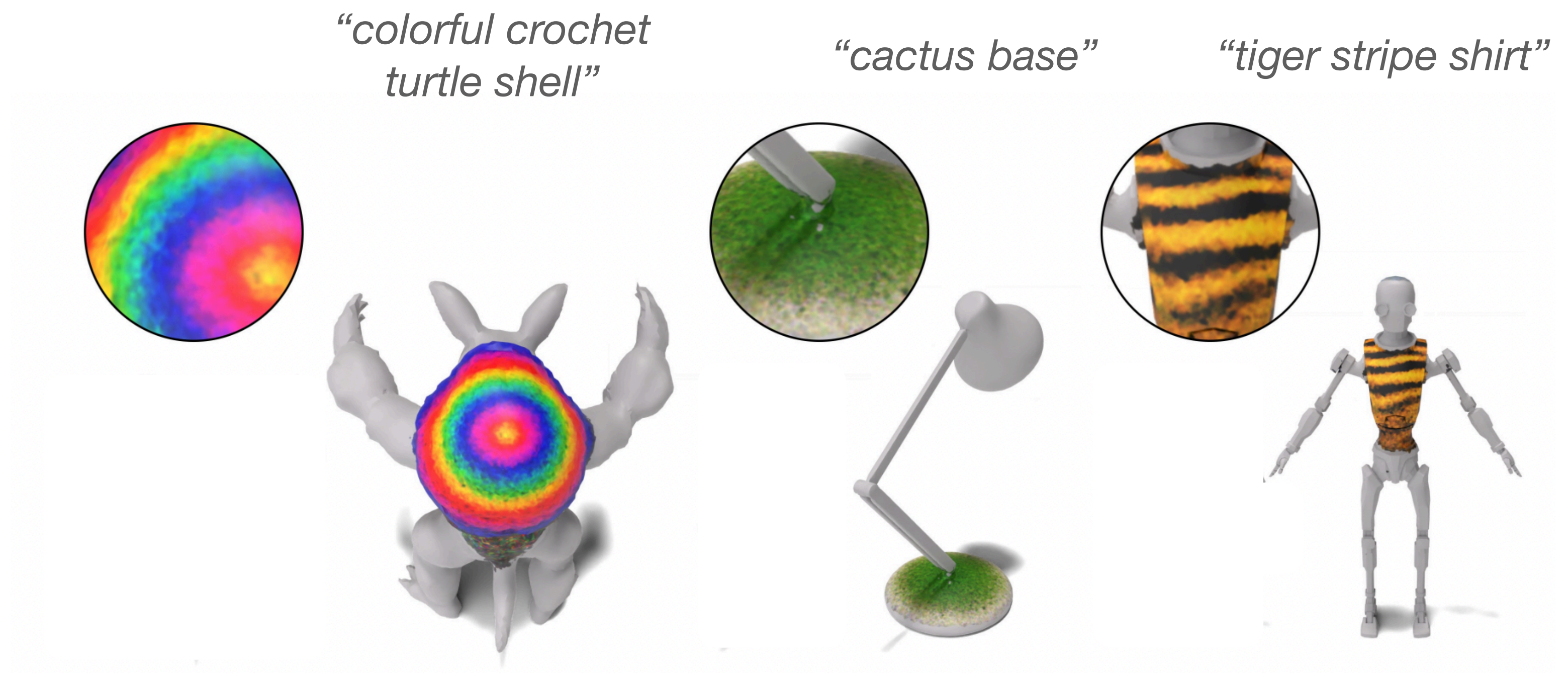
CSD



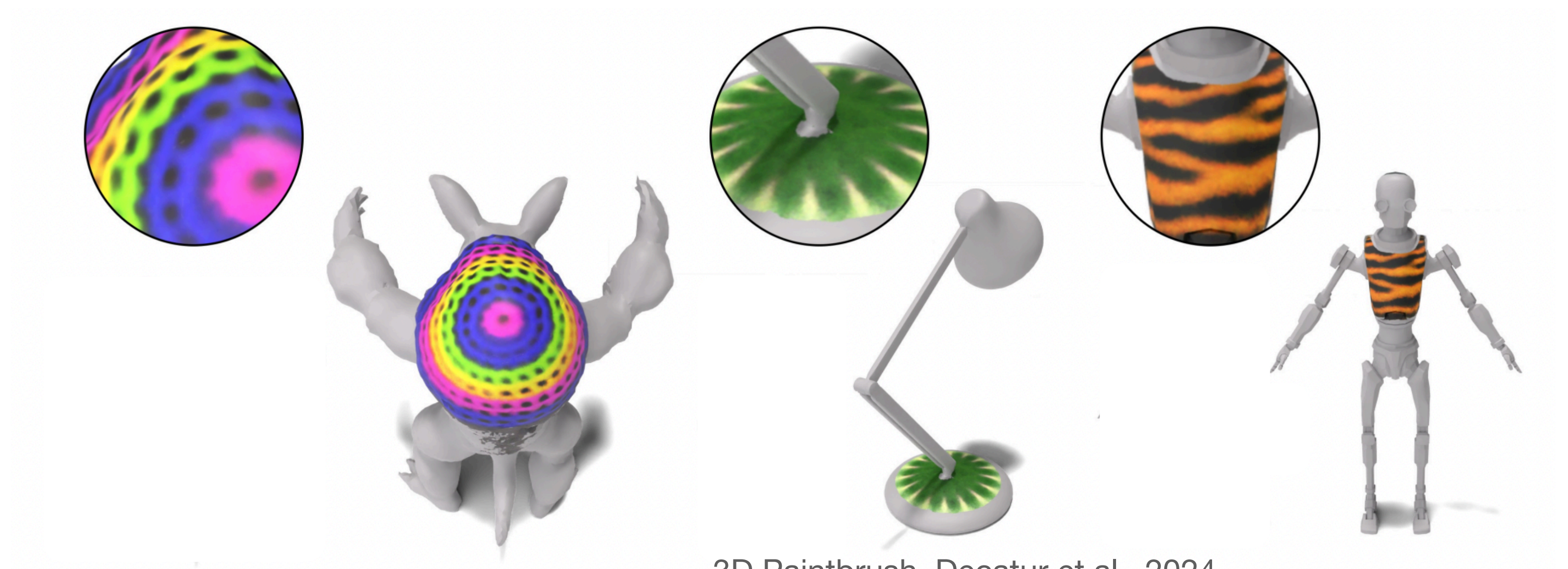


# CSD vs SDS

SDS



CSD

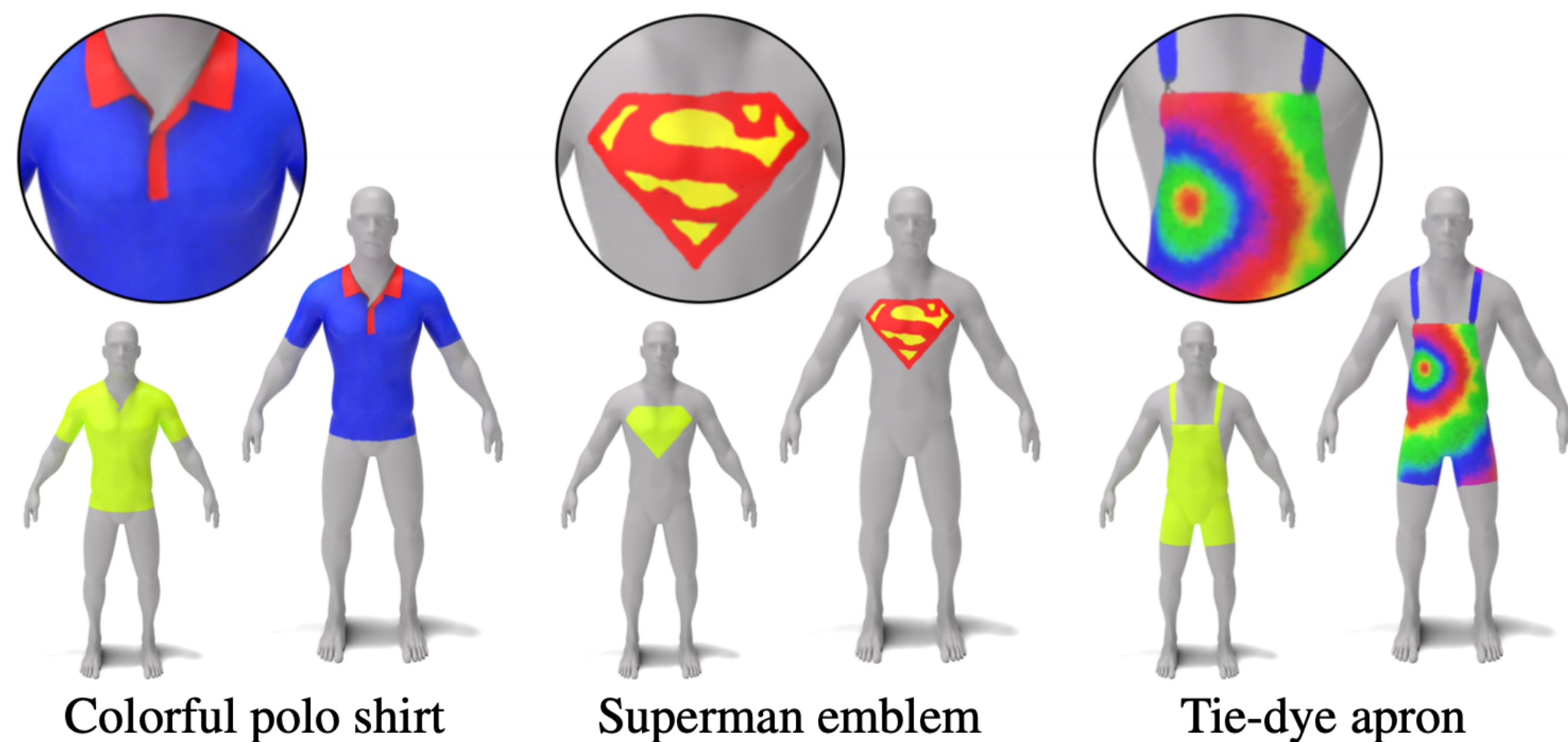




# Applications

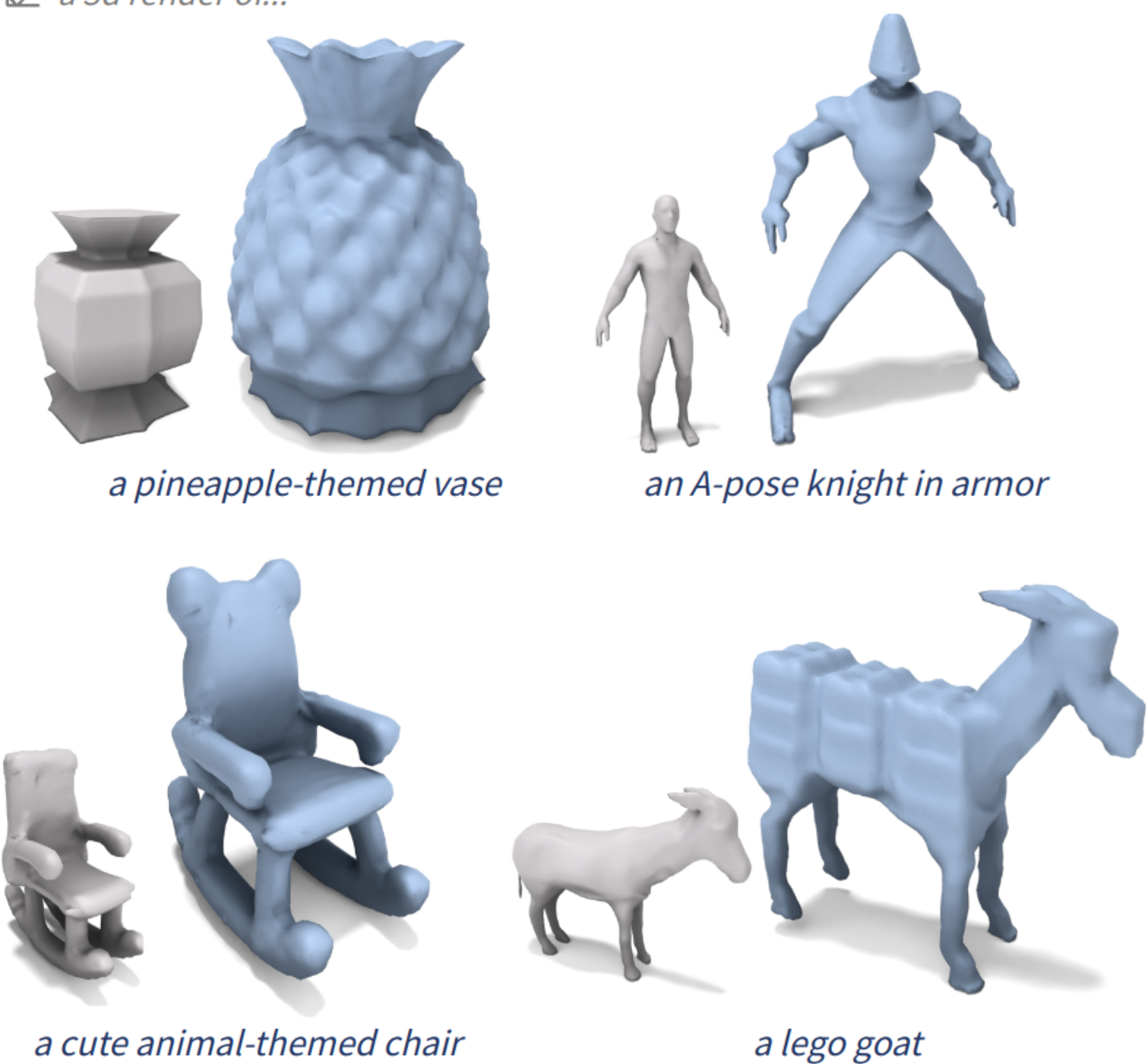
*Theme: “task-specific structure to rein in noisy supervision signal”*

## 3D Paintbrush Decatur et al. 2024



## Geometry in Style Dinh et al. 2025

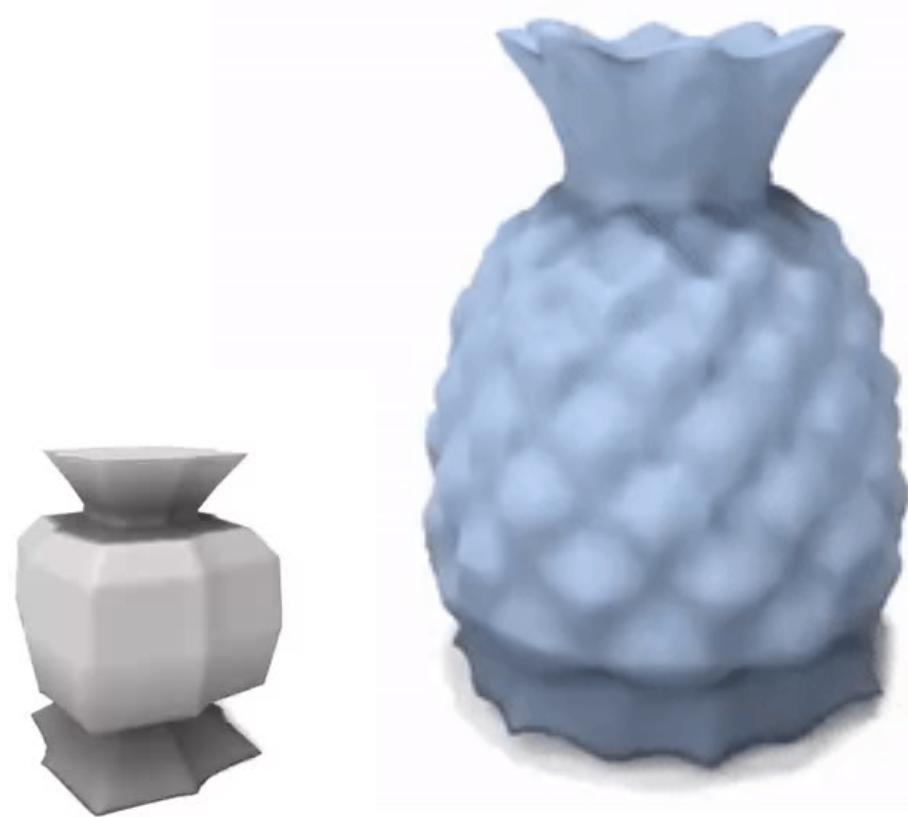
 a 3d render of...



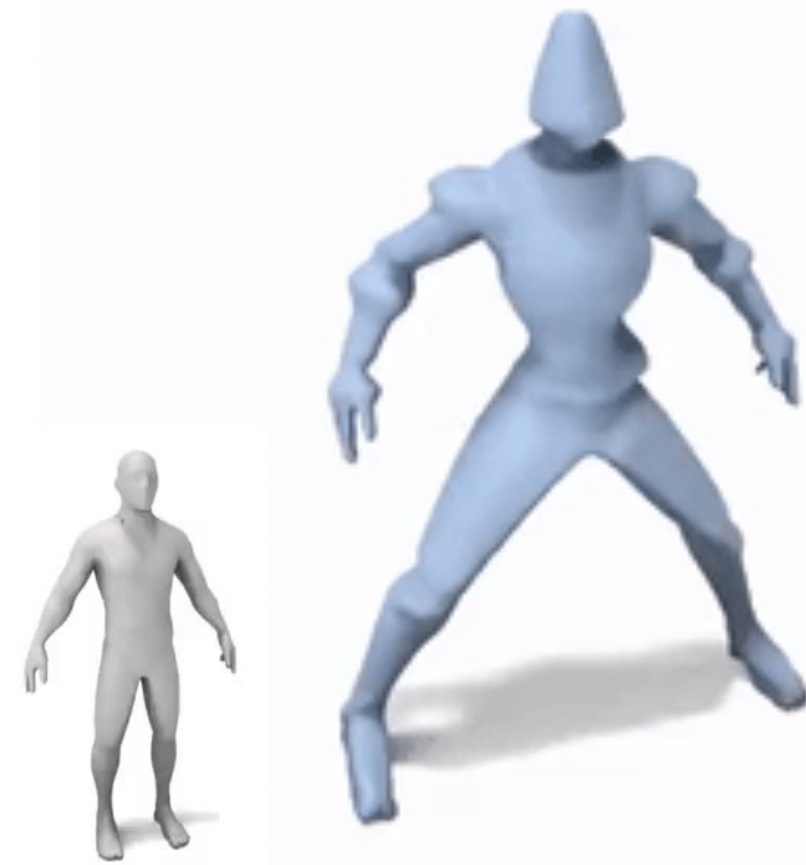


# Application: *Geometry in Style*

 a 3d render of...



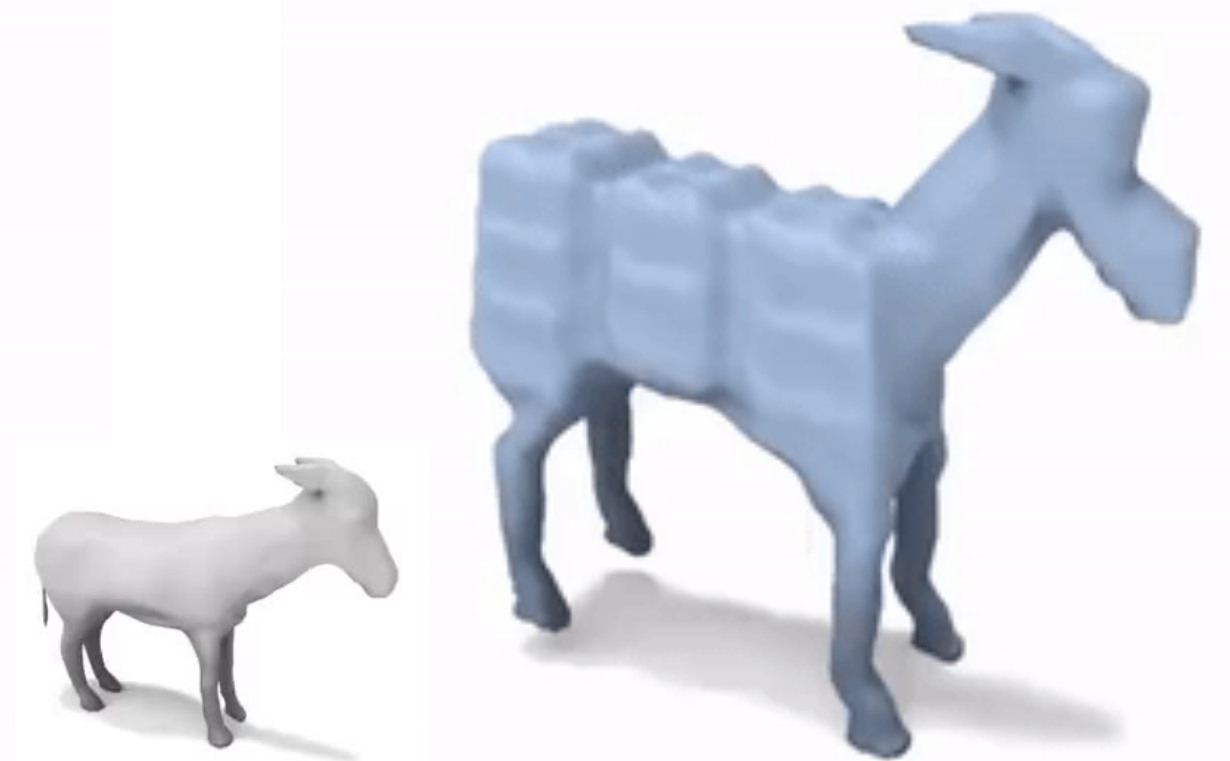
*a pineapple-themed  
vase*



*an A-pose knight  
in armor*



*a cute animal-themed chair*

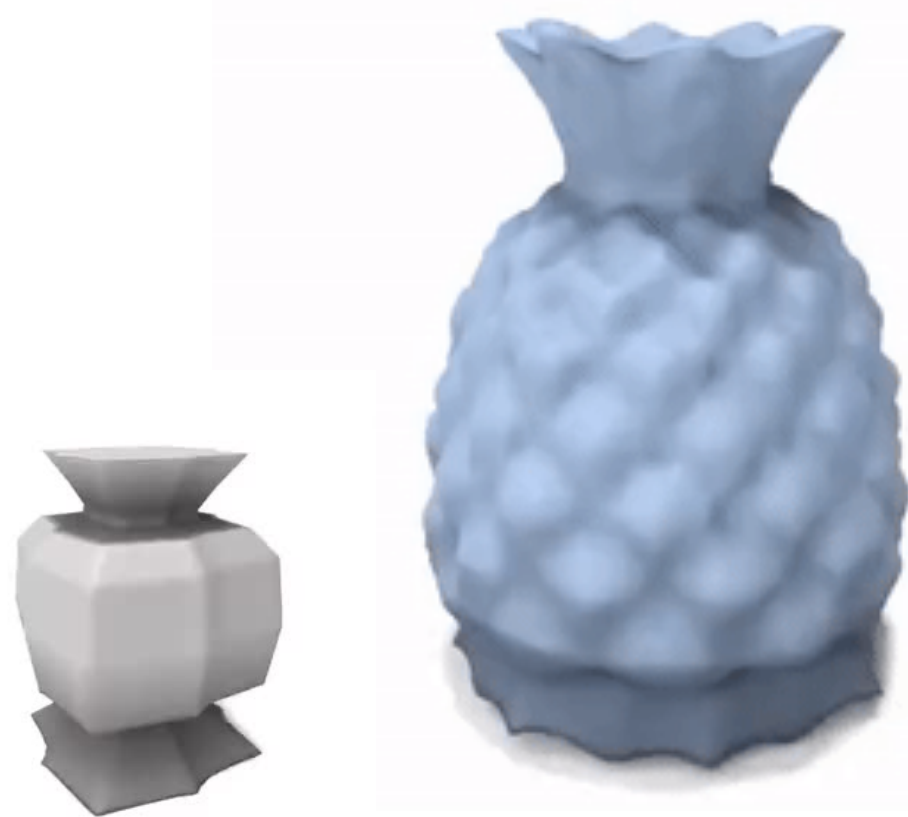


*a lego goat*

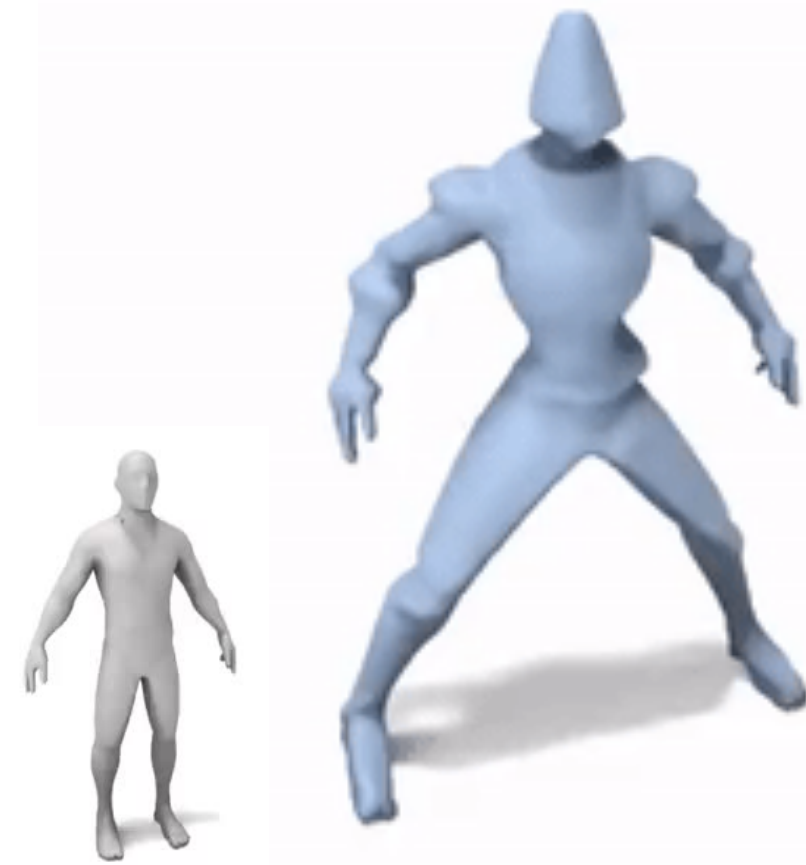
**Geometry in Style**  
Dinh et al. 2025

# Application: *Geometry in Style*

 a 3d render of...



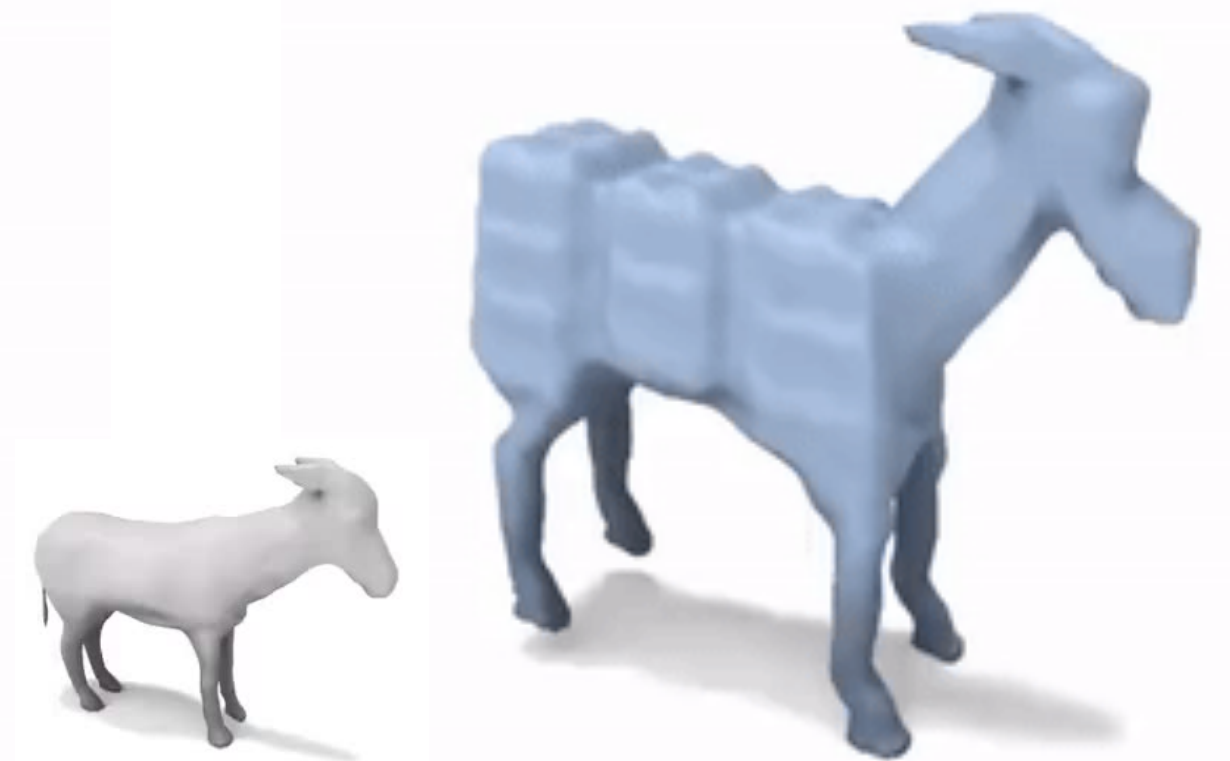
*a pineapple-themed  
vase*



*an A-pose knight  
in armor*



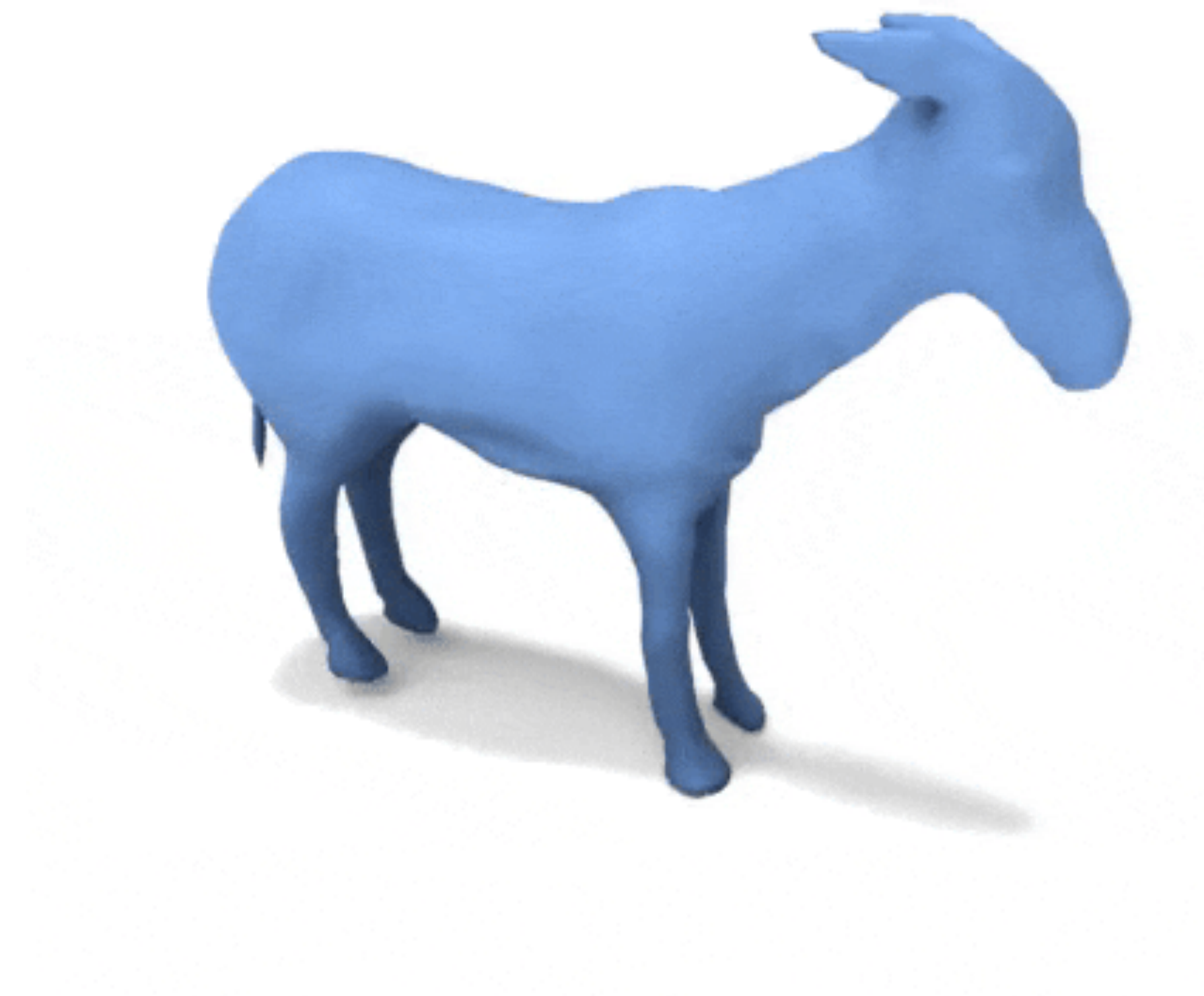
*a cute animal-themed chair*



*a lego goat*

**Geometry in Style**  
Dinh et al. 2025

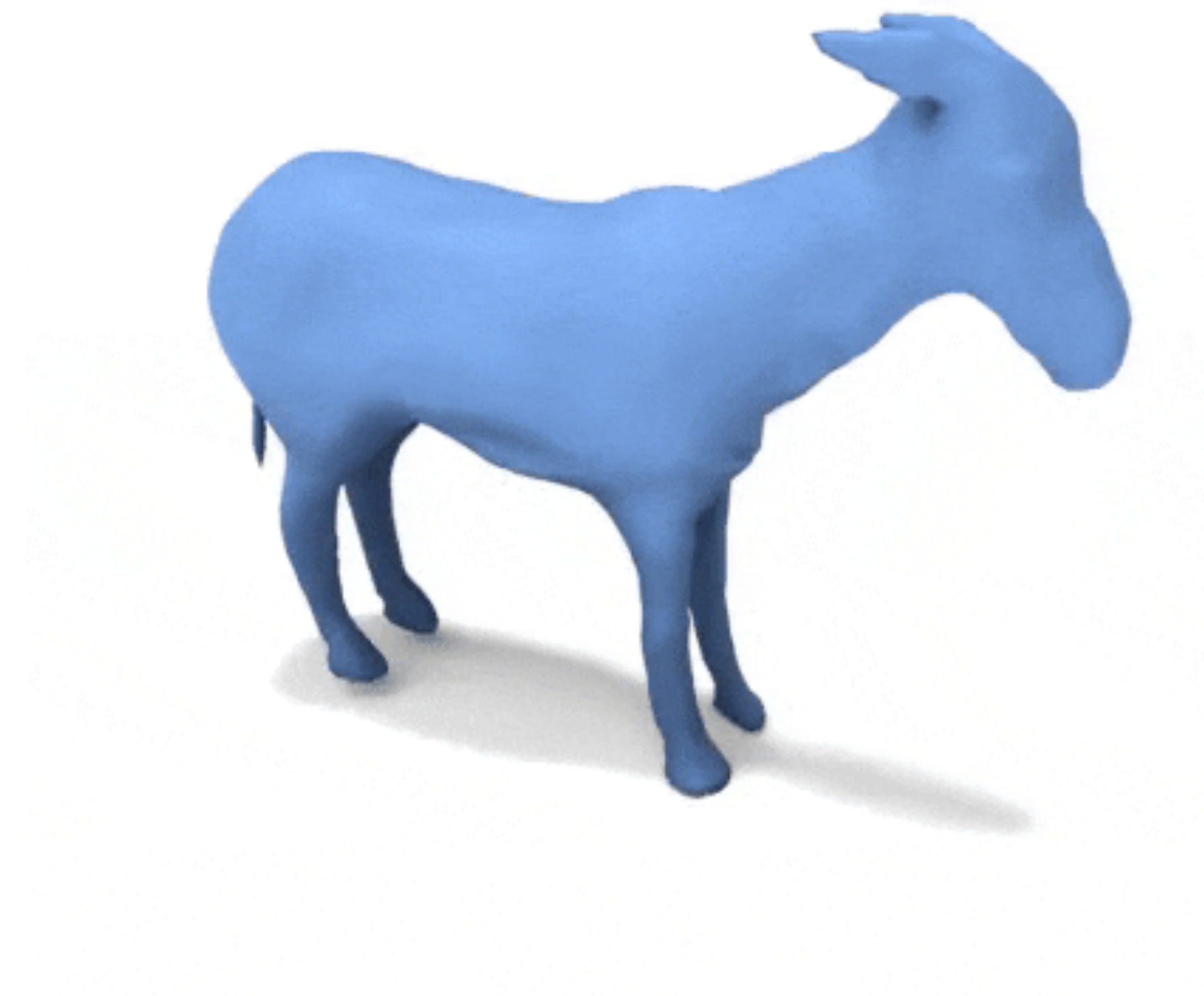
# Identity-preserving geometric stylization



*“a 3d render of a lego goat”*



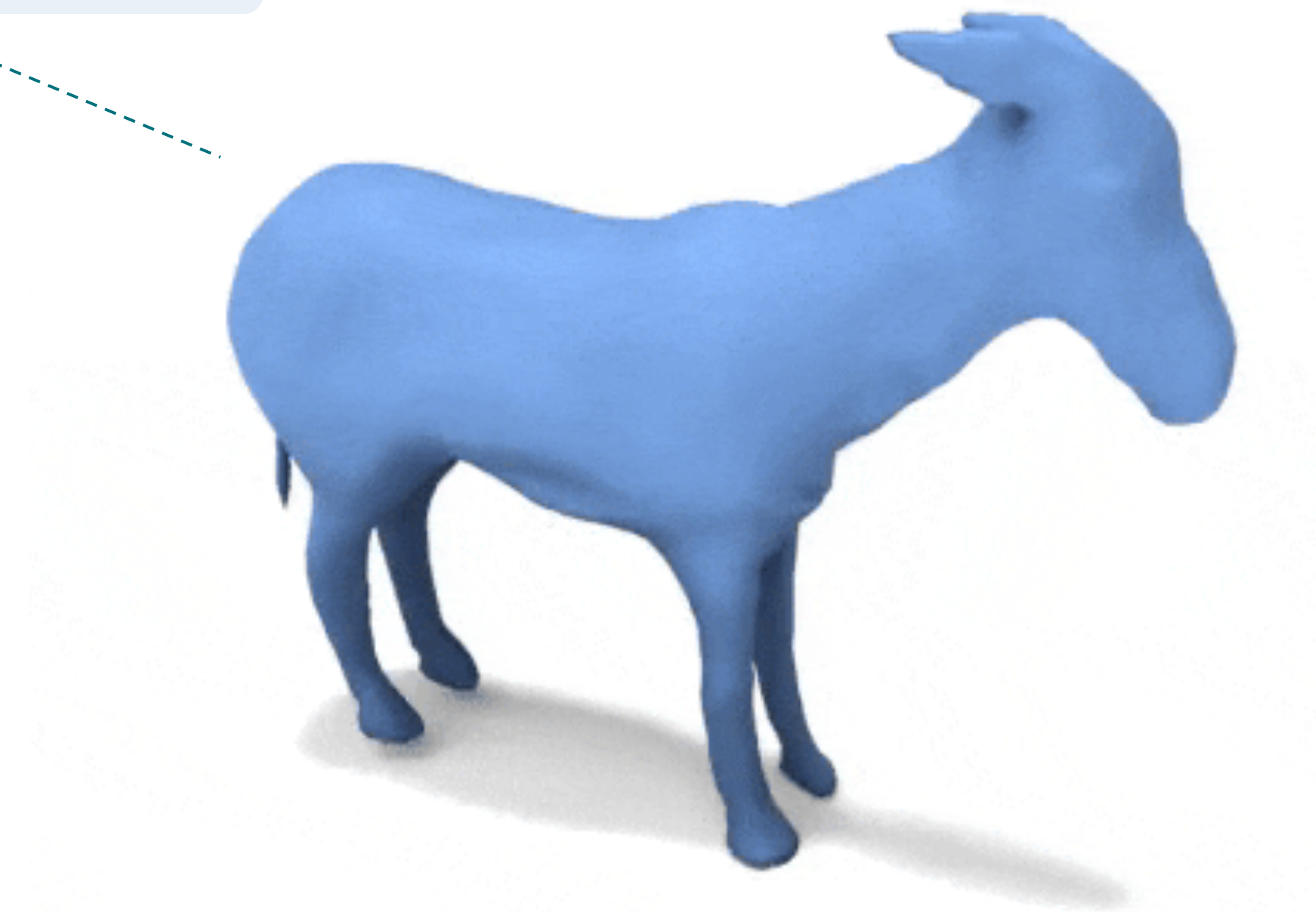
# Identity-preserving geometric stylization



*“a 3d render of a lego goat”*

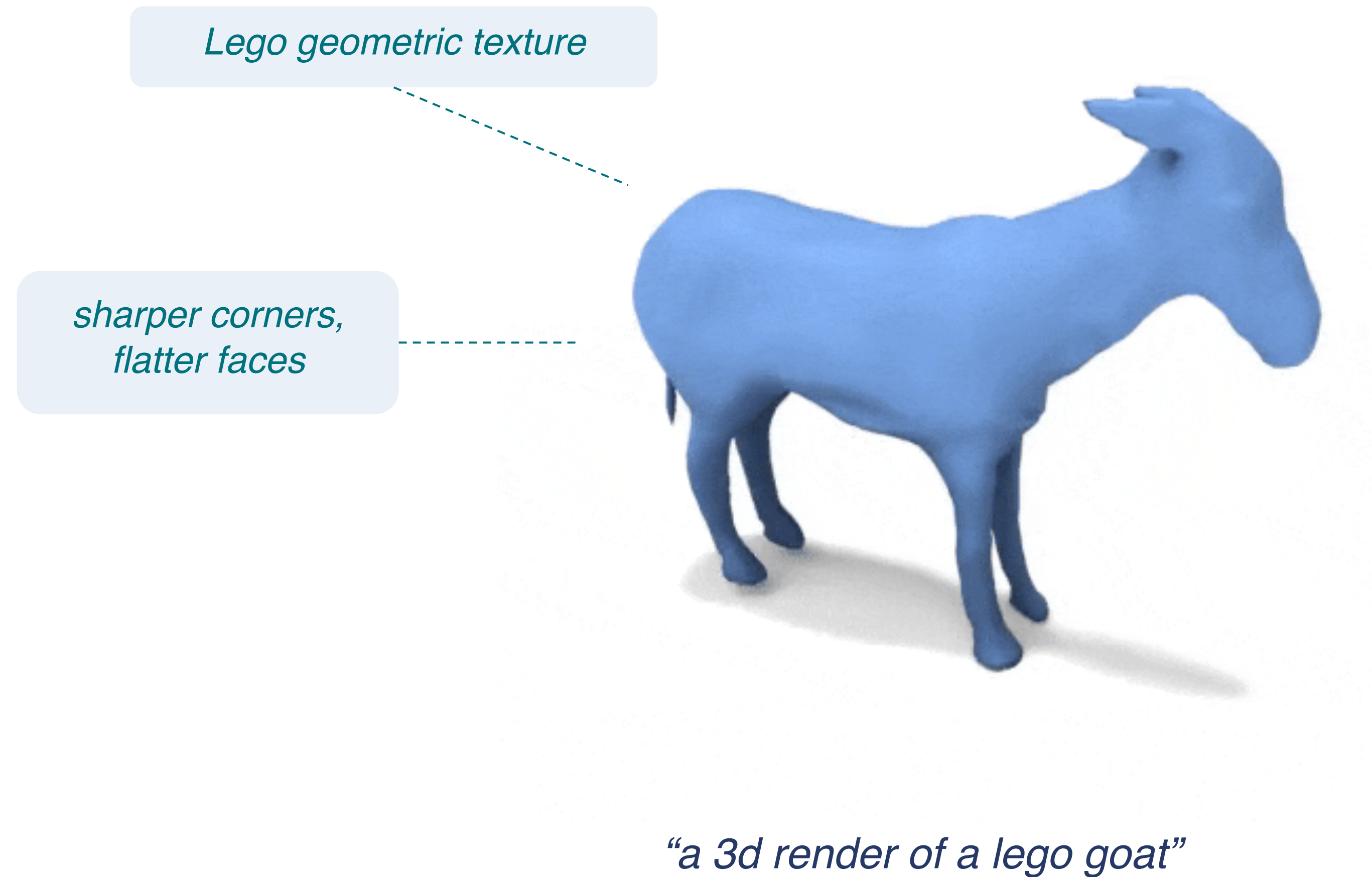
# Identity-preserving geometric stylization

*Lego geometric texture*



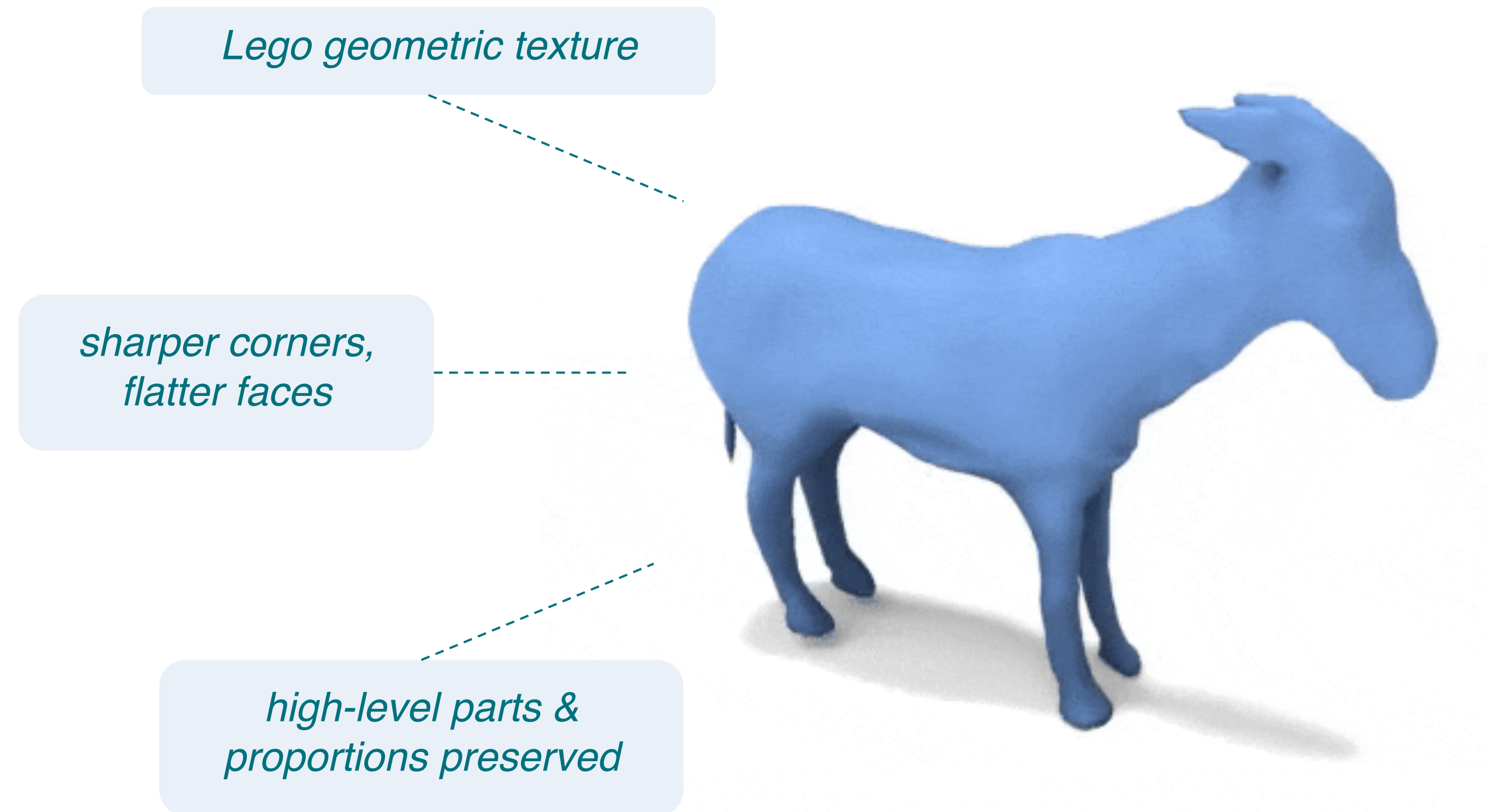
*“a 3d render of a lego goat”*

# Identity-preserving geometric stylization



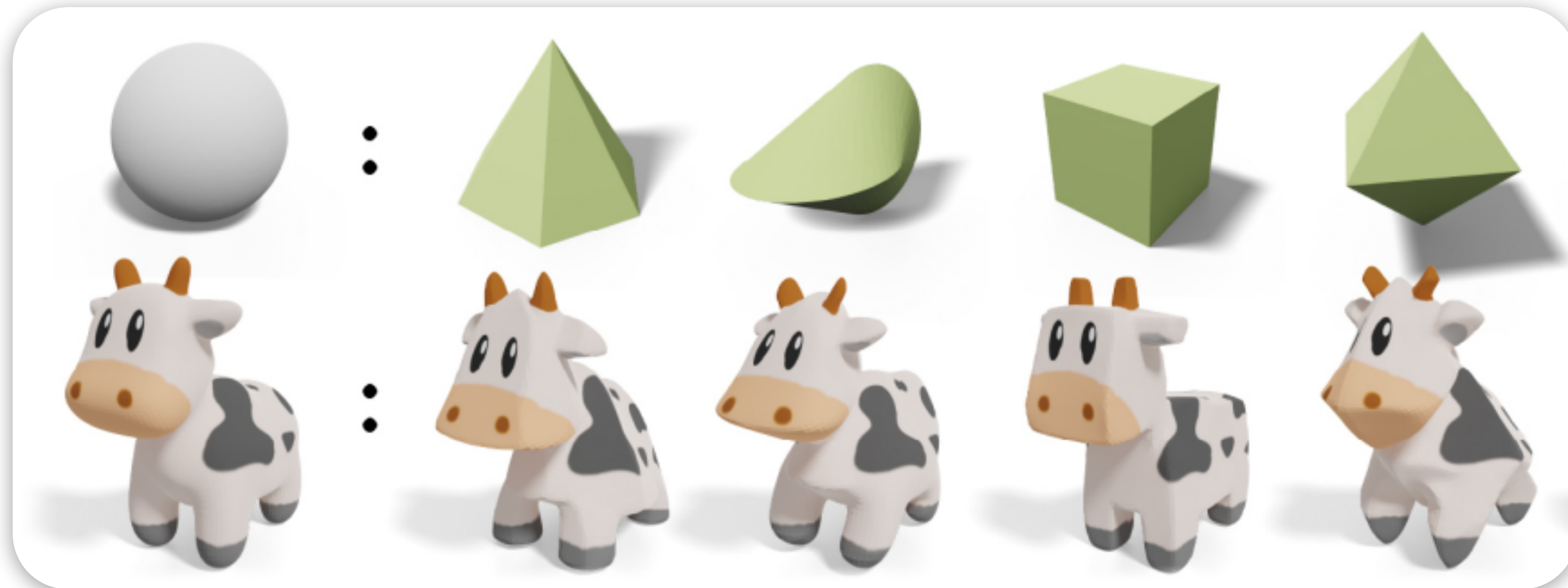


# Identity-preserving geometric stylization



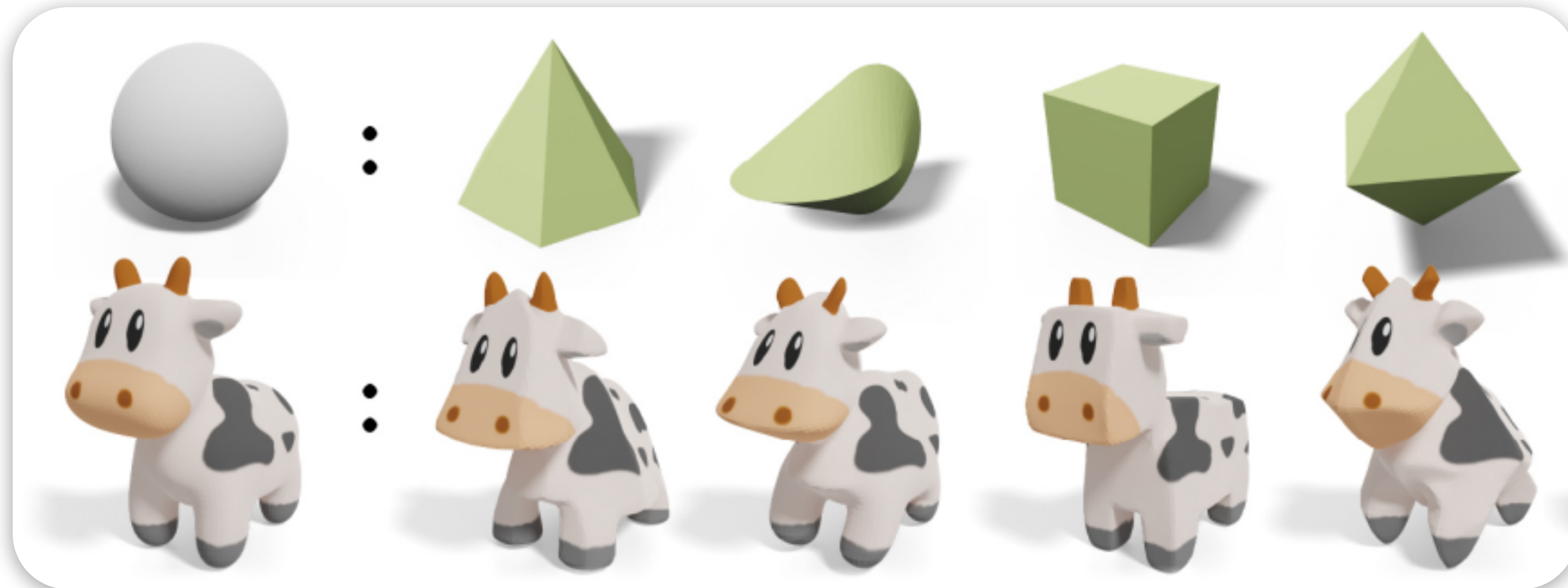
*“a 3d render of a lego goat”*

# Normals as style



Normal-Driven Spherical Shape Analogies, Liu & Jacobson 2021

# Normals as style

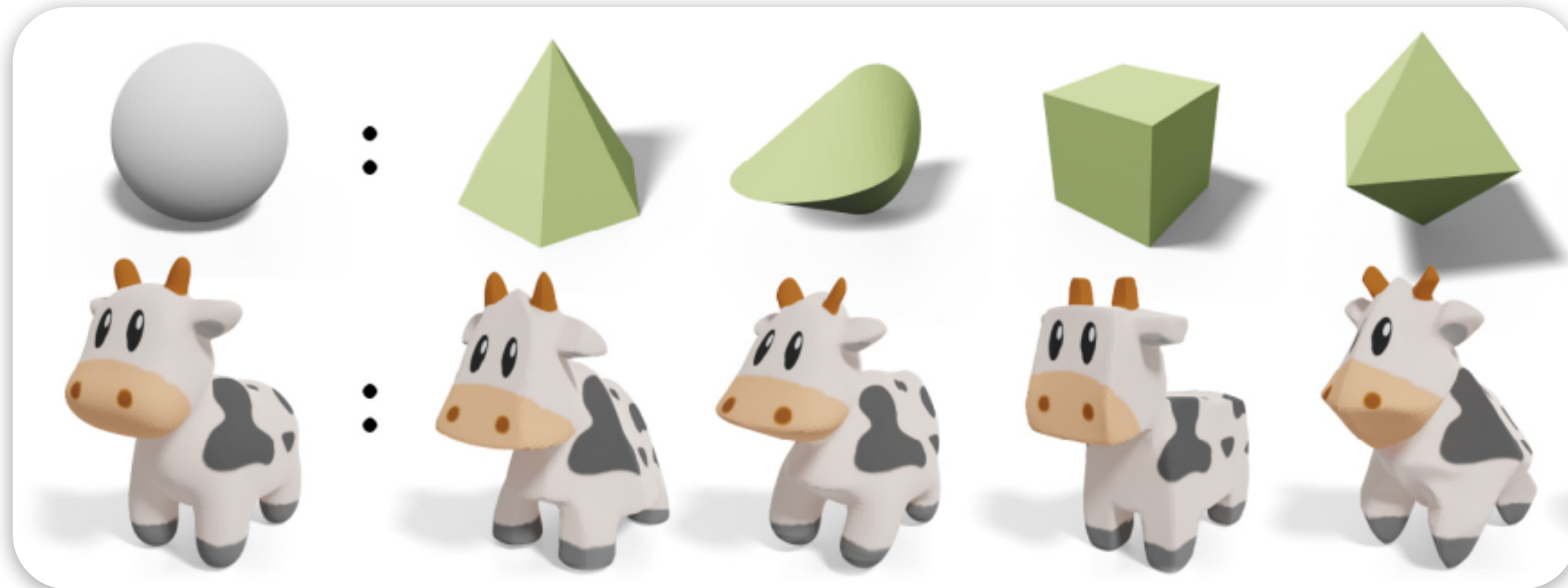


Normal-Driven Spherical Shape Analogies, Liu & Jacobson 2021

*Preserves identity, but  
limited to simple templates*



# Normals as style

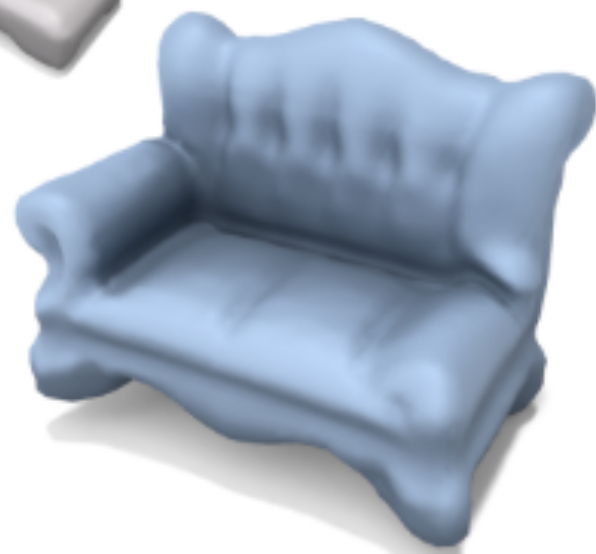


Normal-Driven Spherical Shape Analogies, Liu & Jacobson 2021

*Preserves identity, but  
limited to simple templates*

*What if we learned or optimized normals  
on the shape to be stylized...  
to get any desired style?*

 a 3d render of a(n)...



*antique sofa*



*tropical chair*



*racing chair*



*greek statue chair*



*gothic chair*



*cardboard chair*



*cybernetic glove*



*skull*



*dinosaur statue*



*ornate  
art deco column*



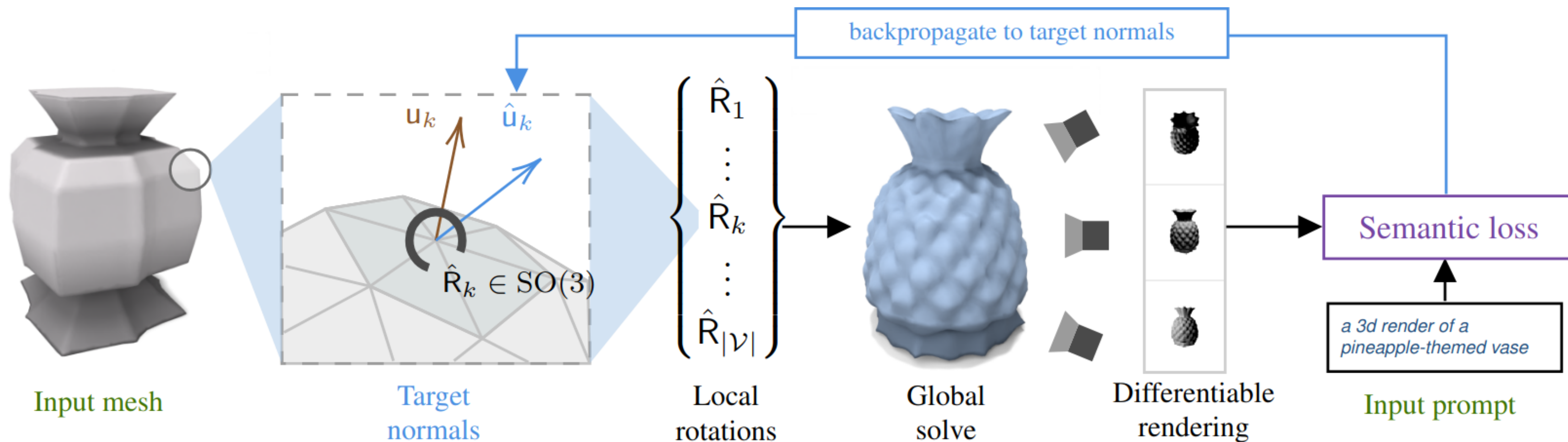
*penguin-themed  
fire hydrant*



*racer bunny*

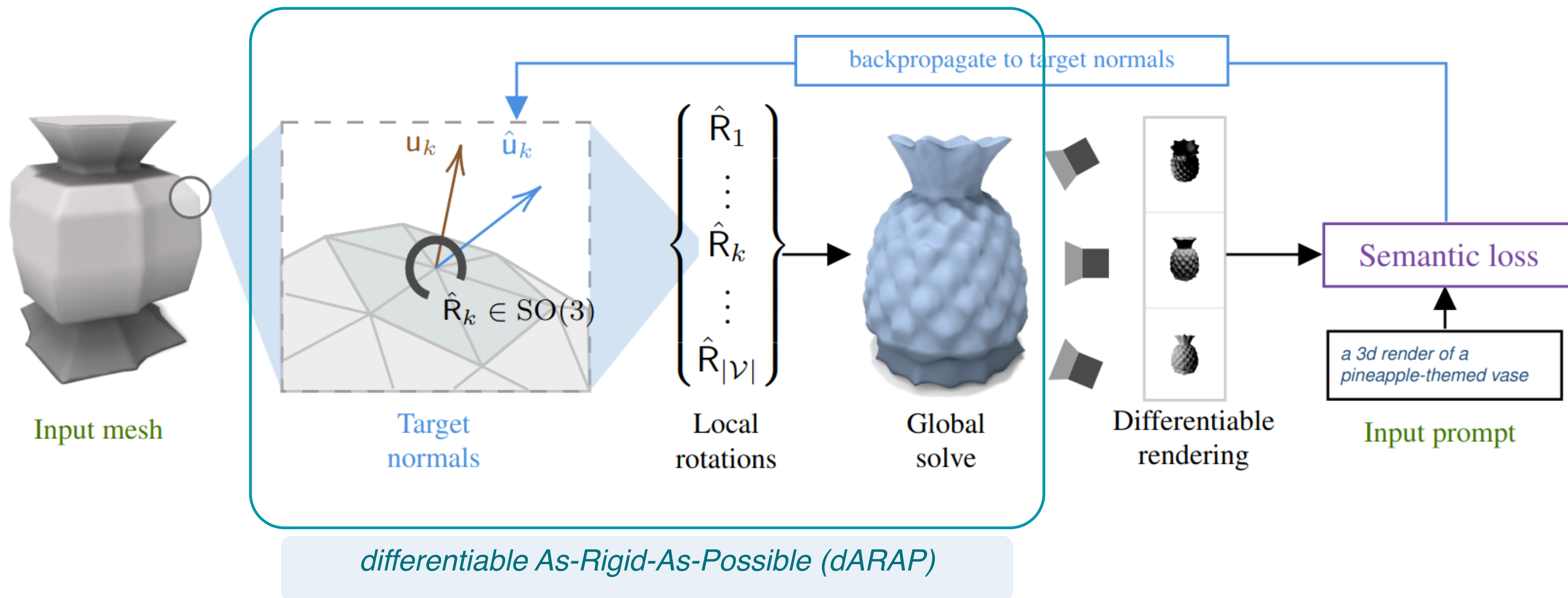


# Geometry in Style: pipeline

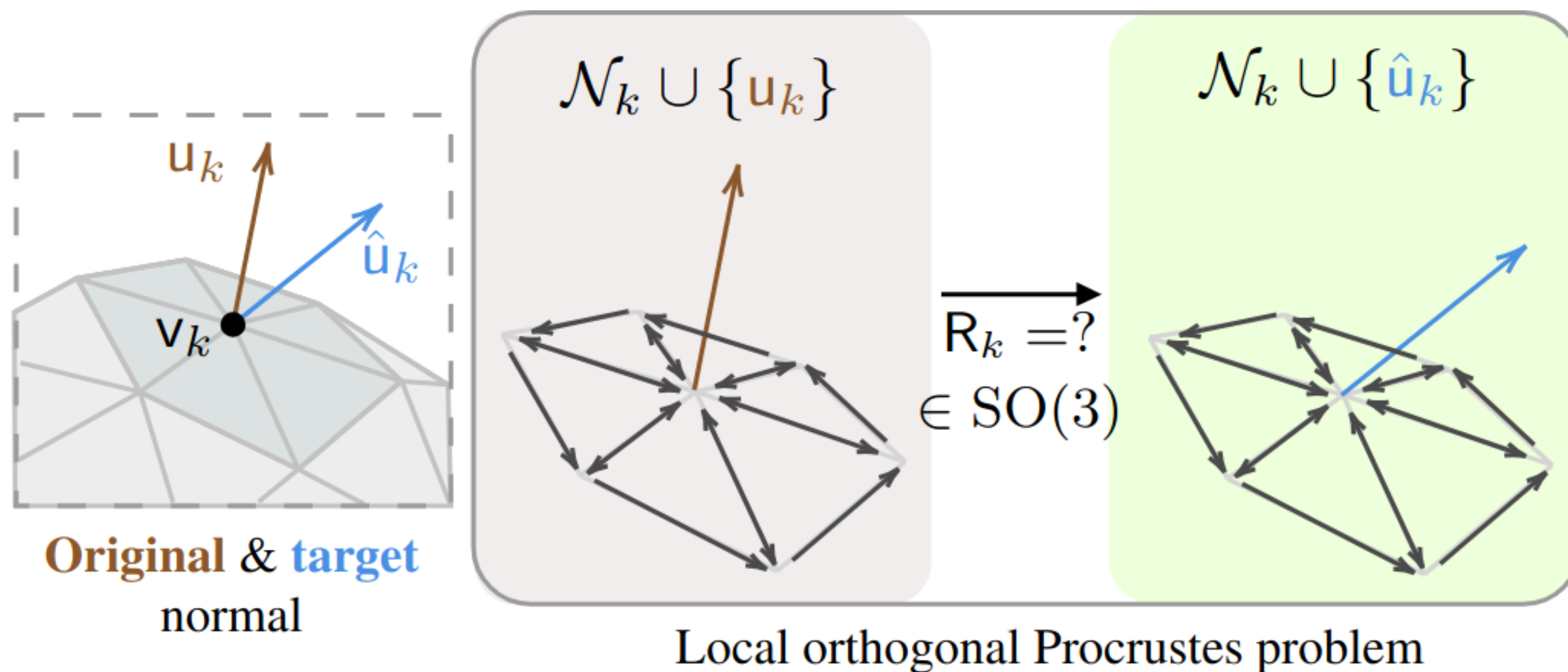




# Geometry in Style: pipeline

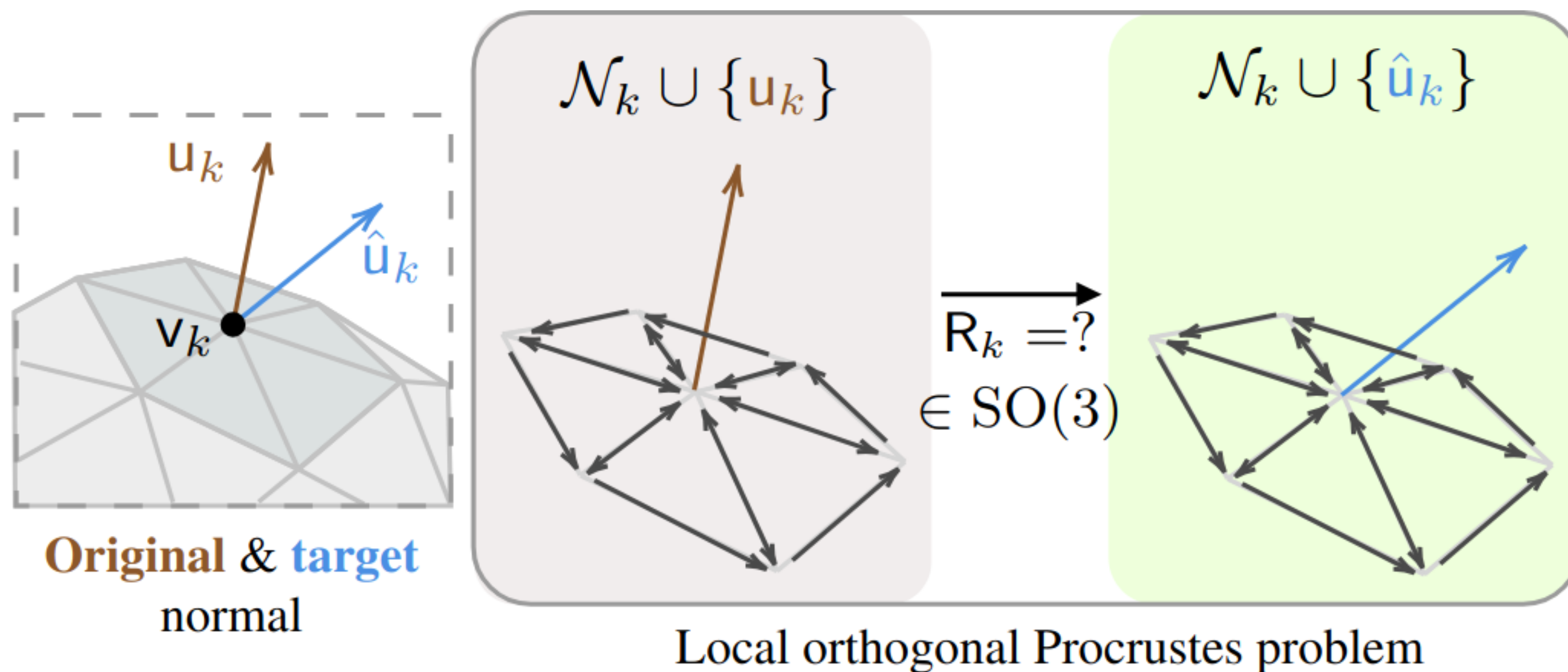


# dARAP: Local step & Normals



$$\hat{R}_k = \underset{R_k}{\operatorname{argmin}} \sum_{(i,j) \in \mathcal{N}_k} w_{ij} \|R_k \mathbf{e}_{ij} - \mathbf{e}_{ij}\|_2^2 + \lambda a_k \|R_k \mathbf{u}_k - \hat{\mathbf{u}}_k\|_2^2$$

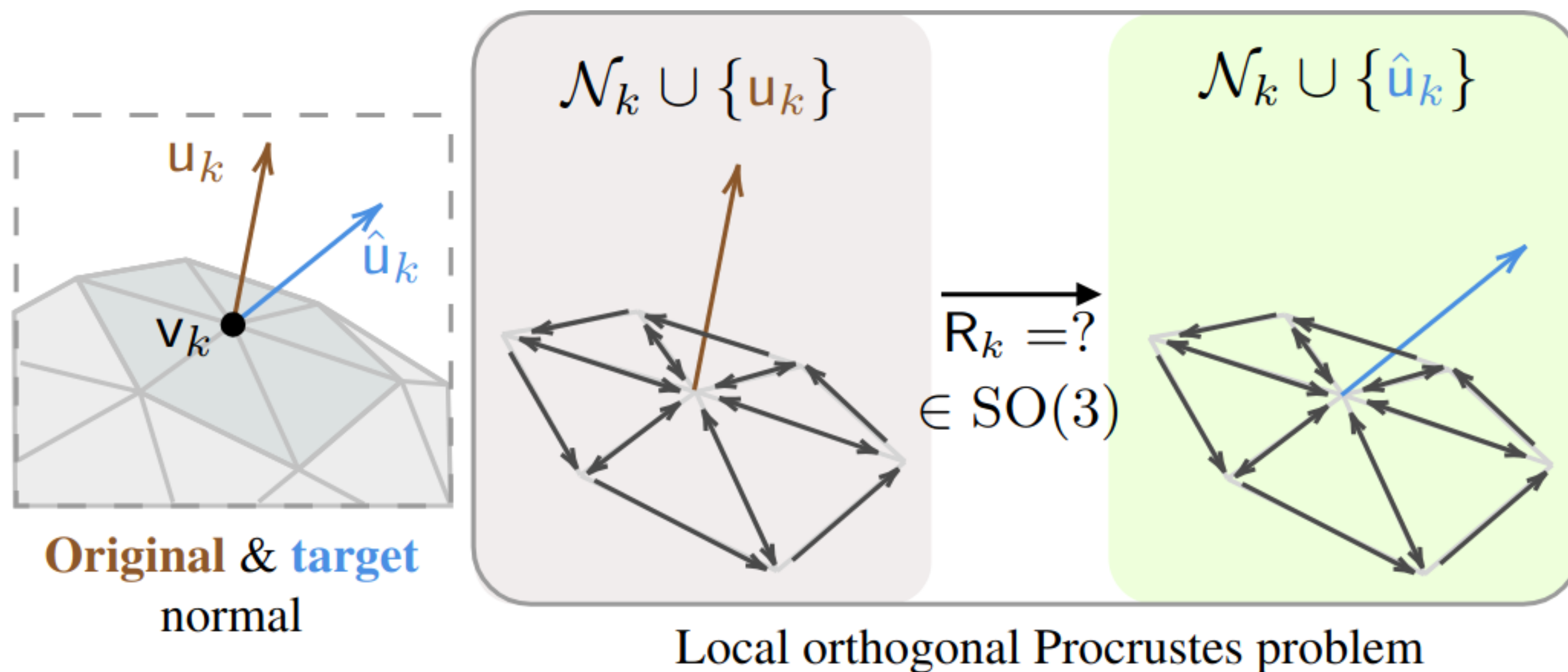
# dARAP: Local step & Normals



$$\hat{R}_k = \underset{R_k}{\operatorname{argmin}} \sum_{(i,j) \in \mathcal{N}_k} w_{ij} \|\underline{R_k \mathbf{e}_{ij}} - \mathbf{e}_{ij}\|_2^2 + \lambda a_k \|R_k \mathbf{u}_k - \hat{\mathbf{u}}_k\|_2^2$$



# dARAP: Local step & Normals



$$\hat{R}_k = \operatorname{argmin}_{R_k} \sum_{(i,j) \in \mathcal{N}_k} w_{ij} \|R_k \mathbf{e}_{ij} - \mathbf{e}_{ij}\|_2^2 + \lambda a_k \|R_k \mathbf{u}_k - \hat{\mathbf{u}}_k\|_2^2$$

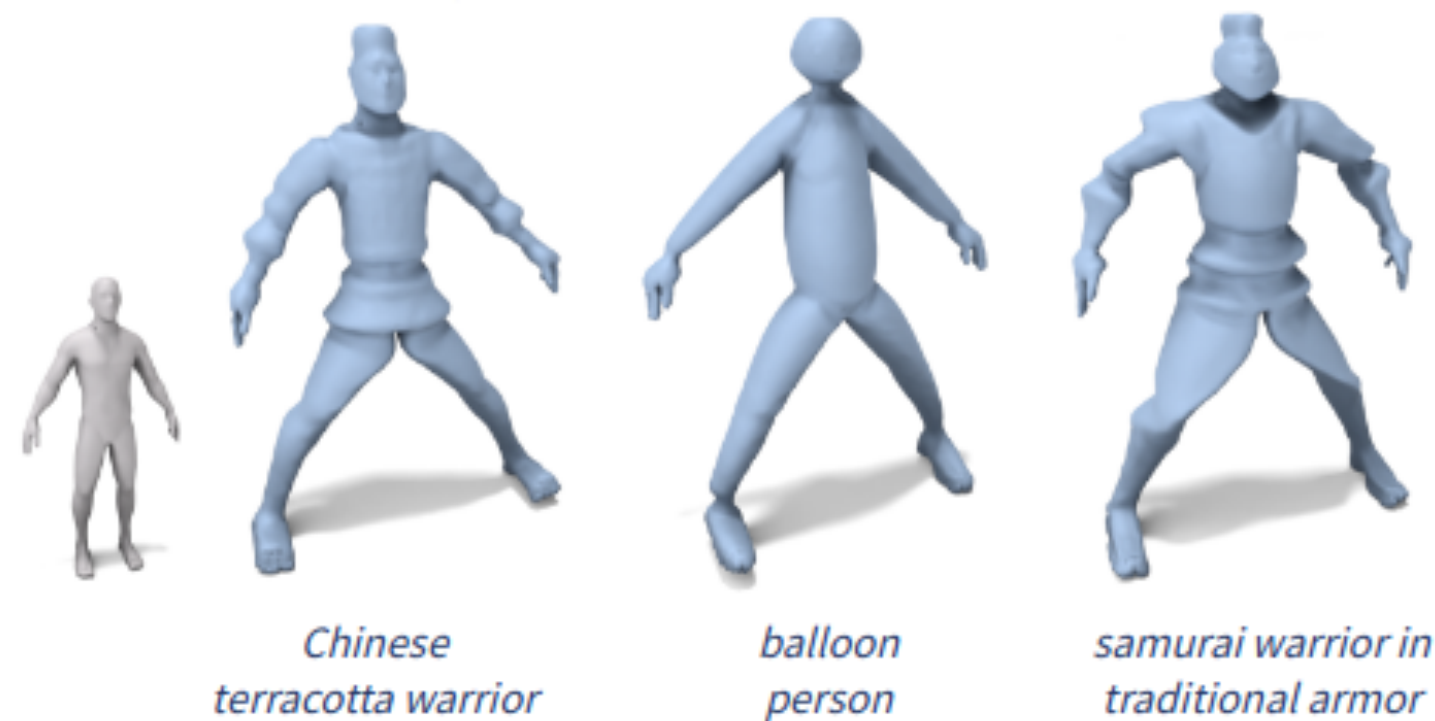
# dARAP: Global step

$$\hat{\mathcal{V}} = \operatorname{argmin}_{\tilde{\mathcal{V}}} \sum_{k \in \{1 \dots |\mathcal{V}|\}} \sum_{(i,j) \in \mathcal{N}_k} w_{ij} \|R_k \mathbf{e}_{ij} - \tilde{\mathbf{e}}_{ij}\|_2^2$$

- Fixing per-vertex rotations, find the best fit deformed vertex positions
- Is a Poisson equation with cotangent laplacian
  - → Use differentiable solving technique from Neural Jacobian Fields (Aigerman et al., 2022)
- One local step + global step pair, no iterating many like in classical ARAP.

# More results

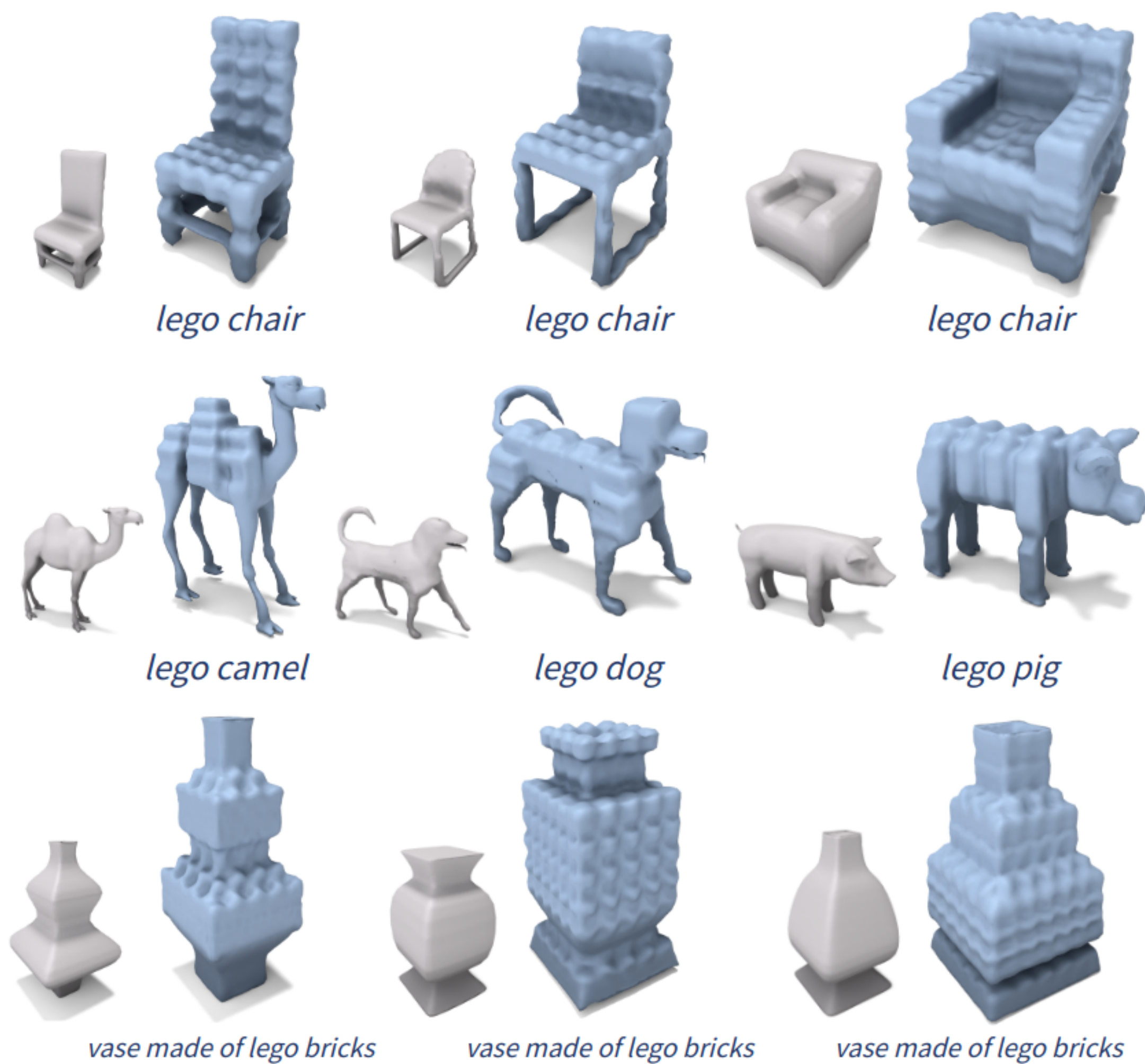
 a 3d render of an A-pose...



 a 3d render of a(n)...



 a 3d render of a...





# What makes this supervision work?



- **Silhouette, shading** in images smoothly reflect the underlying geometry



- Silhouette: High-level shape
- Shading: Surface texture, normals, bumps



- ARAP rotation-based representation gives desired solution space for task
- Linear solves smooth out noisy SDS supervision

a 3d render of a pineapple-themed vase

**More than just stylization...**

# More than just stylization...

$$\hat{q} = [\underbrace{u_x, u_y, u_z}_{\text{per-vertex rotation}}, s_x, s_y, s_z]$$

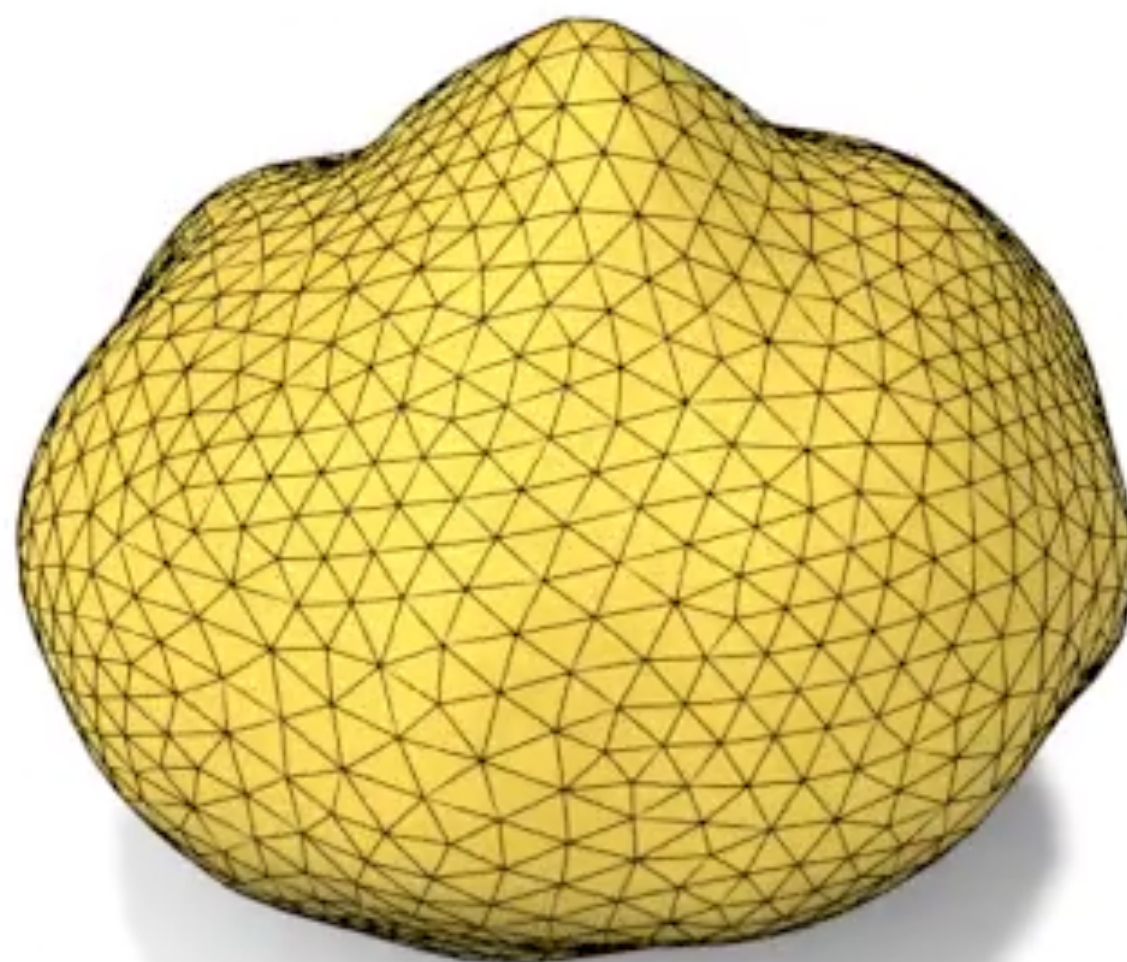
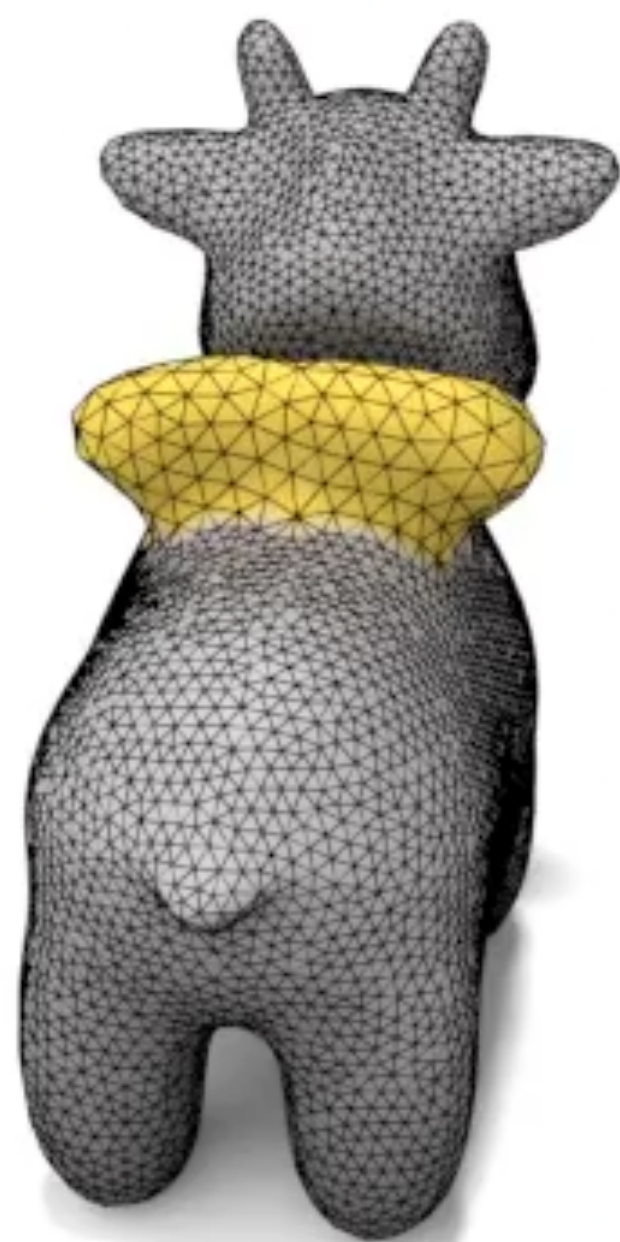


# More than just stylization...

$$\hat{q} = [\underbrace{u_x, u_y, u_z}_{\text{per-vertex rotation}}, \underbrace{s_x, s_y, s_z}_{\text{per-vertex scale (NEW)}}] + \text{remeshing}$$

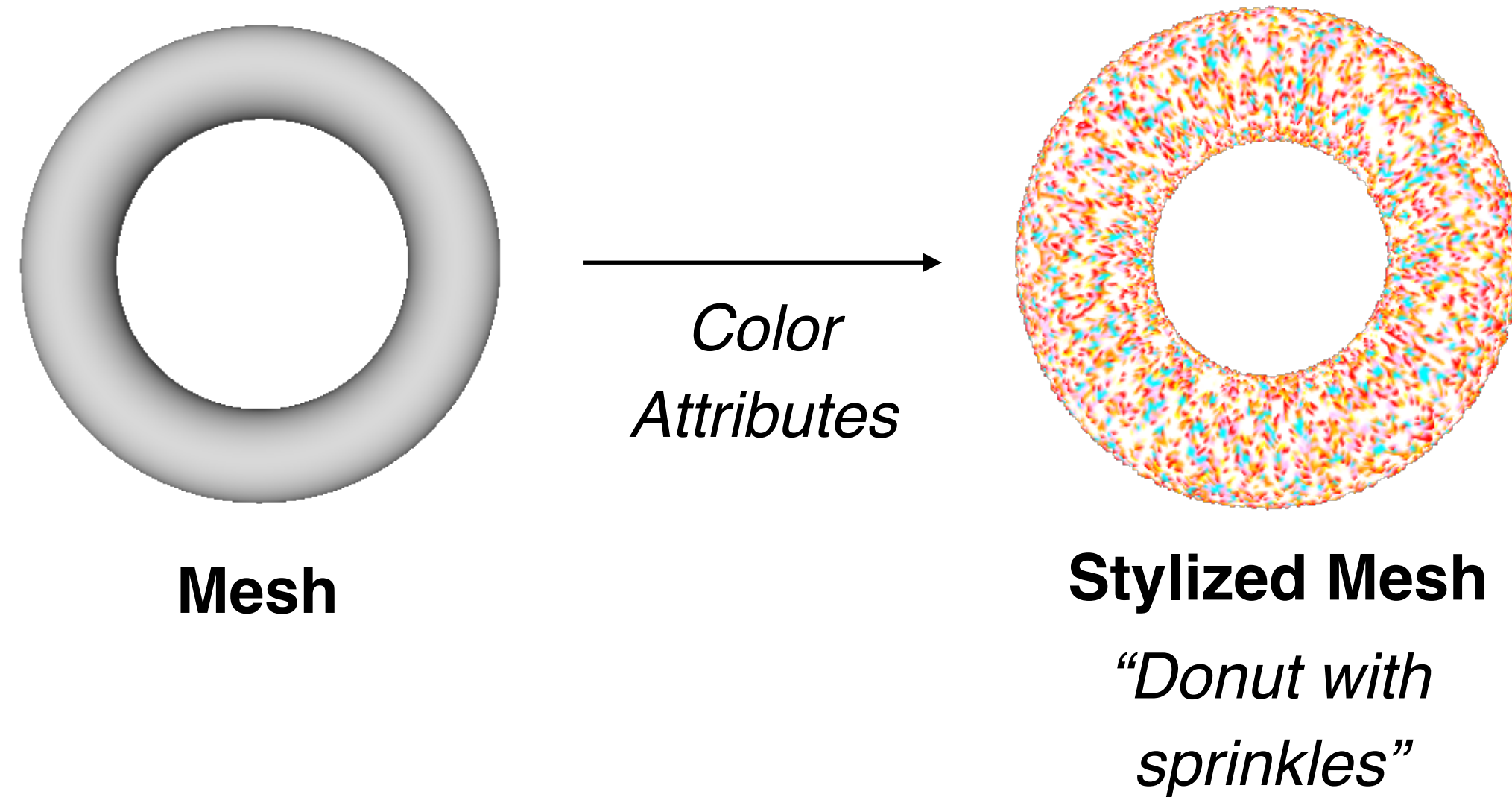
# More than just stylization...

$$\hat{q} = [\underbrace{u_x, u_y, u_z}_{\text{per-vertex rotation}}, \underbrace{s_x, s_y, s_z}_{\text{per-vertex scale (NEW)}}] + \text{remeshing}$$



# Why does optimization work so well for generation?

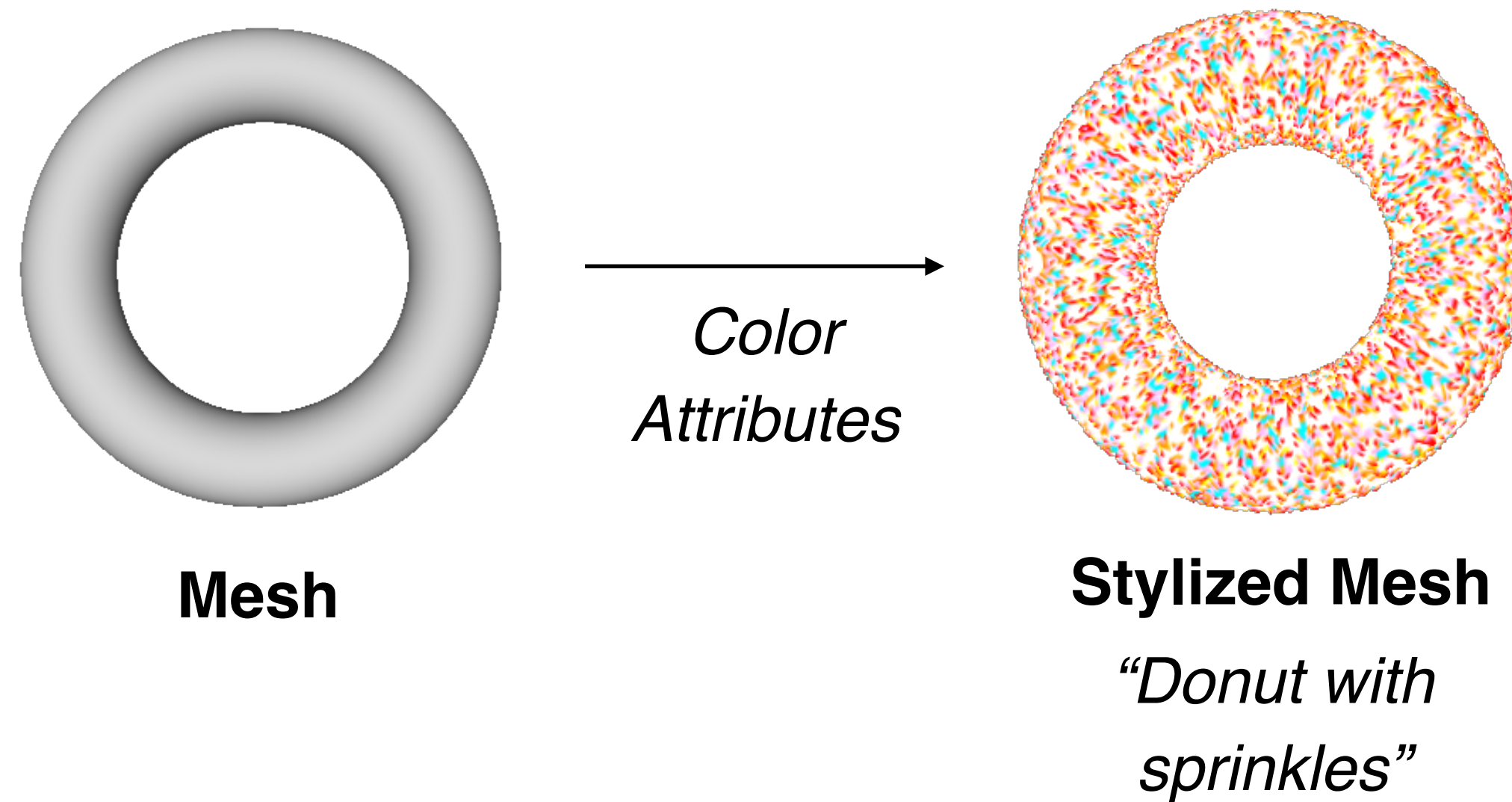
Generative methods typically have these key properties...





# Why does optimization work so well for generation?

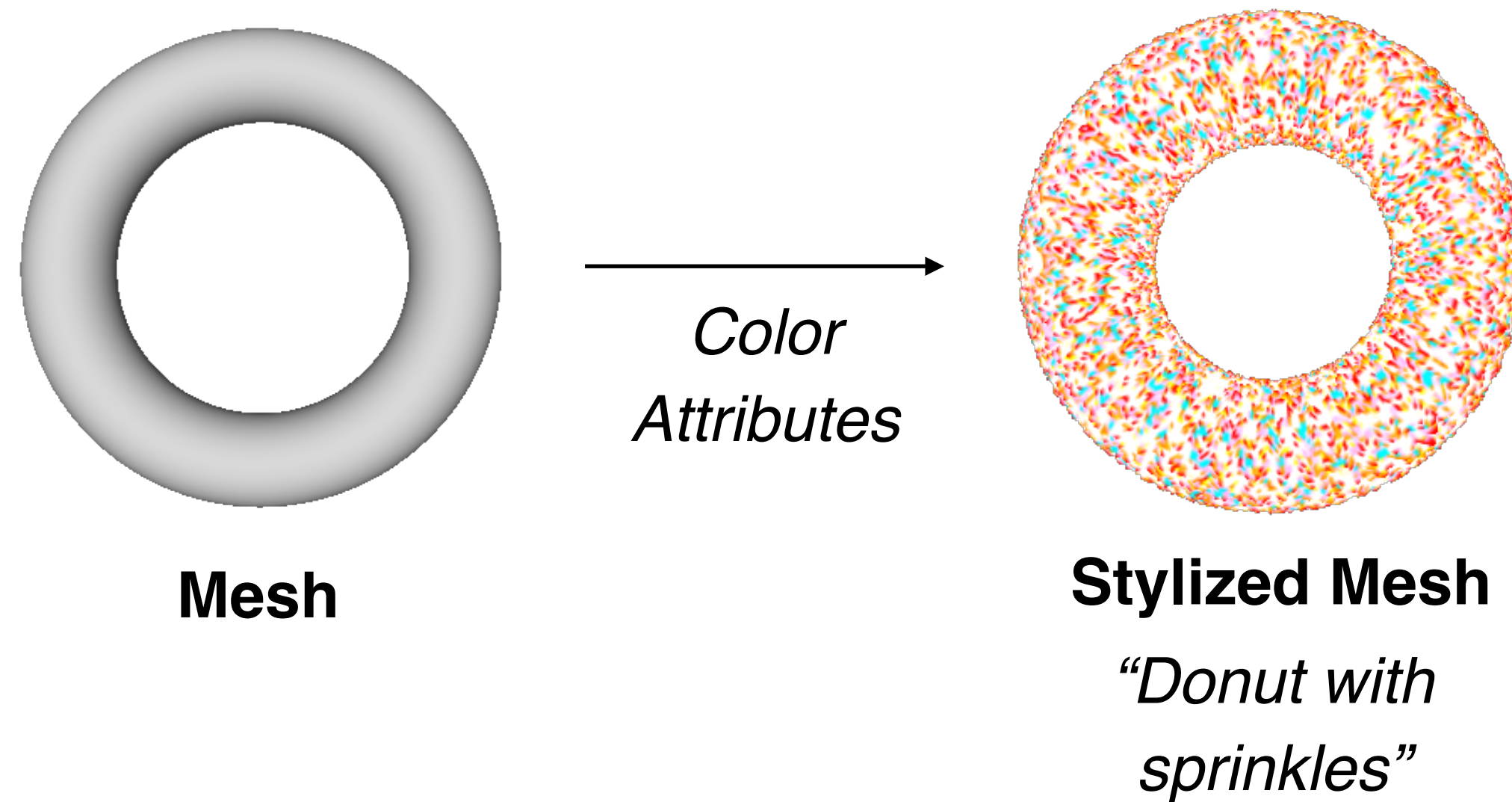
Generative methods typically have these key properties...



1. Naturally visualizable (colors, geometry)

# Why does optimization work so well for generation?

Generative methods typically have these key properties...



1. Naturally visualizable (colors, geometry)
2. Small changes in the 3D quantity result in smooth, interpretable changes in the visualization

# Visualizable Quantities



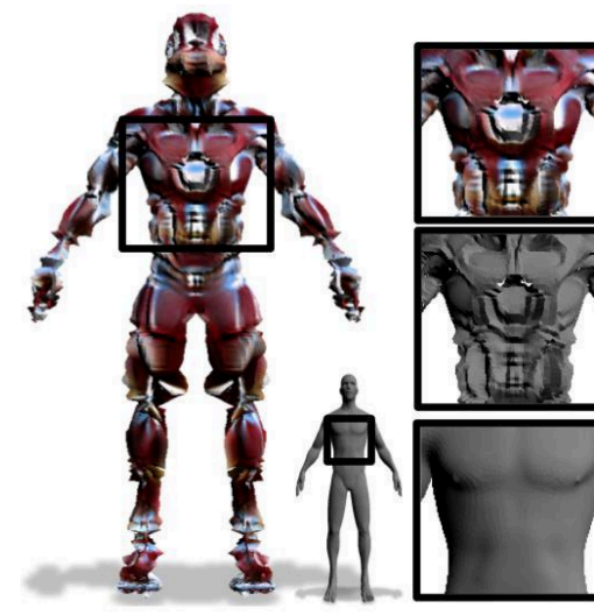
# Visualizable Quantities

Stylization

# Visualizable Quantities

Stylization

Visualize RGB  
Colors



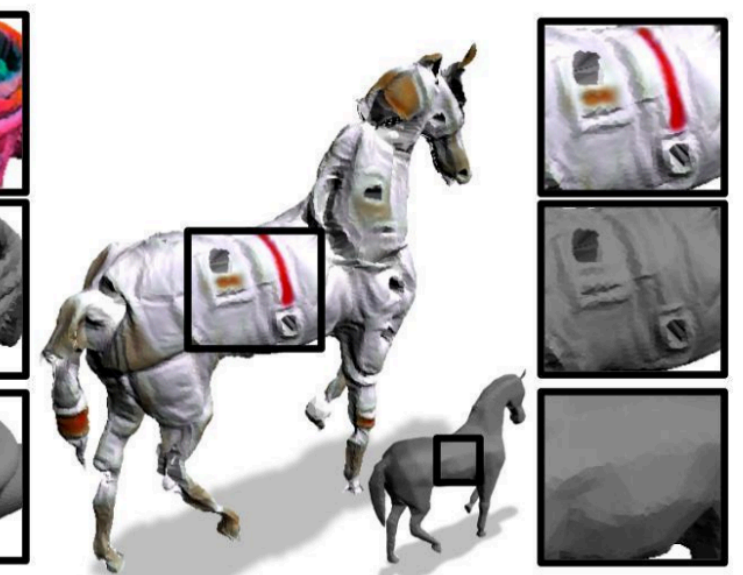
Iron Man



Brick Lamp



Colorful Crochet Candle

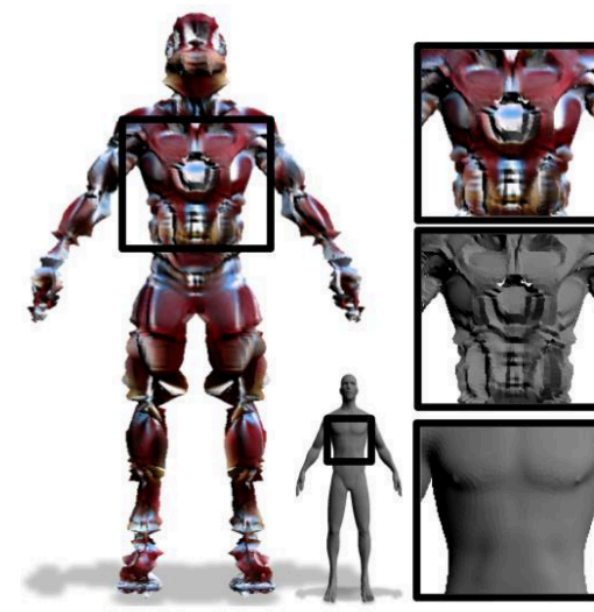


Astronaut Horse

# Visualizable Quantities

Stylization

Visualize RGB  
Colors



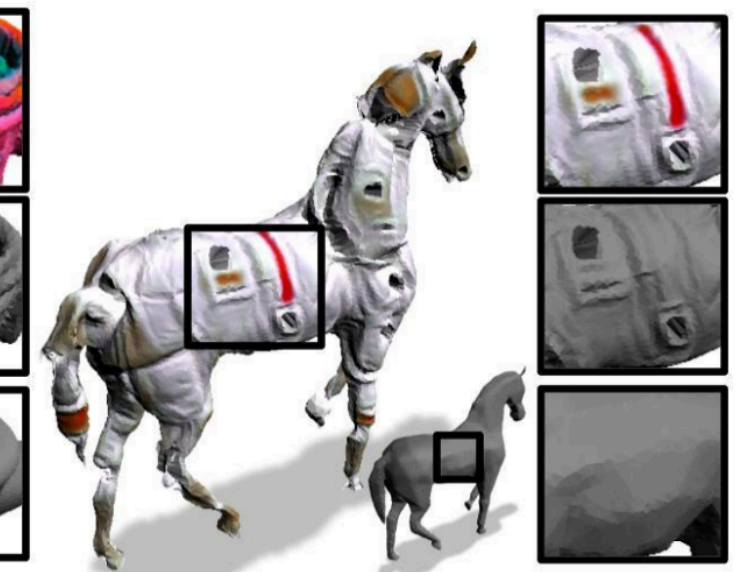
Iron Man



Brick Lamp



Colorful Crochet Candle



Astronaut Horse

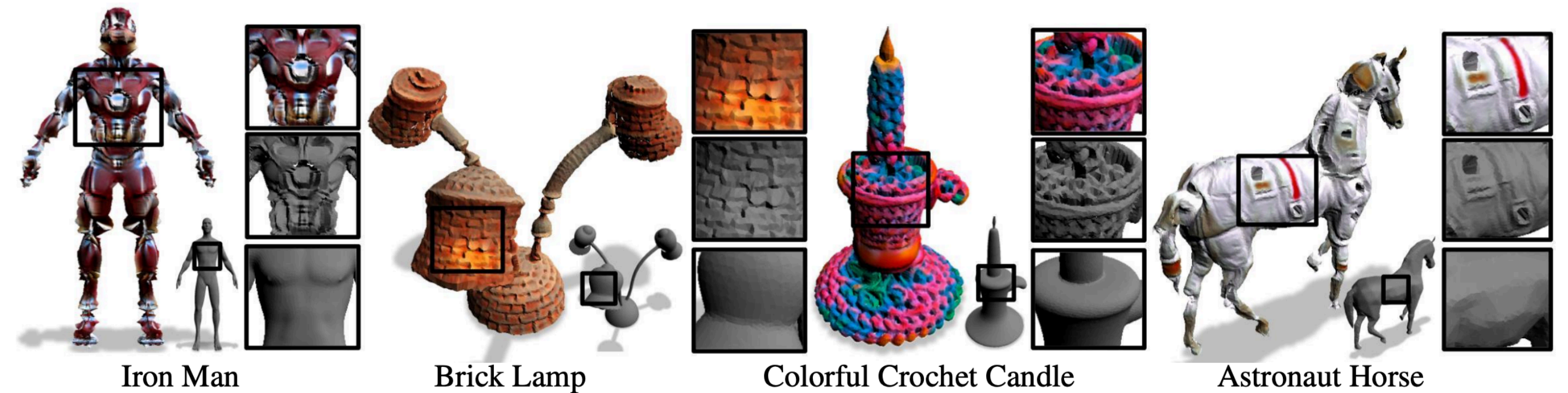
Deformation



# Visualizable Quantities

Stylization

Visualize RGB  
Colors



Deformation

Visualize  
Geometry



# Analysis via Synthesis

How can we use these optimization based losses beyond generation?

# Analysis via Synthesis

How can we use these optimization based losses beyond generation?

- Can we apply this to analytical quantities as well?



# Analysis via Synthesis

How can we use these optimization based losses beyond generation?

- Can we apply this to analytical quantities as well?
- Need to be able to produce smooth visualizations of our analytical quantity!

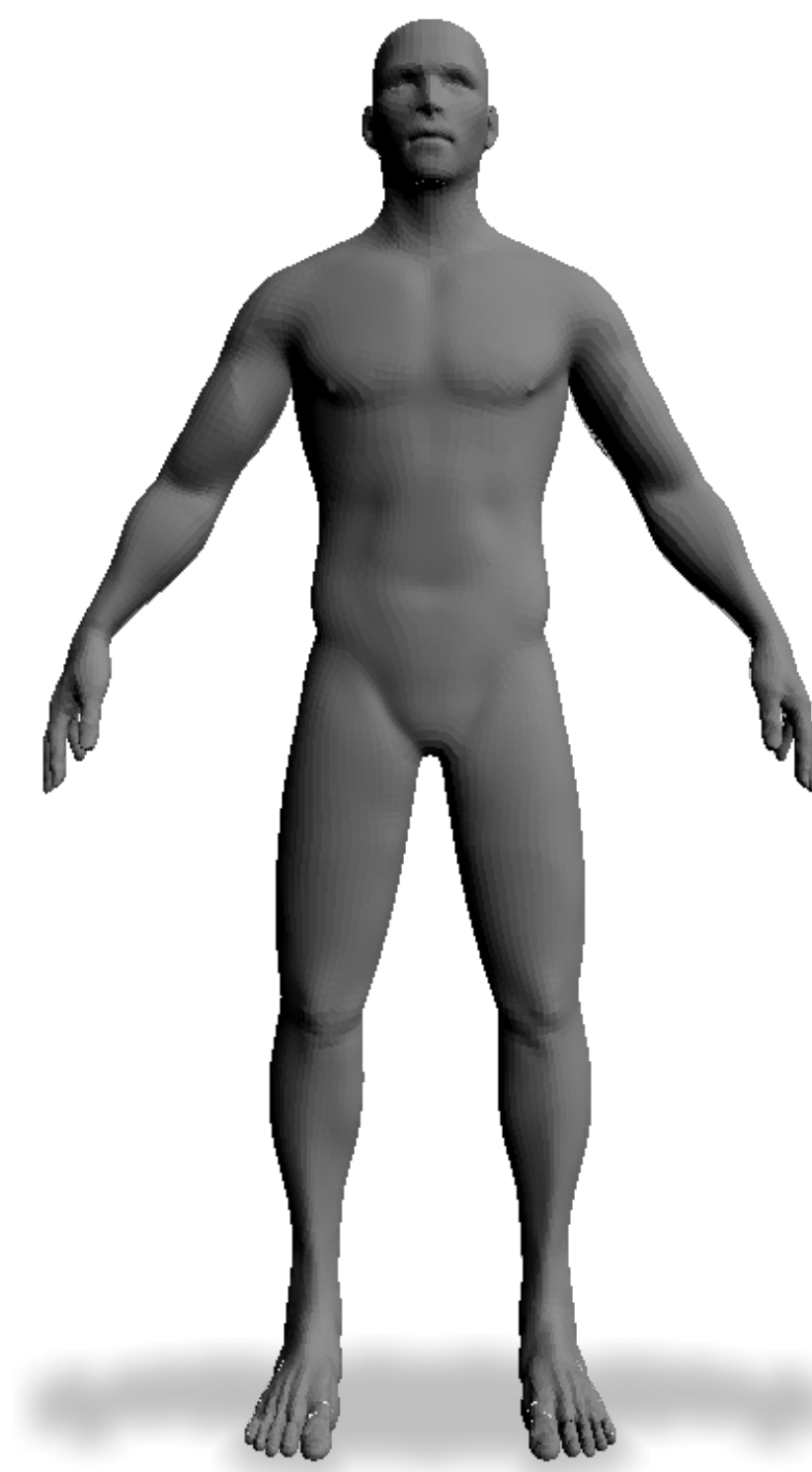
# Analysis via Synthesis

How can we use these optimization based losses beyond generation?

- Can we apply this to analytical quantities as well?
- Need to be able to produce smooth visualizations of our analytical quantity!

Can we apply this to something like segmentation?

# Analysis via Synthesis



**Input**

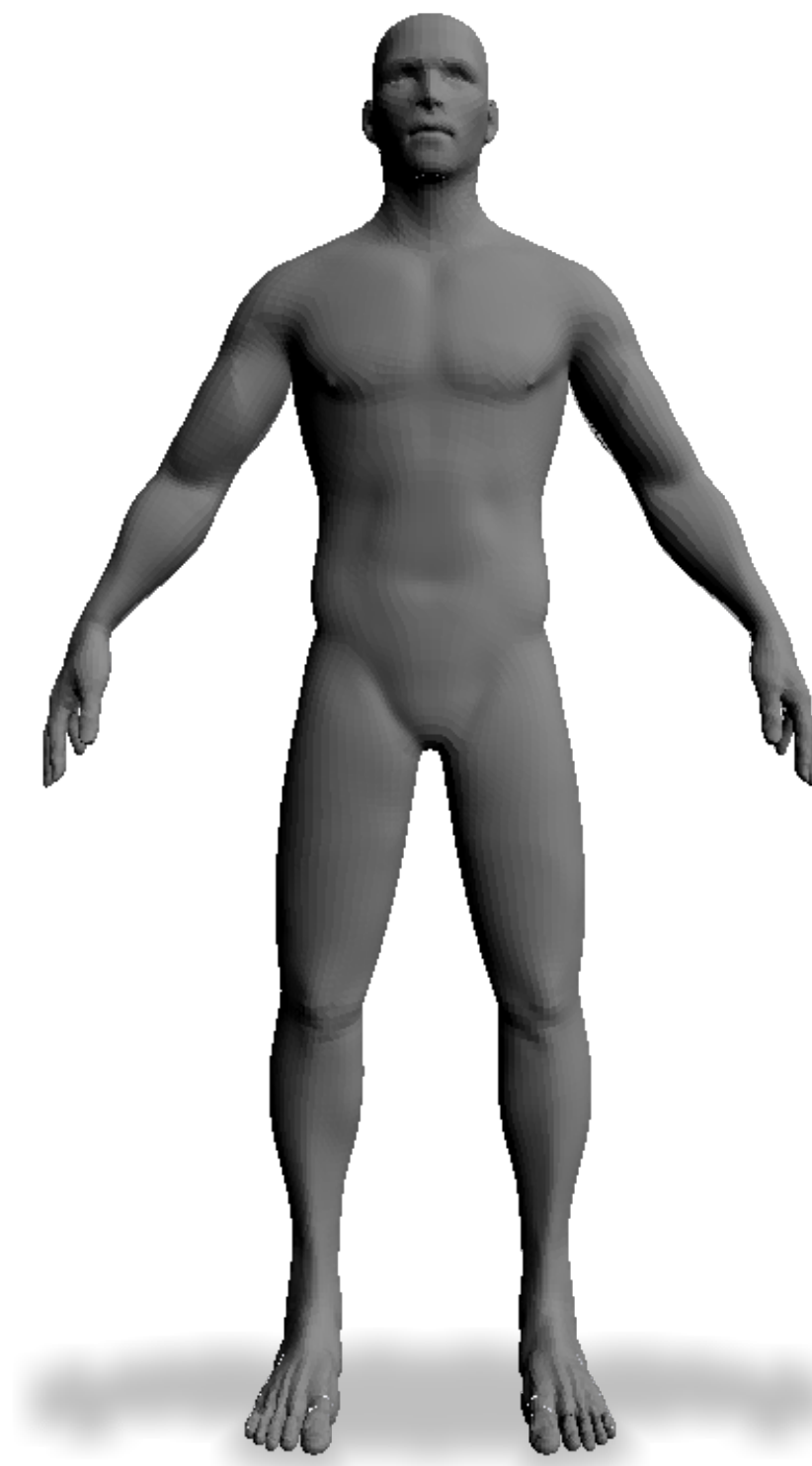


**Steve Jobs**



# Analysis via Synthesis

- Generative approaches demonstrate semantic part understanding



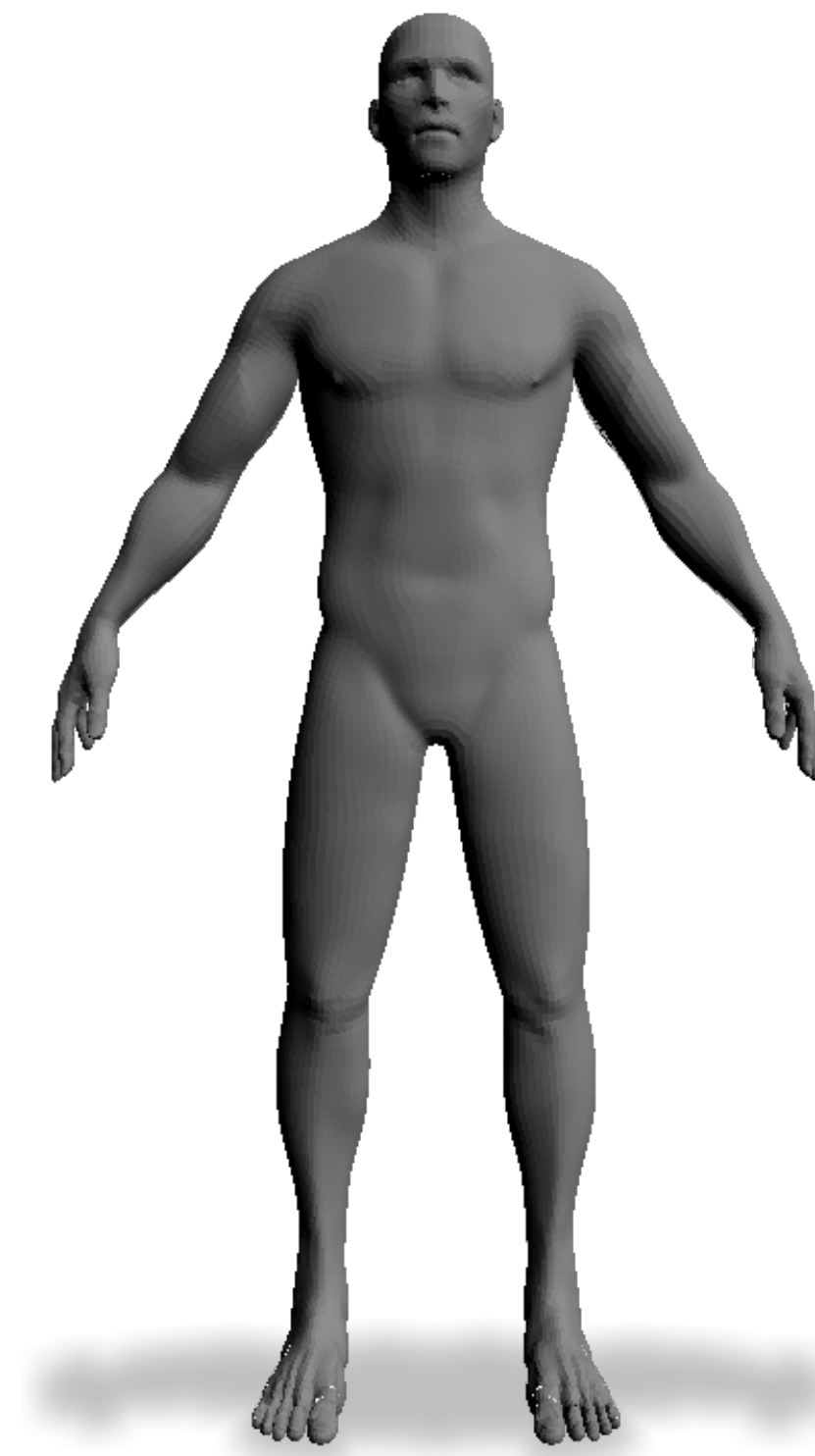
**Input**



**Steve Jobs**

# Analysis via Synthesis

- Generative approaches demonstrate semantic part understanding



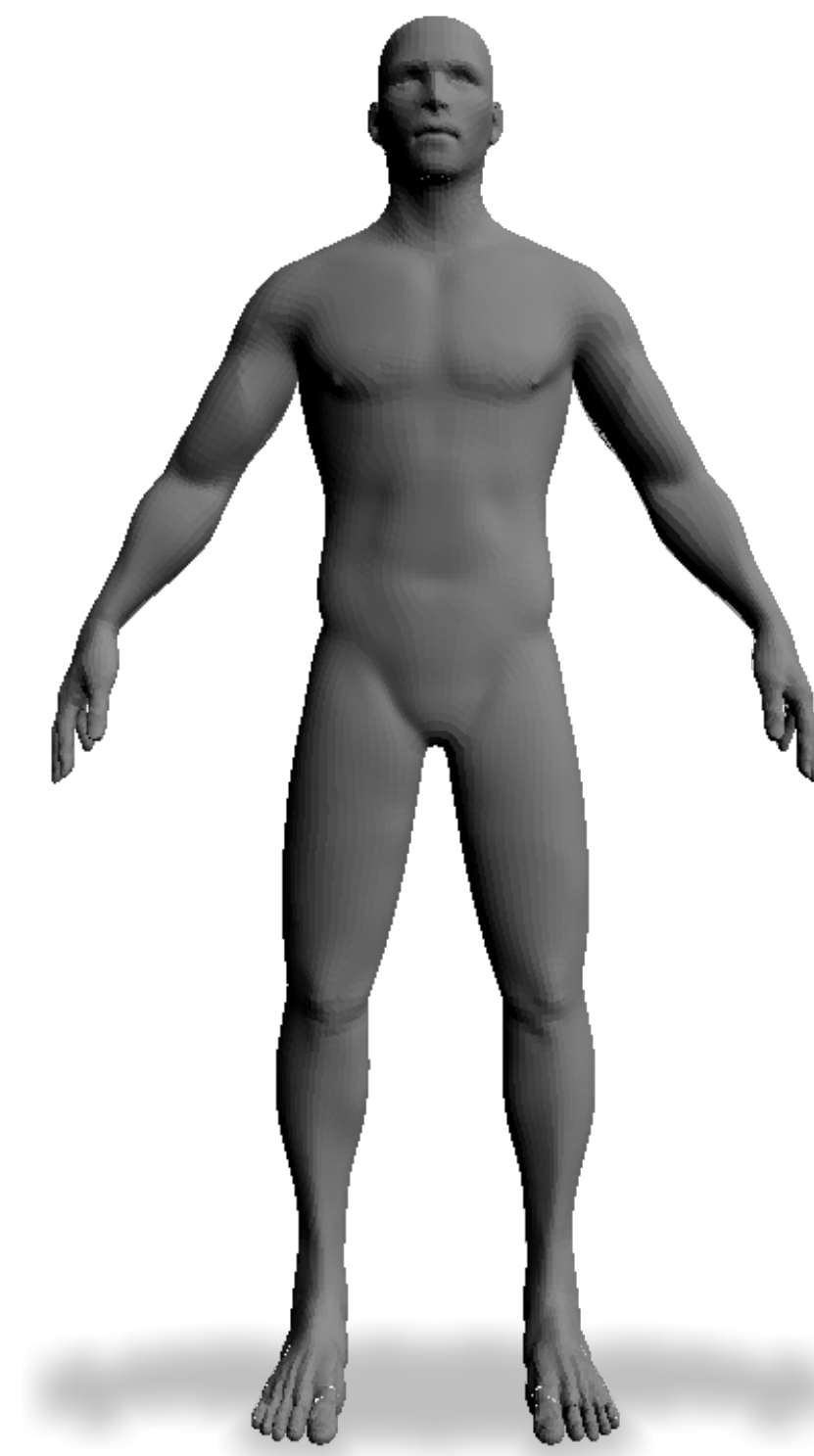
**Input**



**Steve Jobs**

# Analysis via Synthesis

- Generative approaches demonstrate semantic part understanding



**Input**

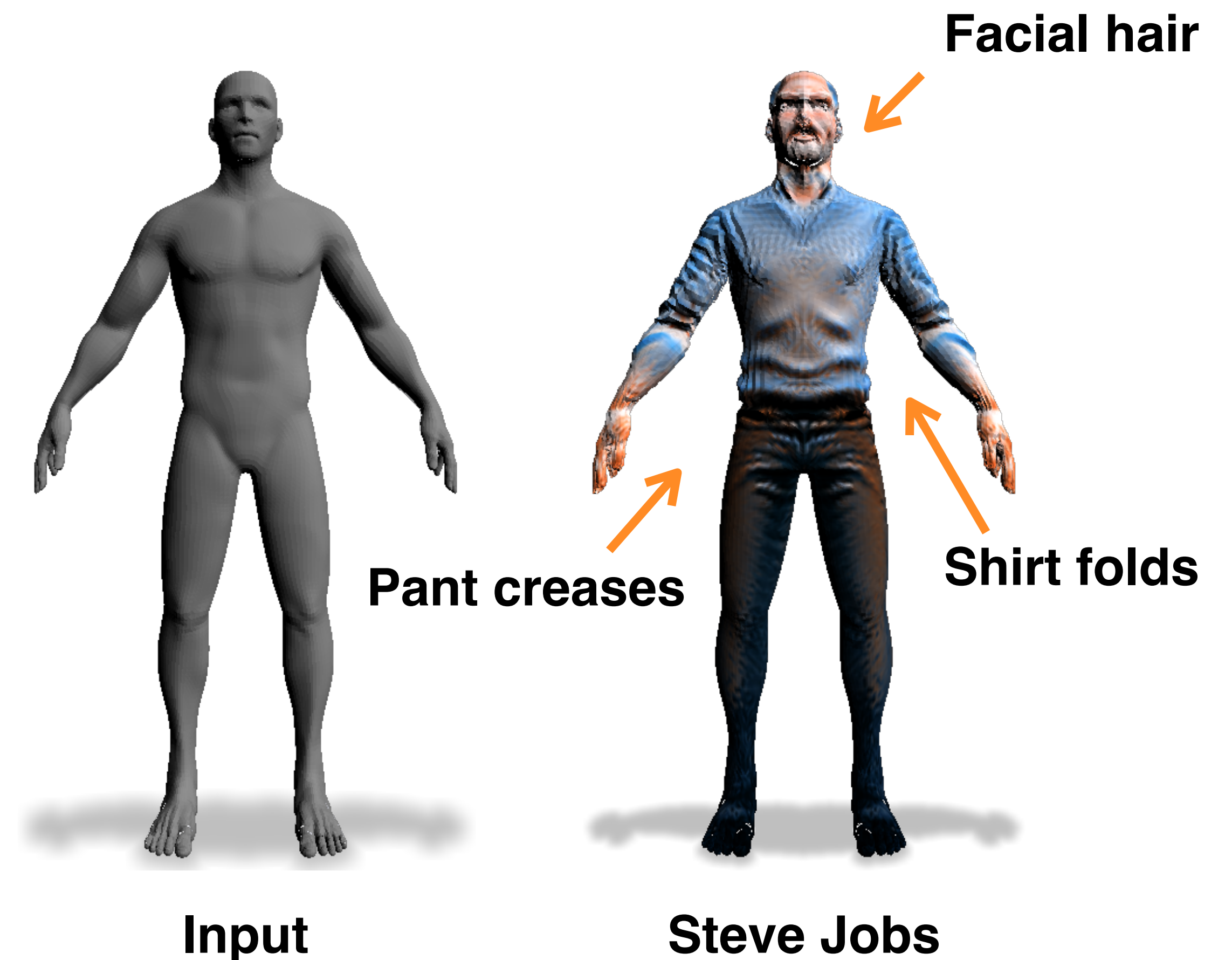


**Steve Jobs**



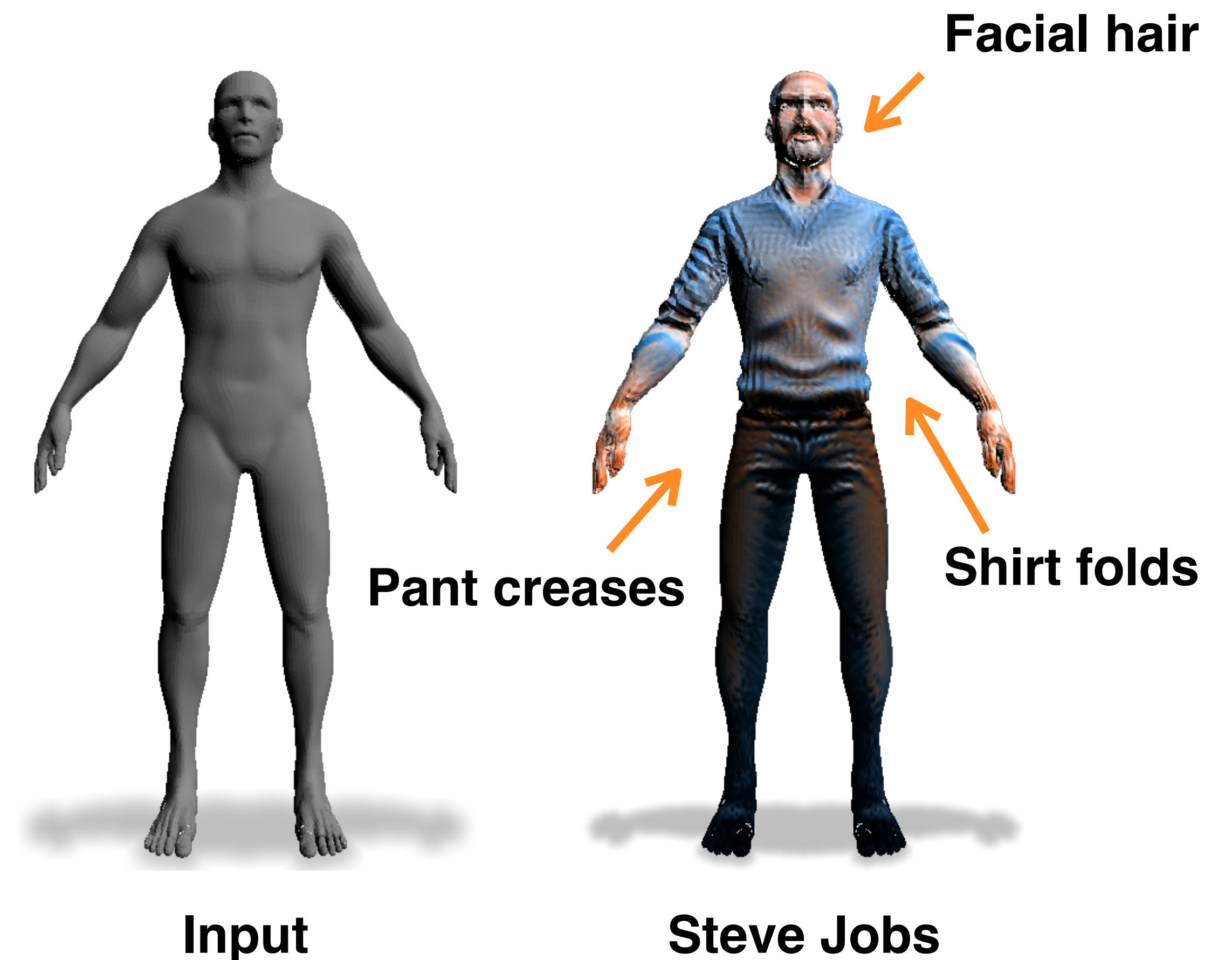
# Analysis via Synthesis

- Generative approaches demonstrate semantic part understanding



# Analysis via Synthesis

- Generative approaches demonstrate semantic part understanding
- No explicit segmentation, but can we tease it out with a visualization + optimization?



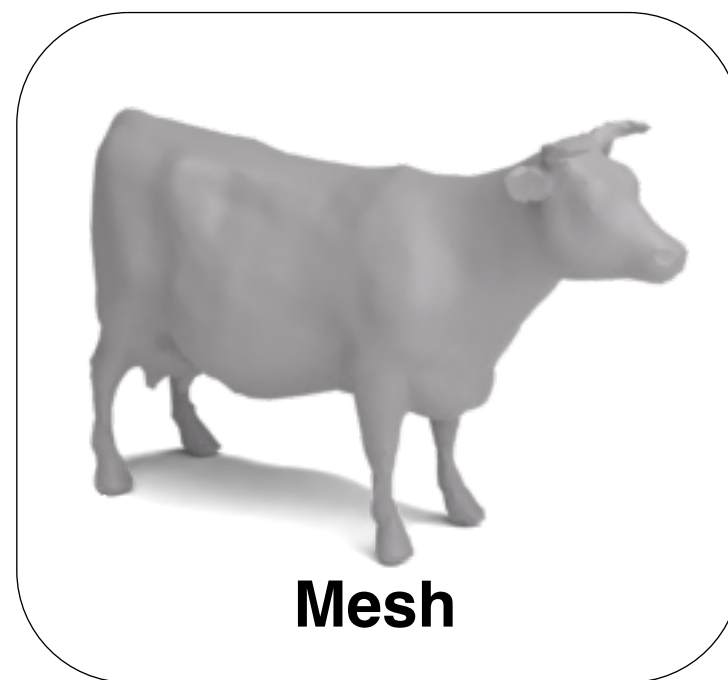
# How do we visually represent a localization?





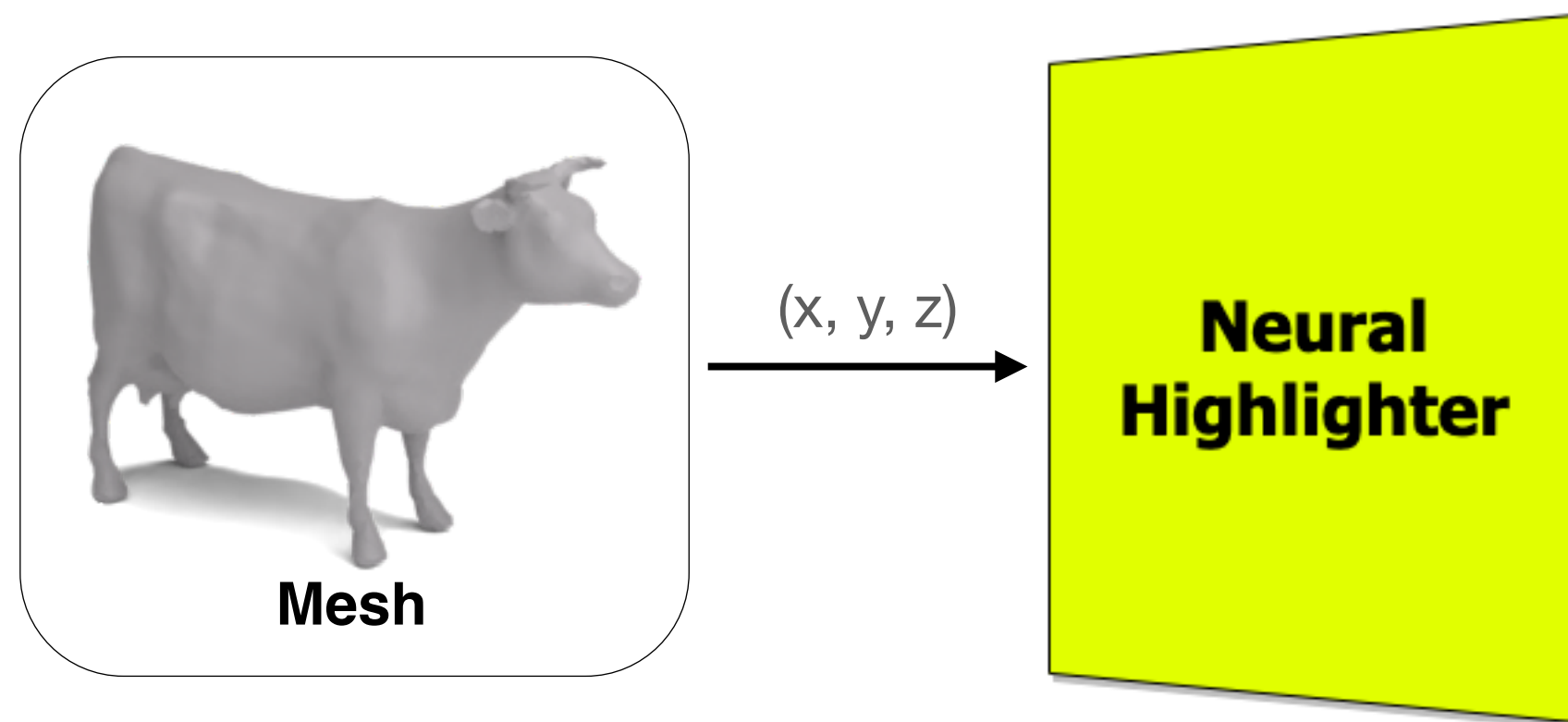
# How do we visually represent a localization?

- Segmentations are typically discrete...



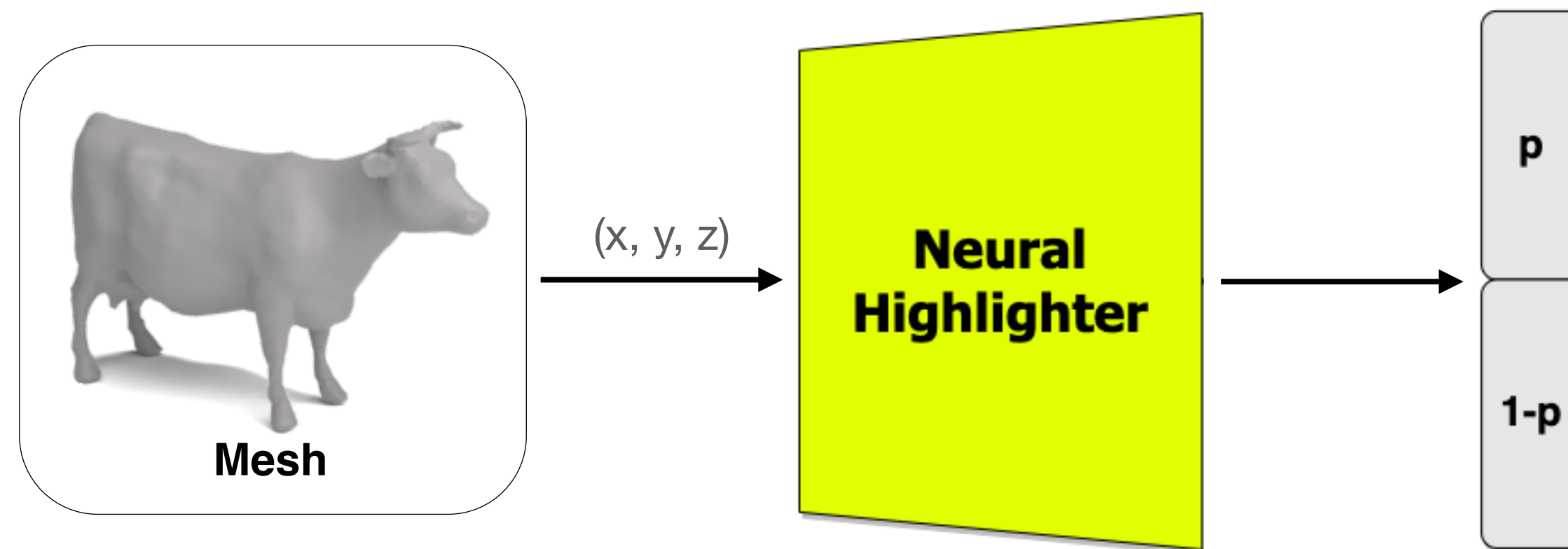
# How do we visually represent a localization?

- Segmentations are typically discrete...



# How do we visually represent a localization?

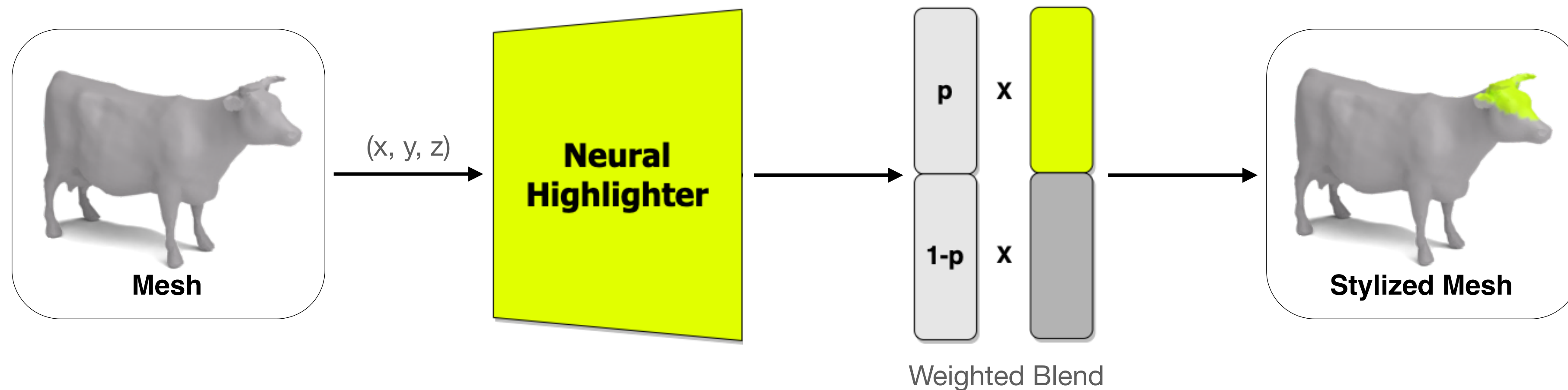
- Segmentations are typically discrete...



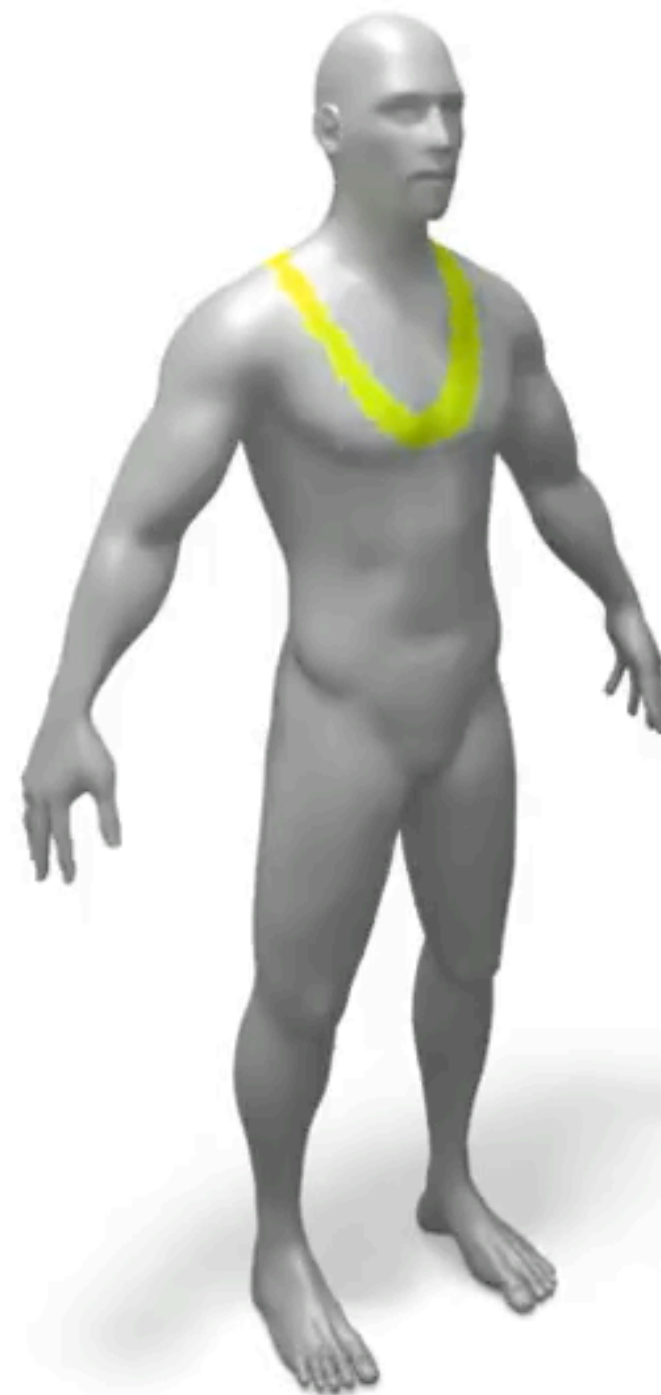


# How do we visually represent a localization?

- Segmentations are typically discrete...



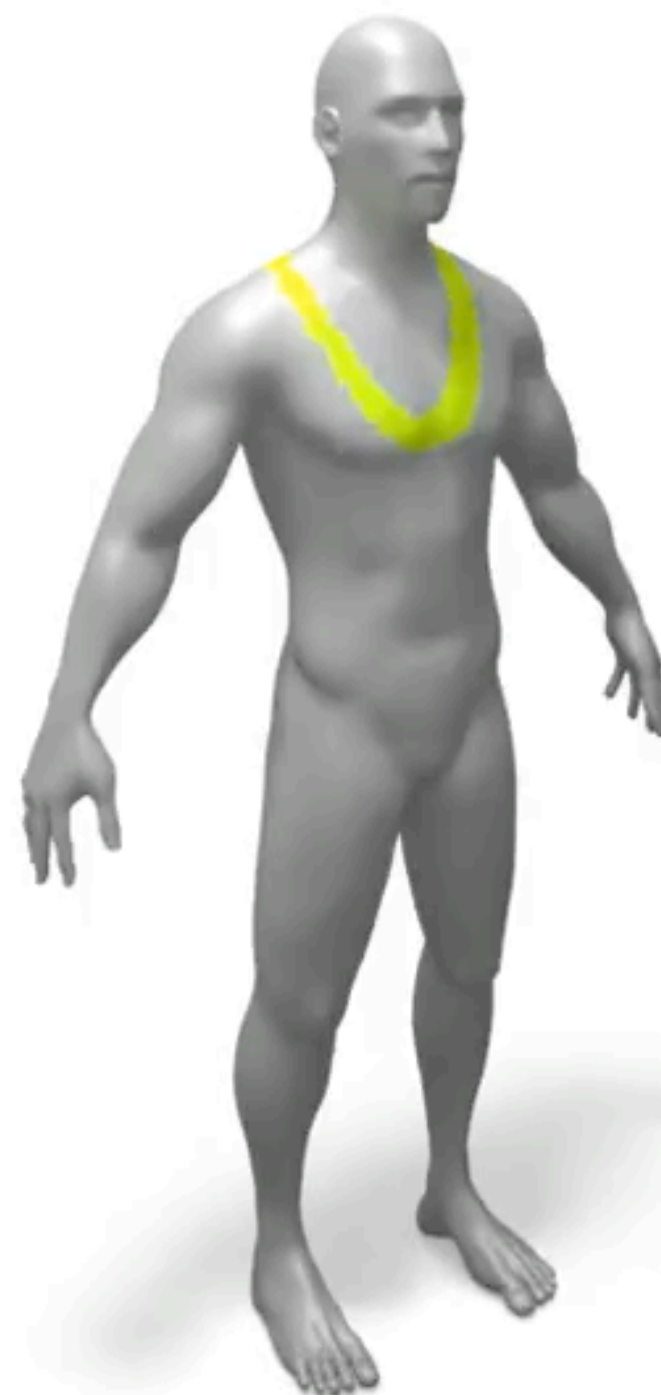
# Results!



# Results!



Poncho



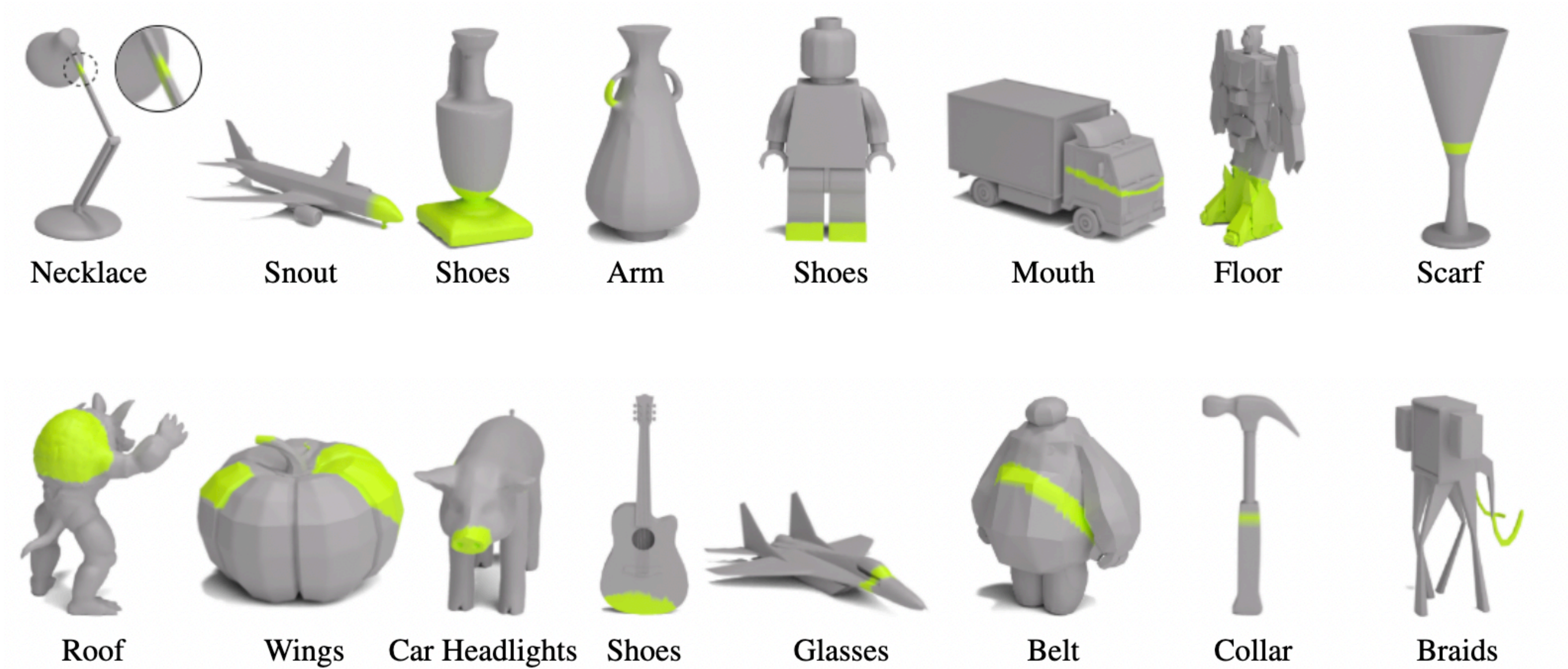
Necklace



Headphones



# Out of Distribution Localization



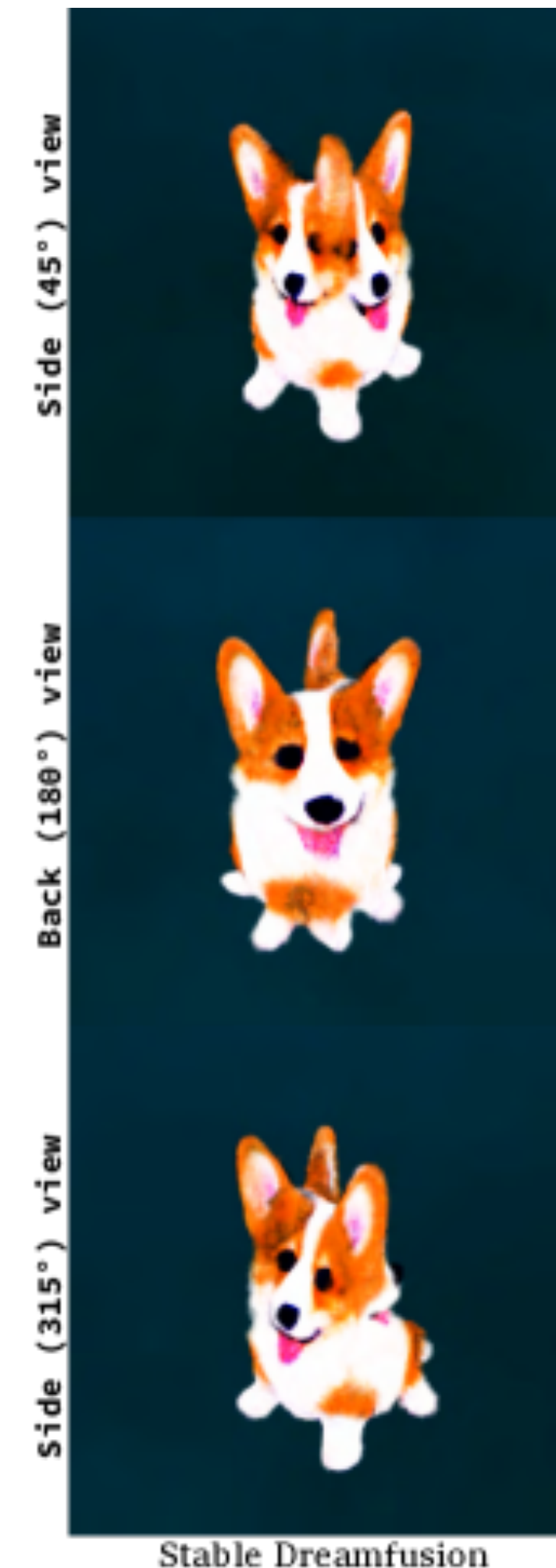
# Review of Optimization-Based Methods

## Pros

- Can distill the deep understanding of large 2D-trained models to 3D
- Usually no sharp boundaries between specific views due to many iterations

## Cons

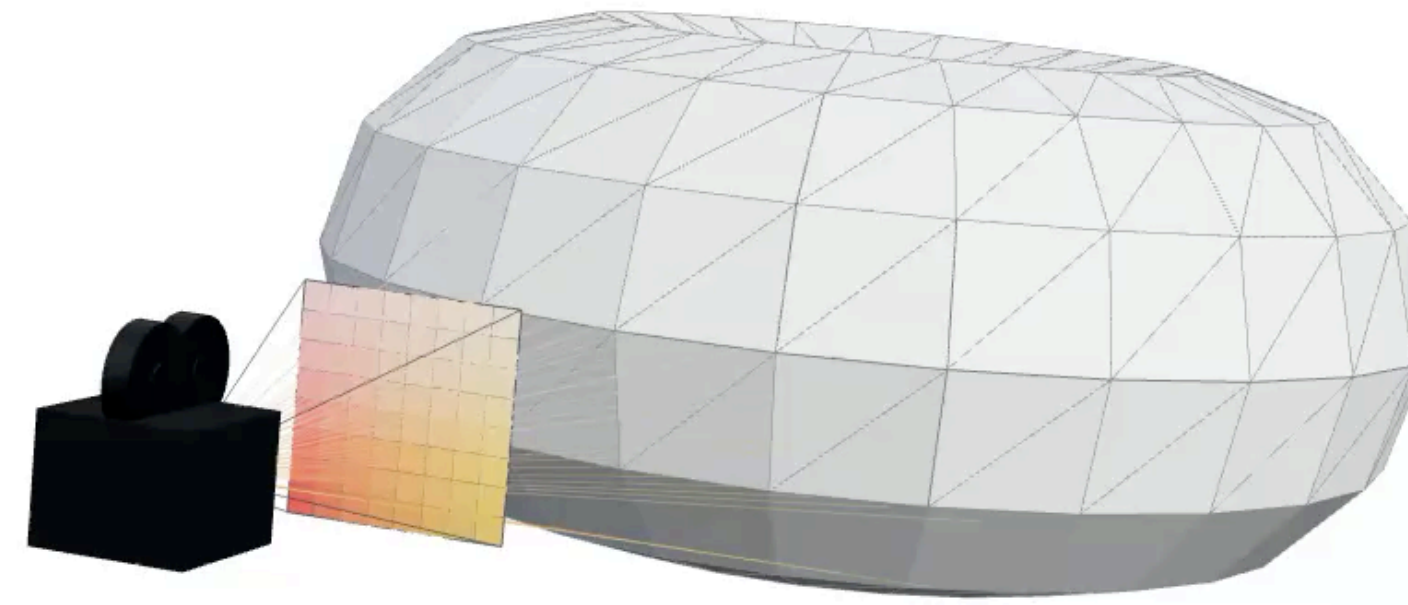
- Janus effect
- Very slow!



Stable Dreamfusion

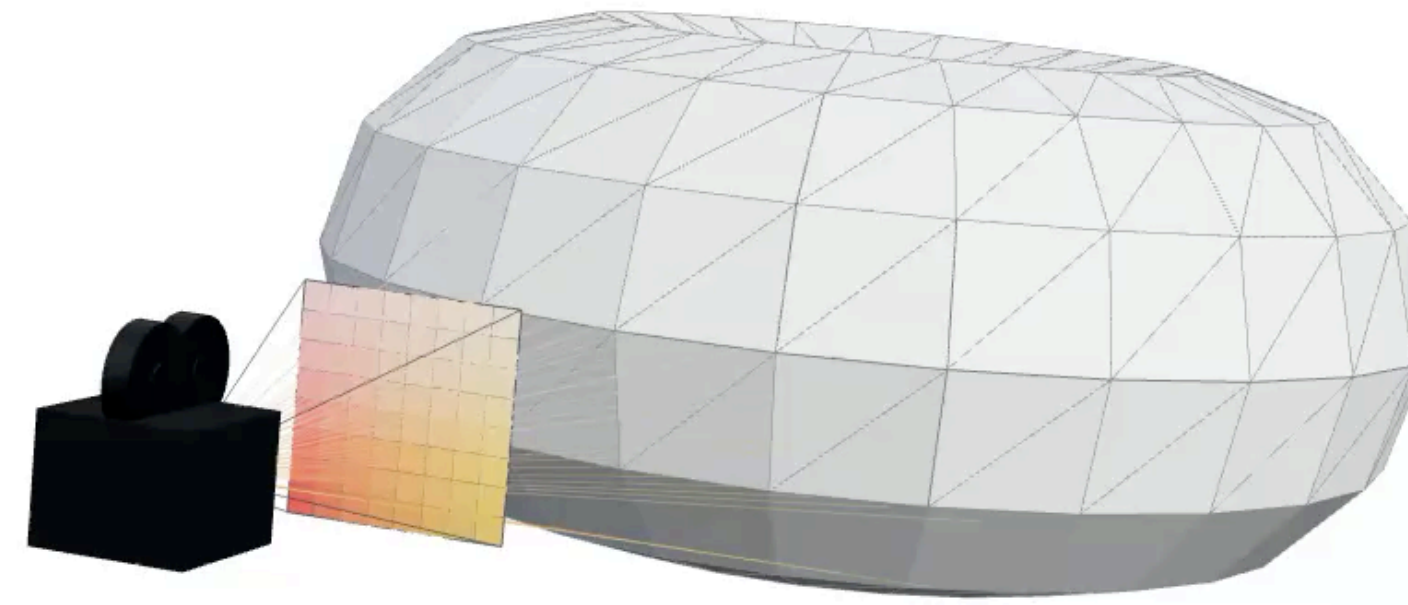


# Backprojection-Based Methods



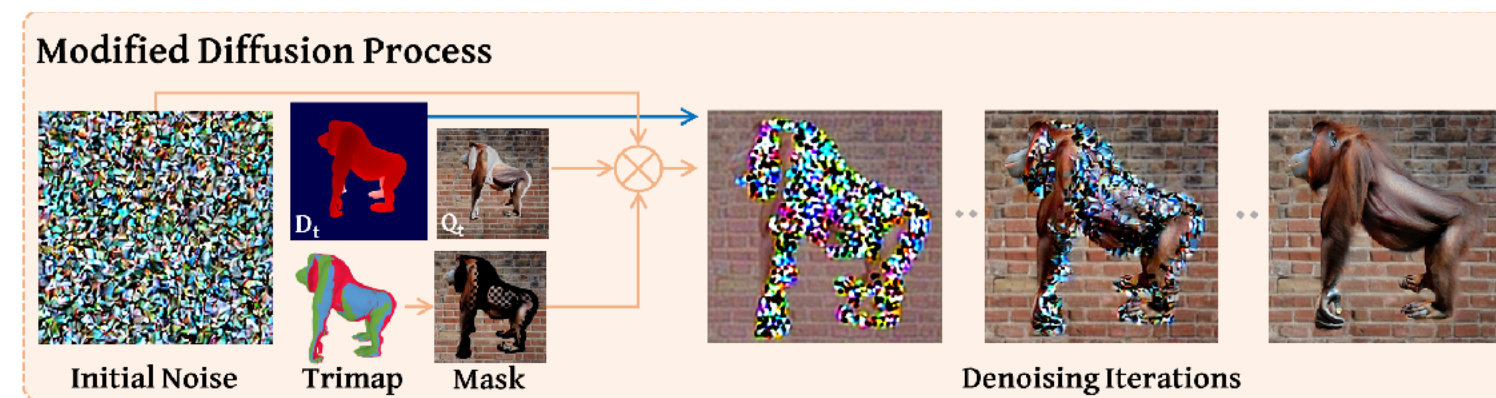


# Backprojection-Based Methods

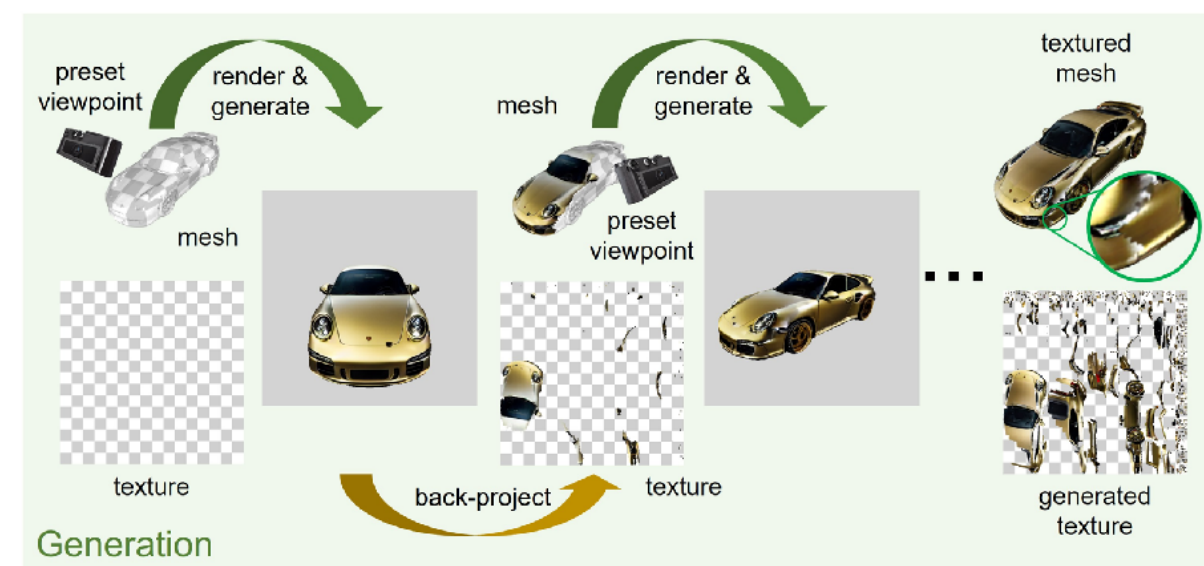


# Backprojection of...

## RGB Values

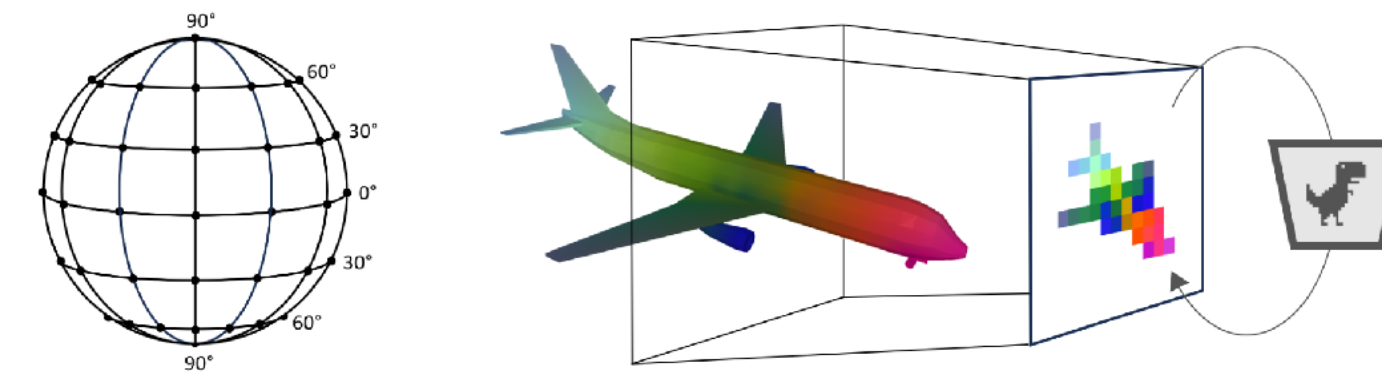


## TEXTure, Richardson et al. 2023

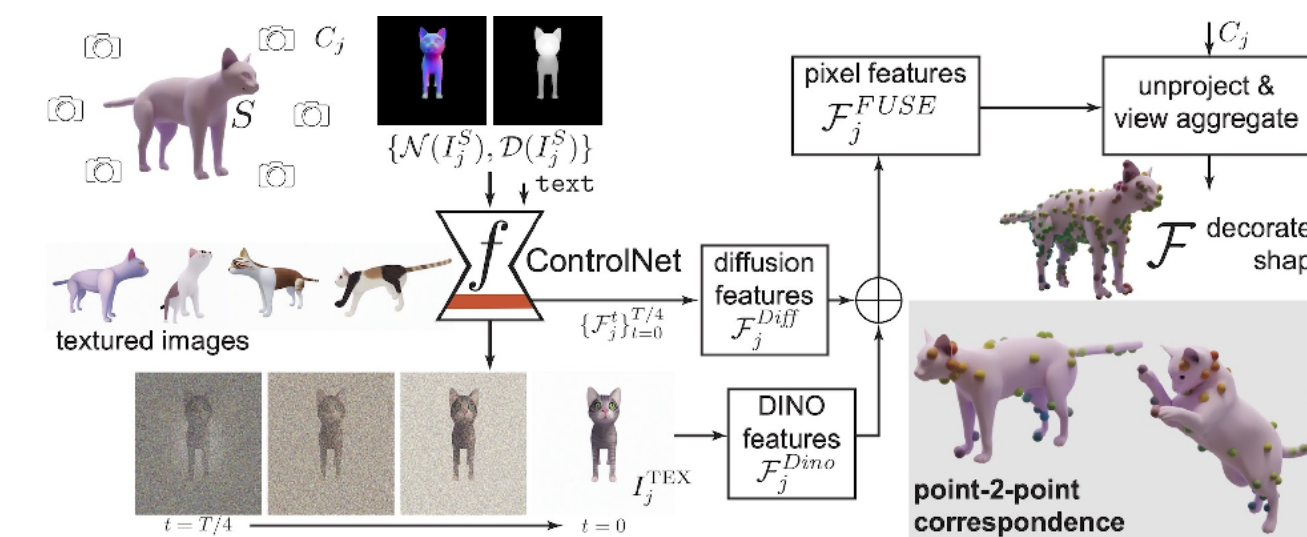


## Text2Tex, Chen et al. 2023

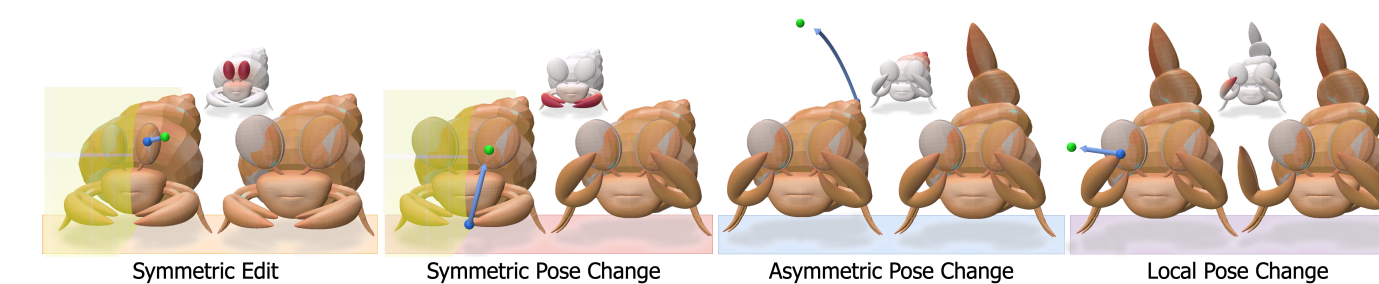
## Deep Features



## Back to 3D, Wimmer et al. 2023

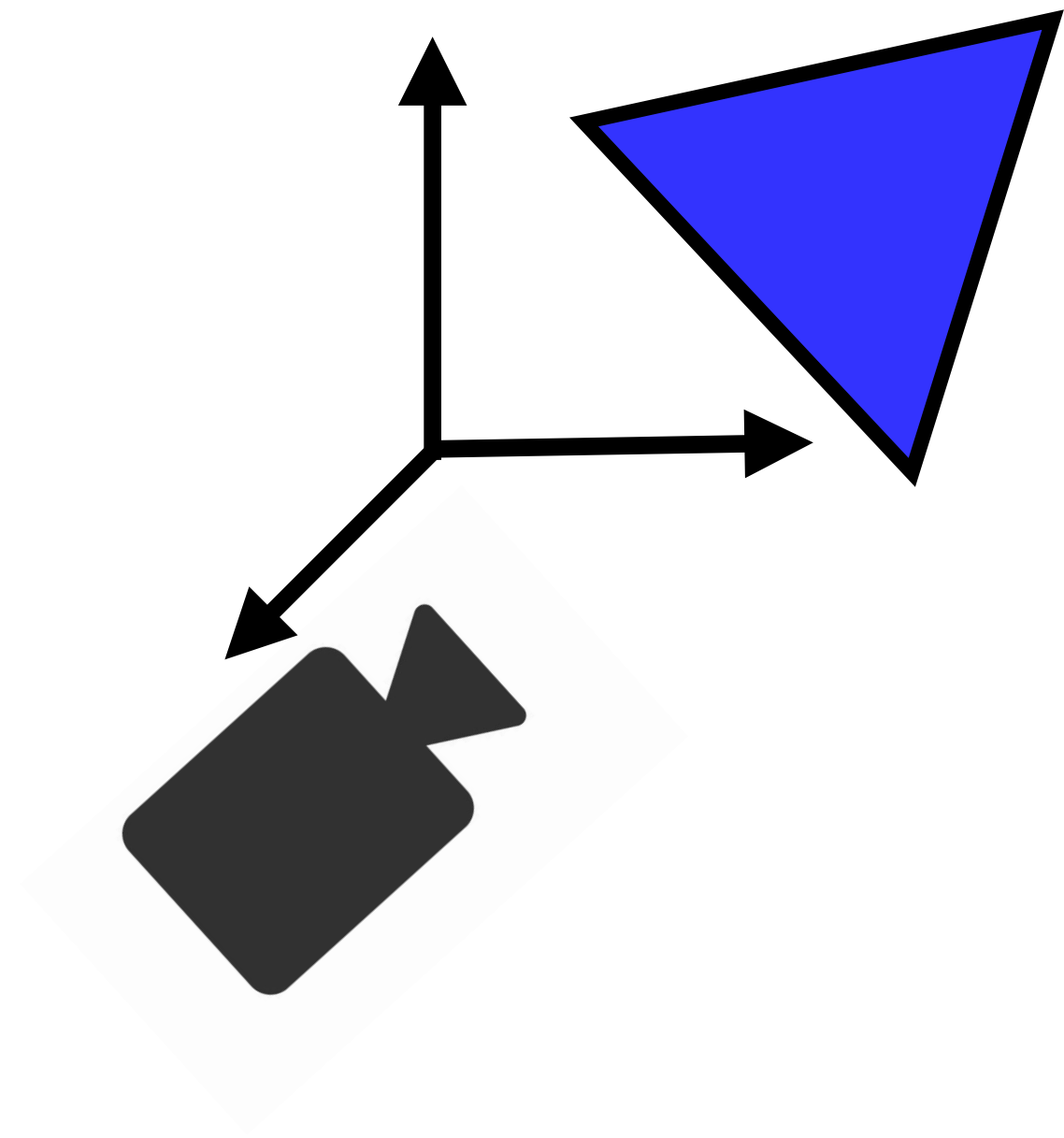


## Diff3f, Dutt et al. 2024



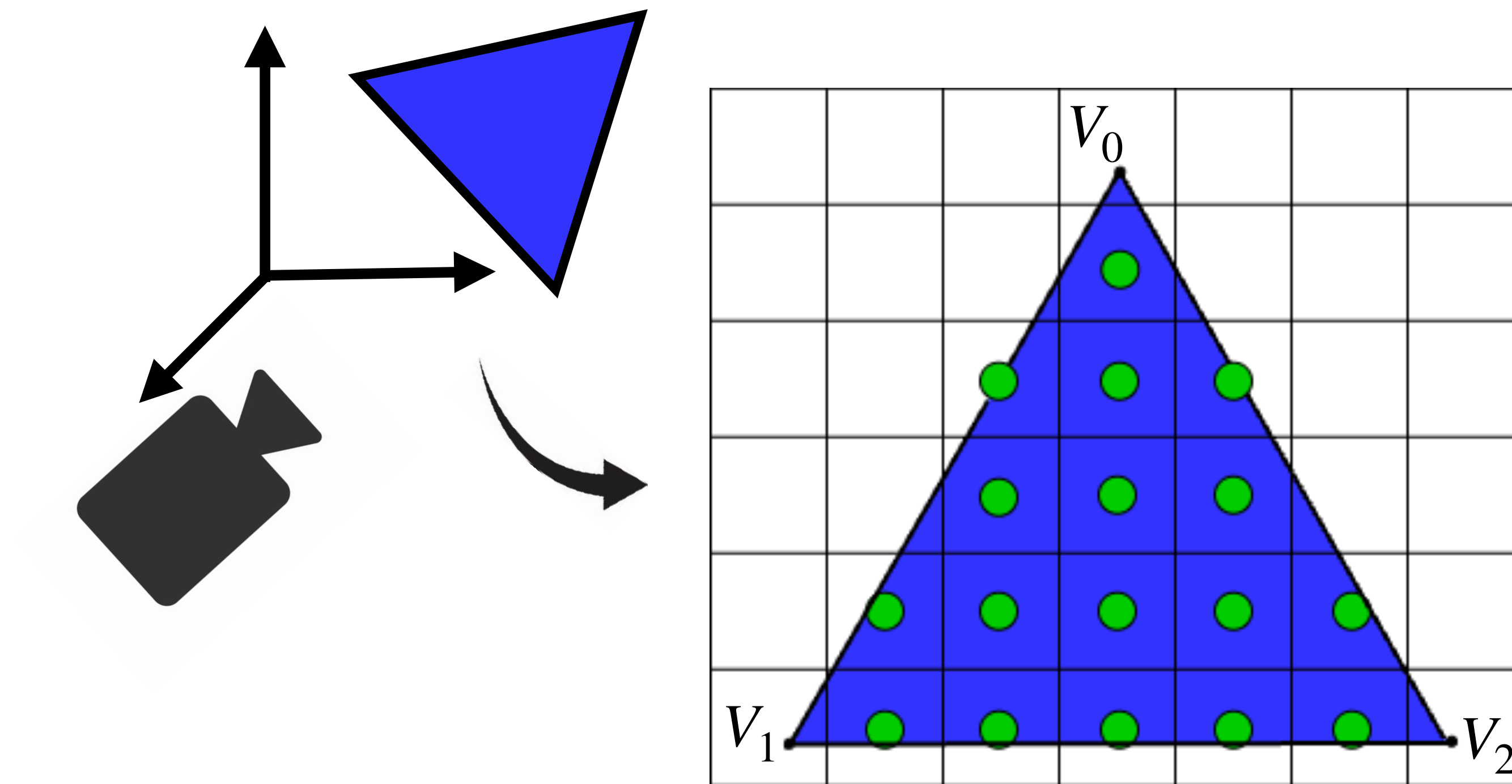
## DFD, Liu et al. 2026

# Pixel to 3D Mapping



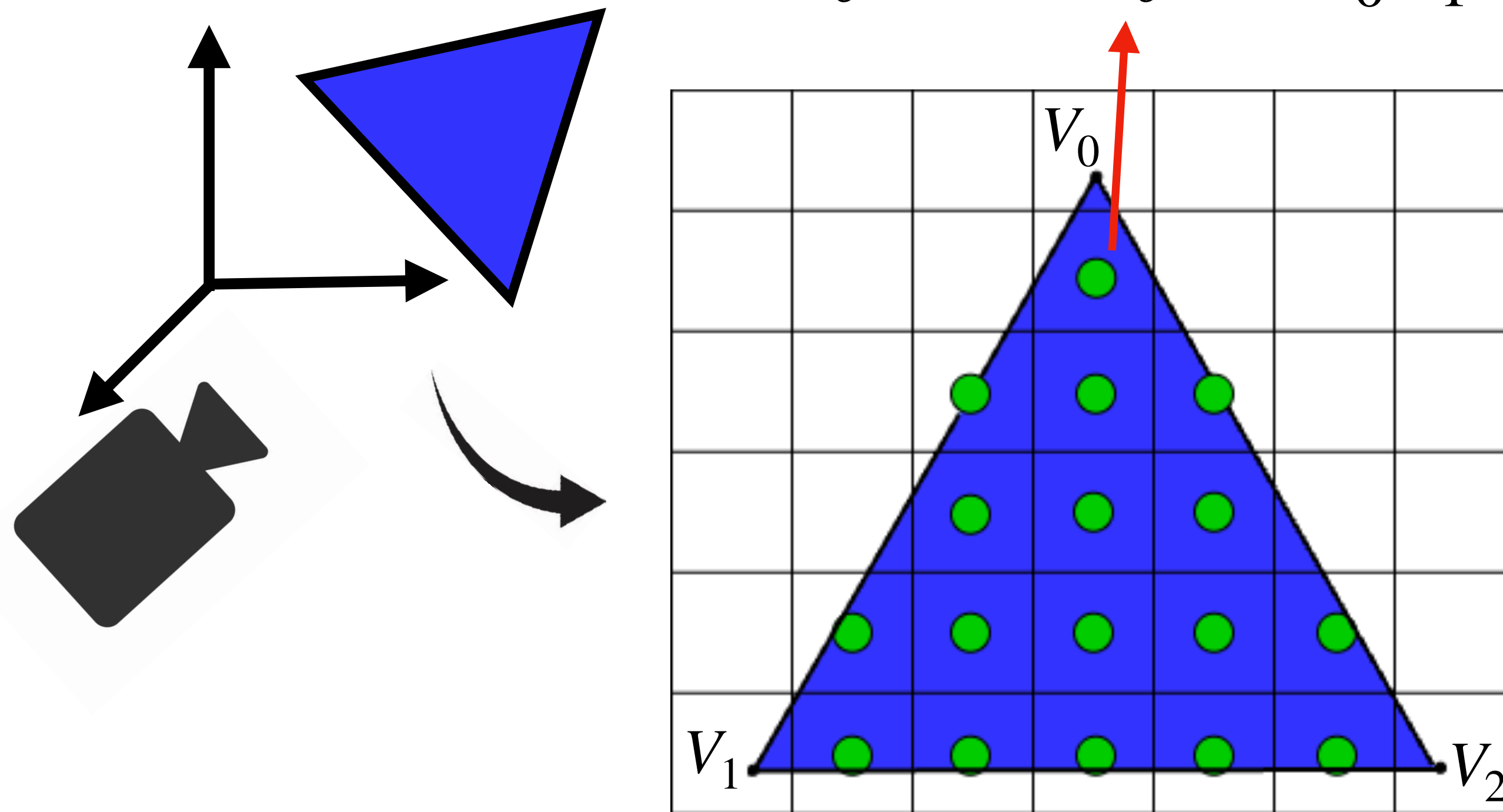


# Pixel to 3D Mapping



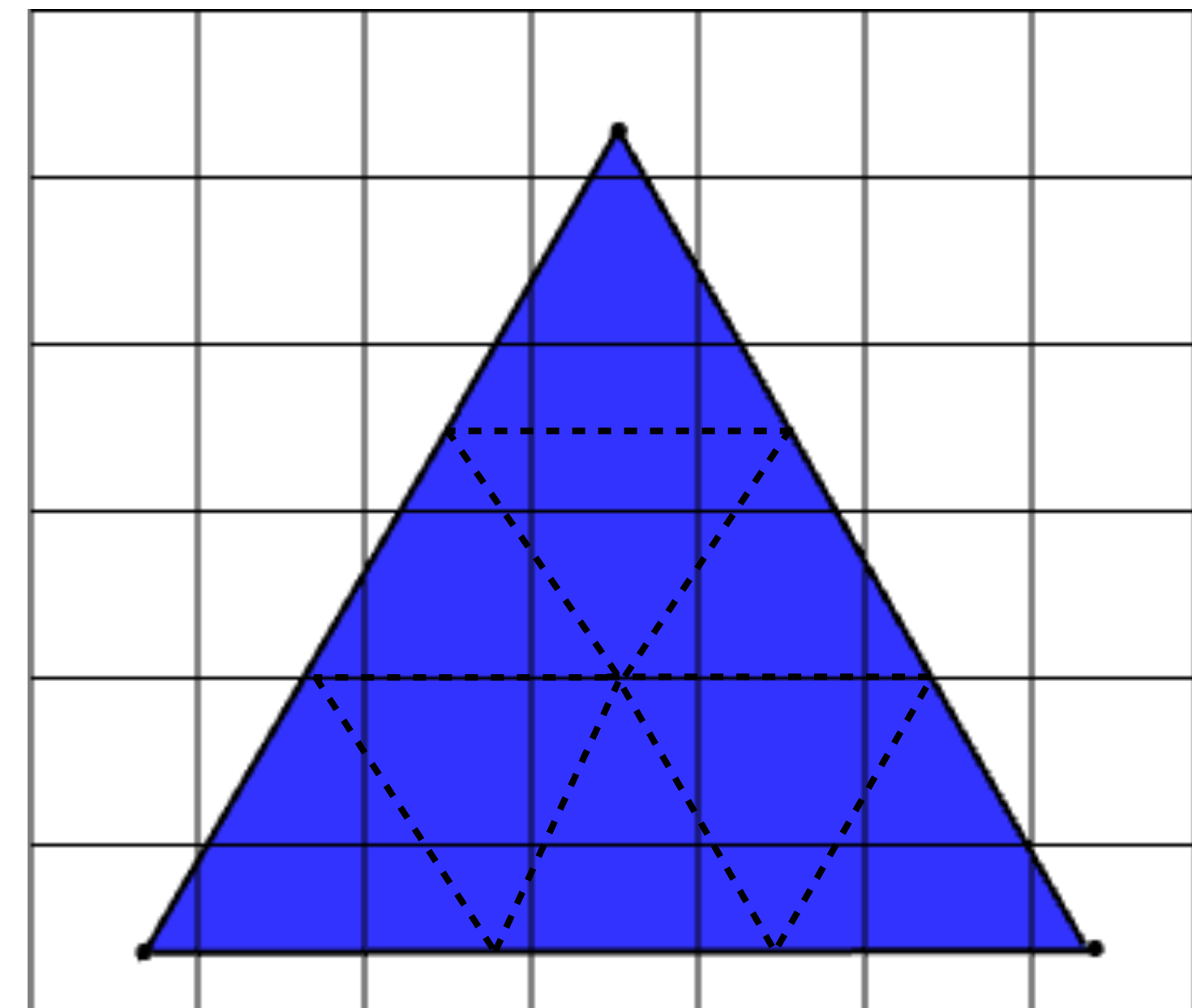
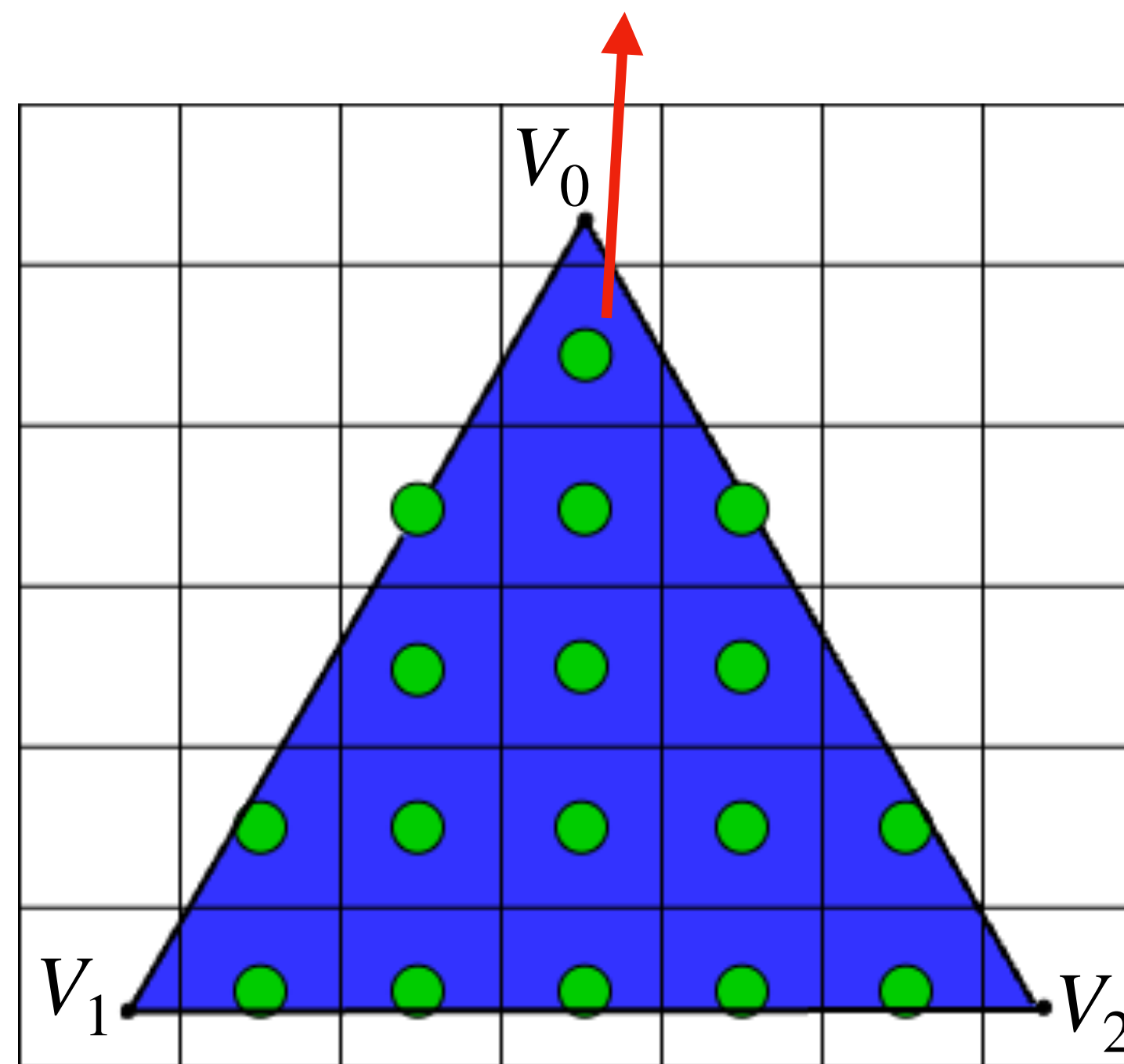
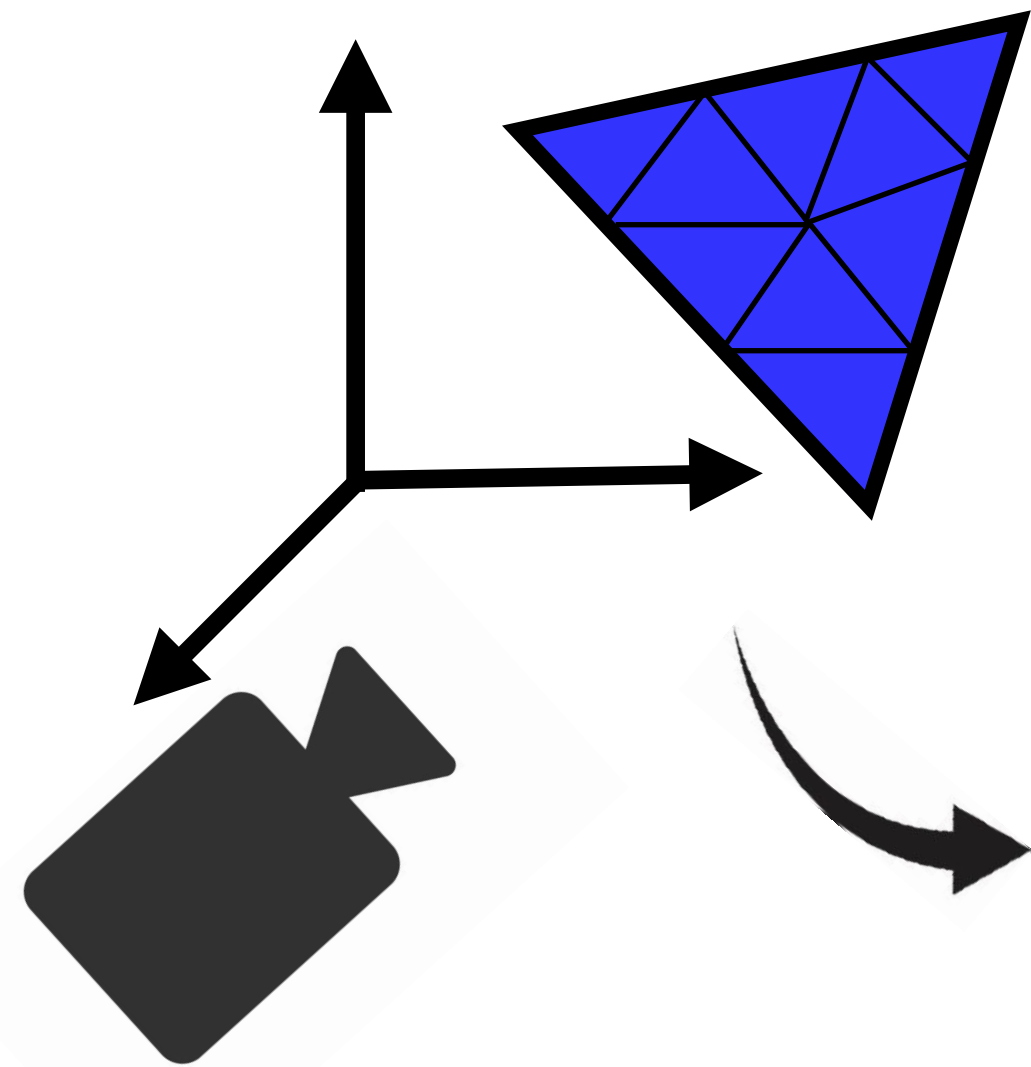
# Pixel to 3D Mapping

$$\mathbb{R}^3 \ni P(i,j) = \lambda(i,j) \cdot [V_0 V_1 V_2]$$



# Pixel to 3D Mapping

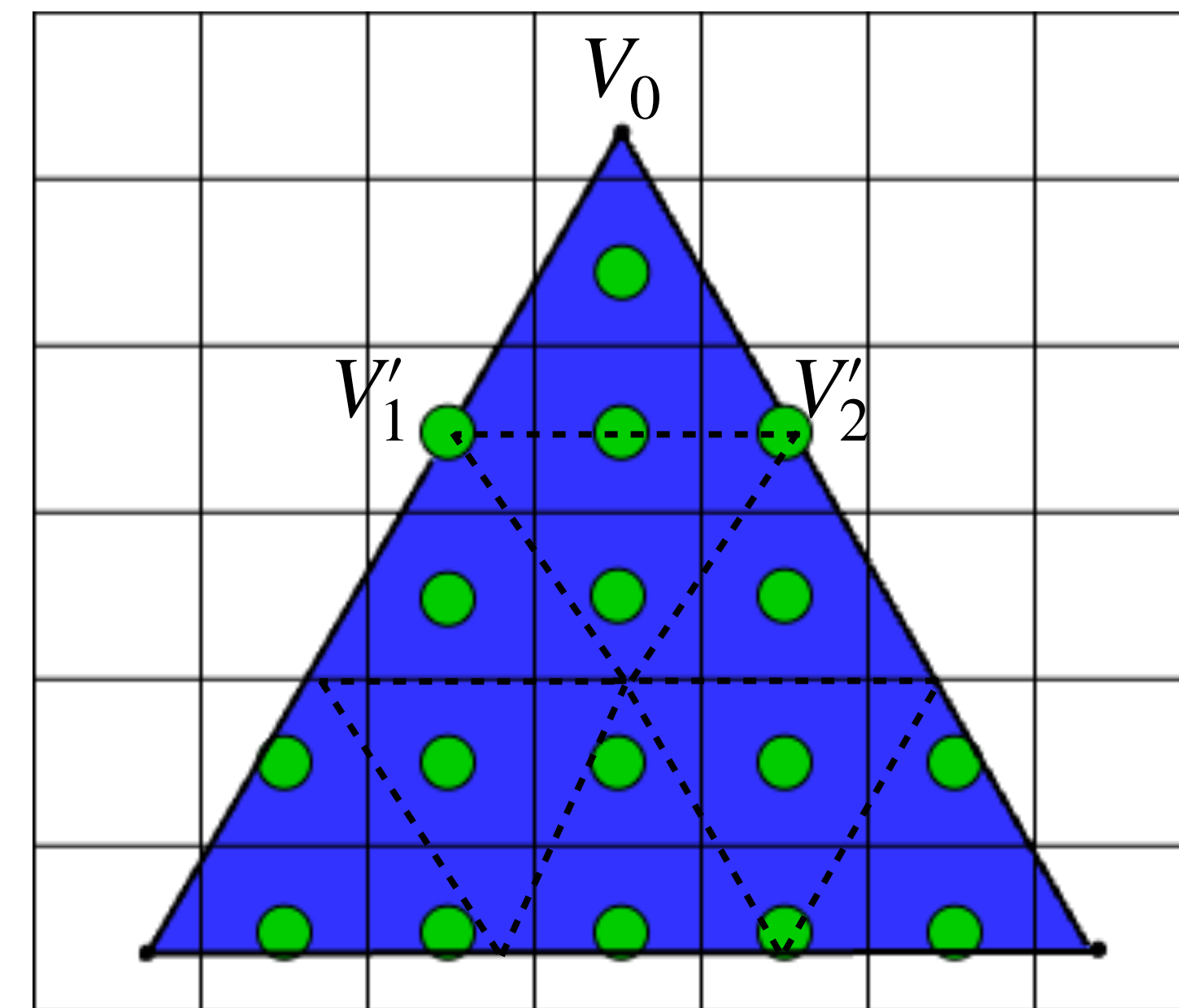
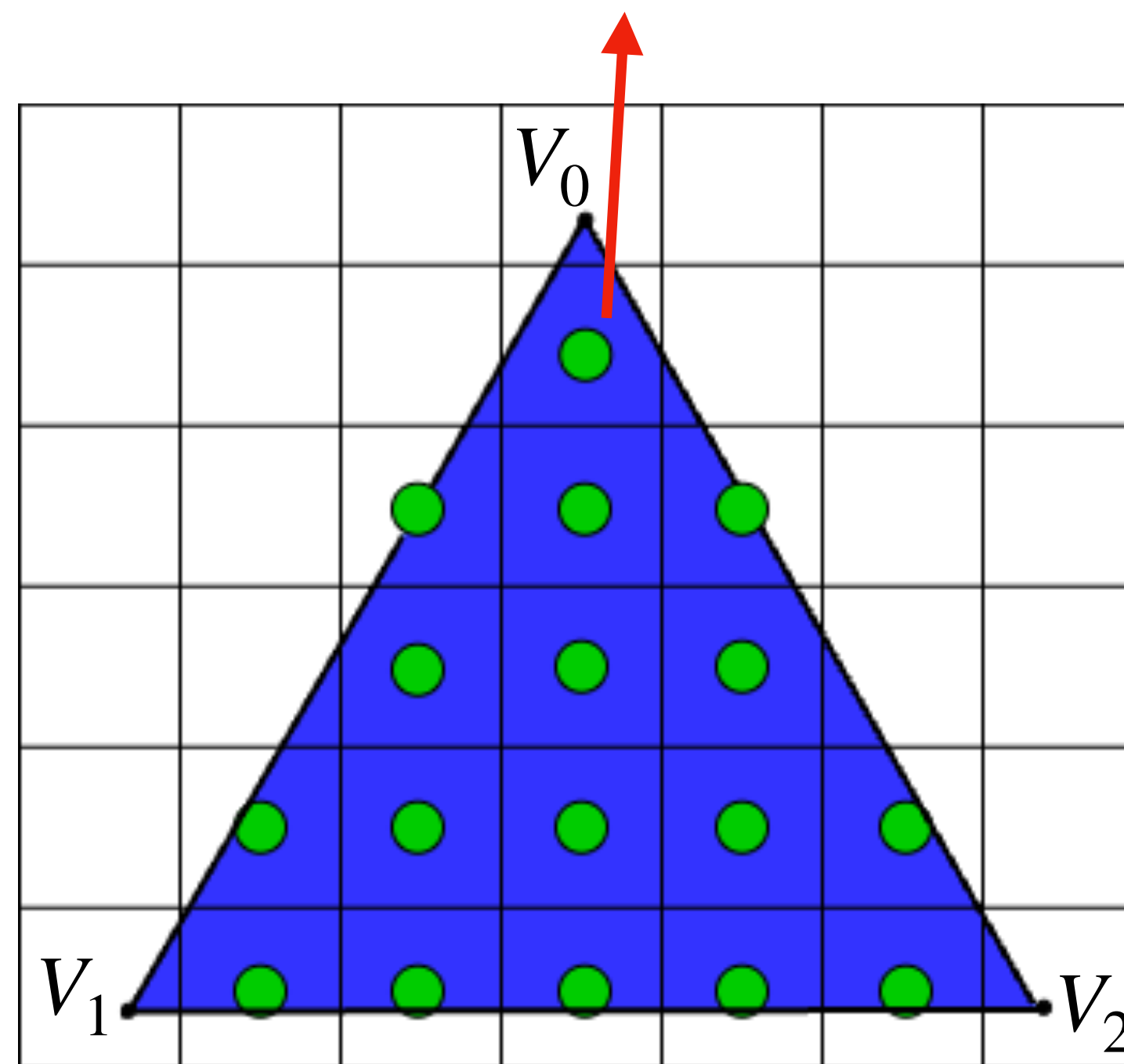
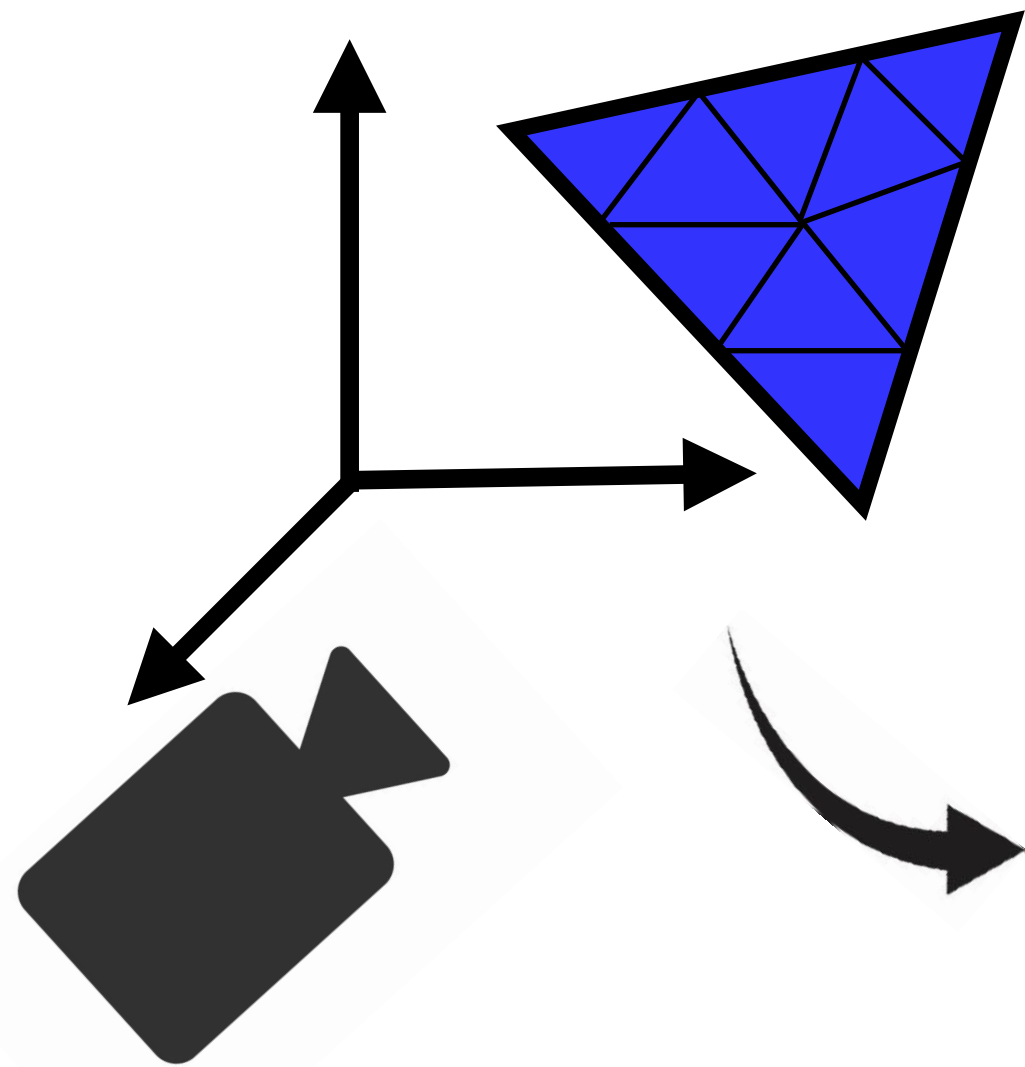
$$\mathbb{R}^3 \ni P(i,j) = \lambda(i,j) \cdot [V_0 V_1 V_2]$$





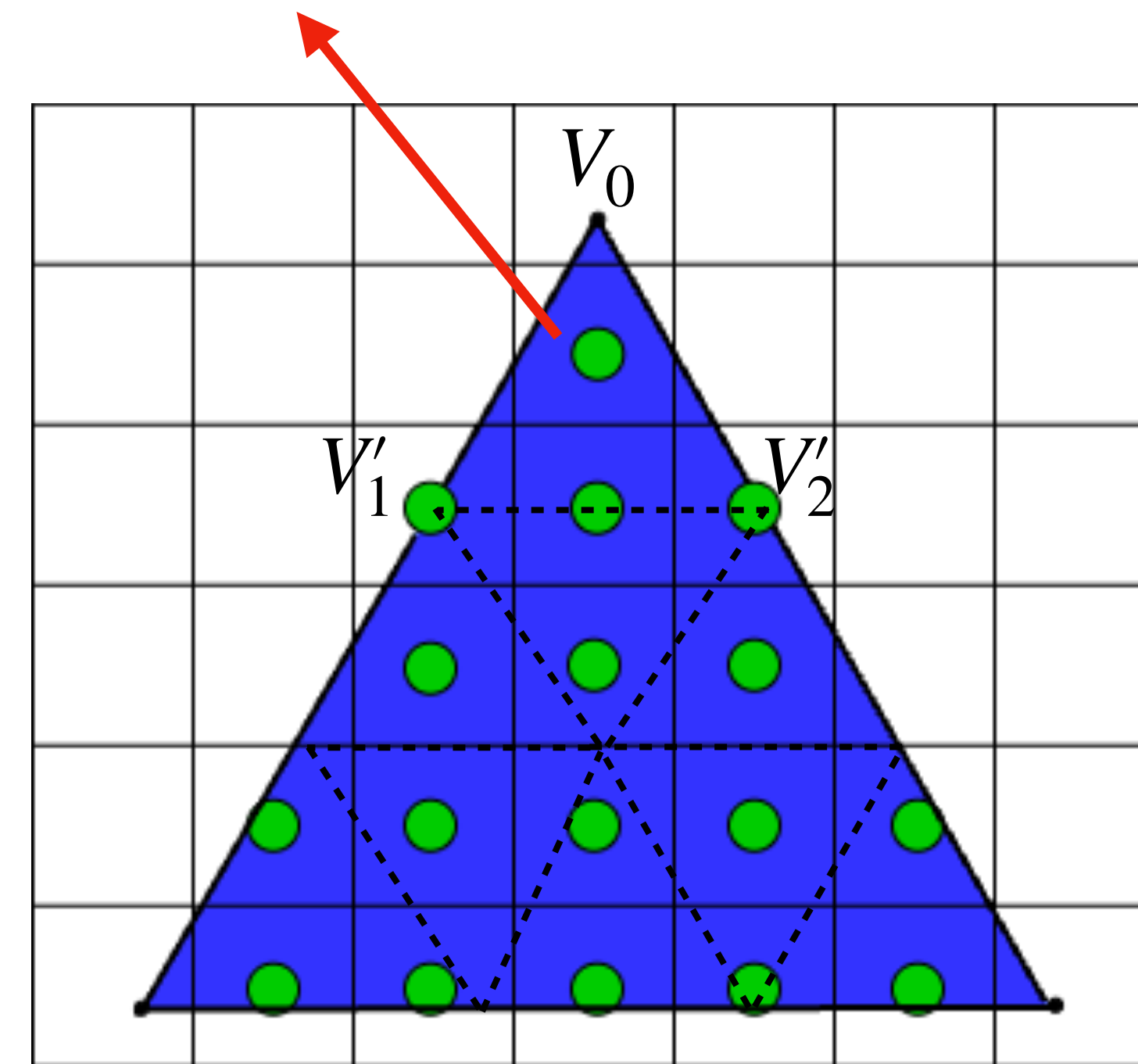
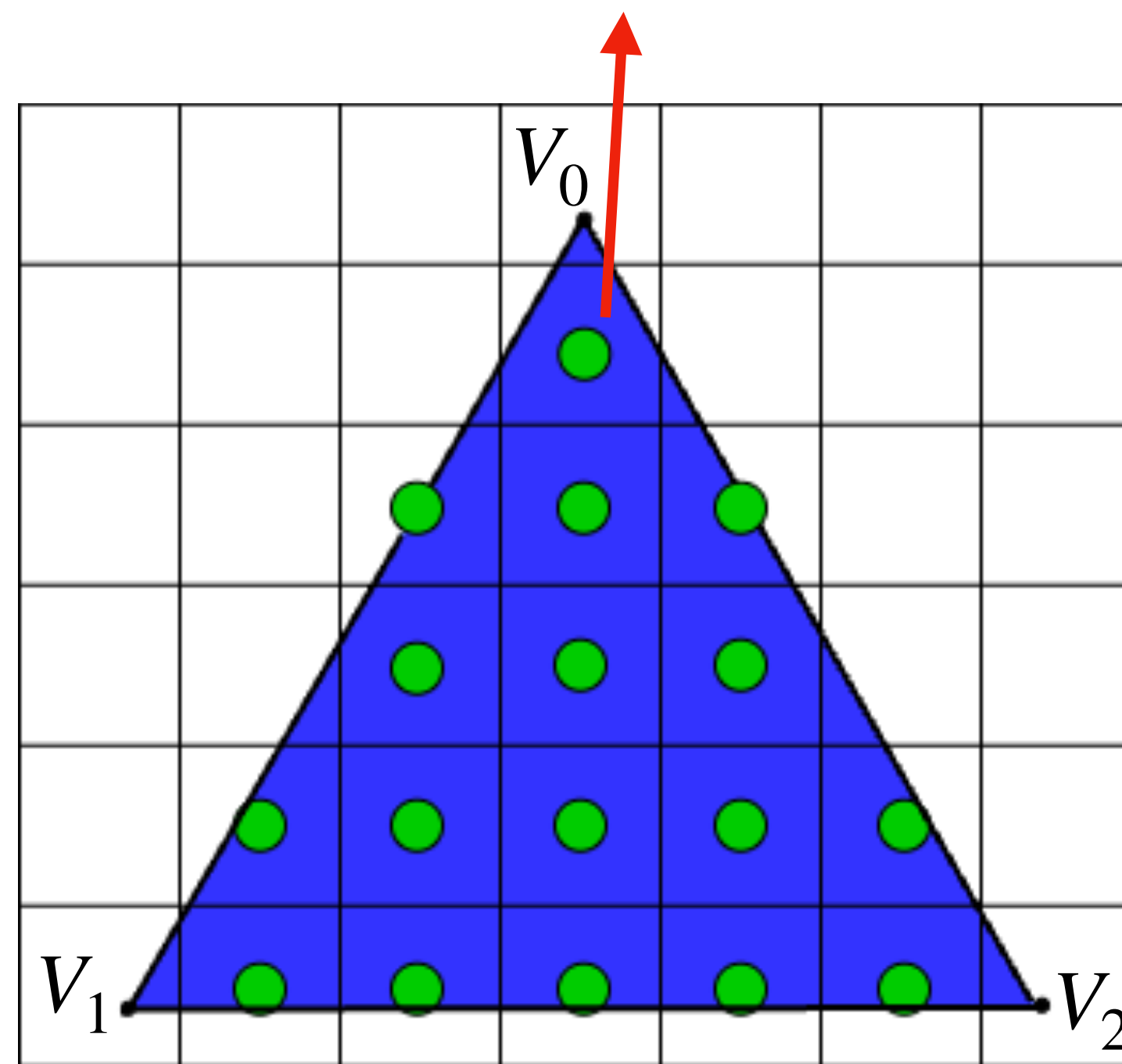
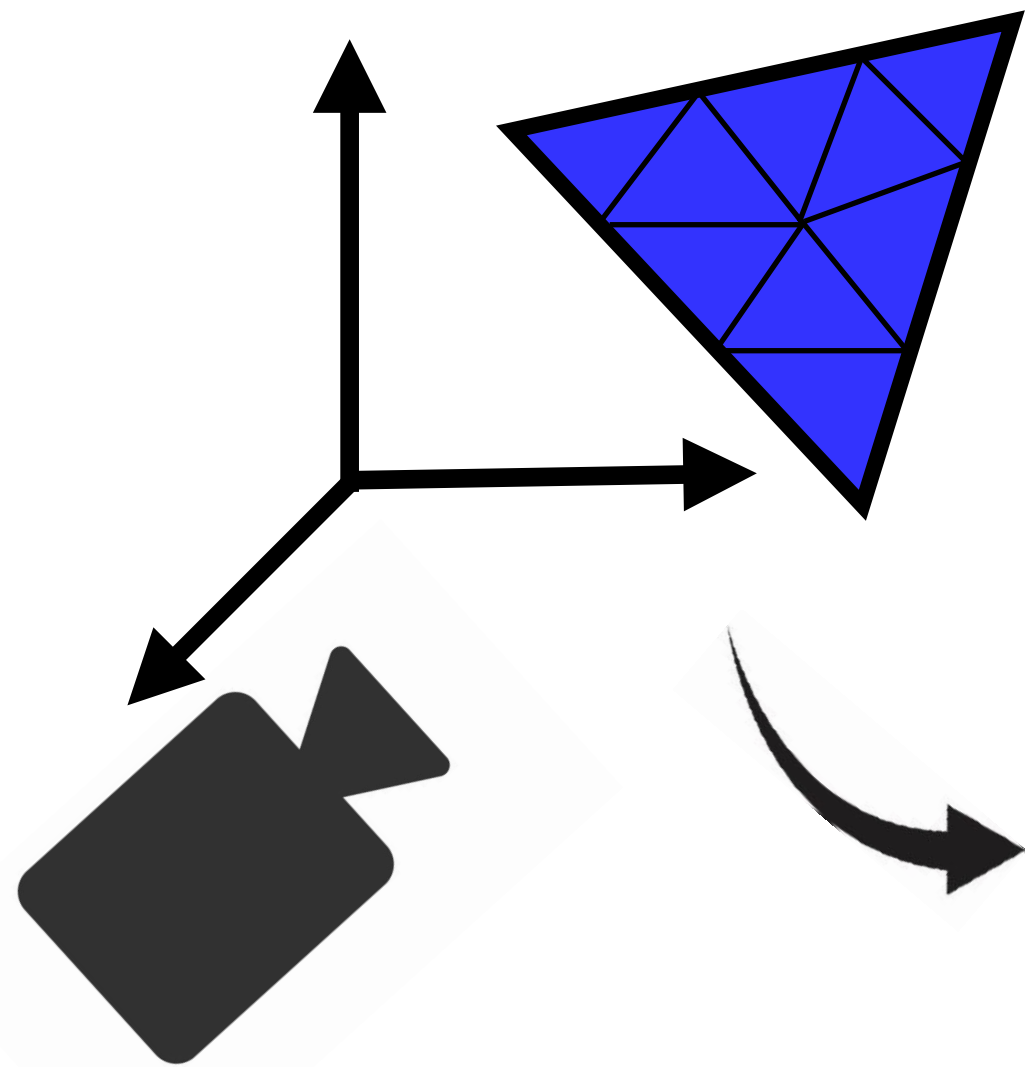
# Pixel to 3D Mapping

$$\mathbb{R}^3 \ni P(i,j) = \lambda(i,j) \cdot [V_0 V_1 V_2]$$



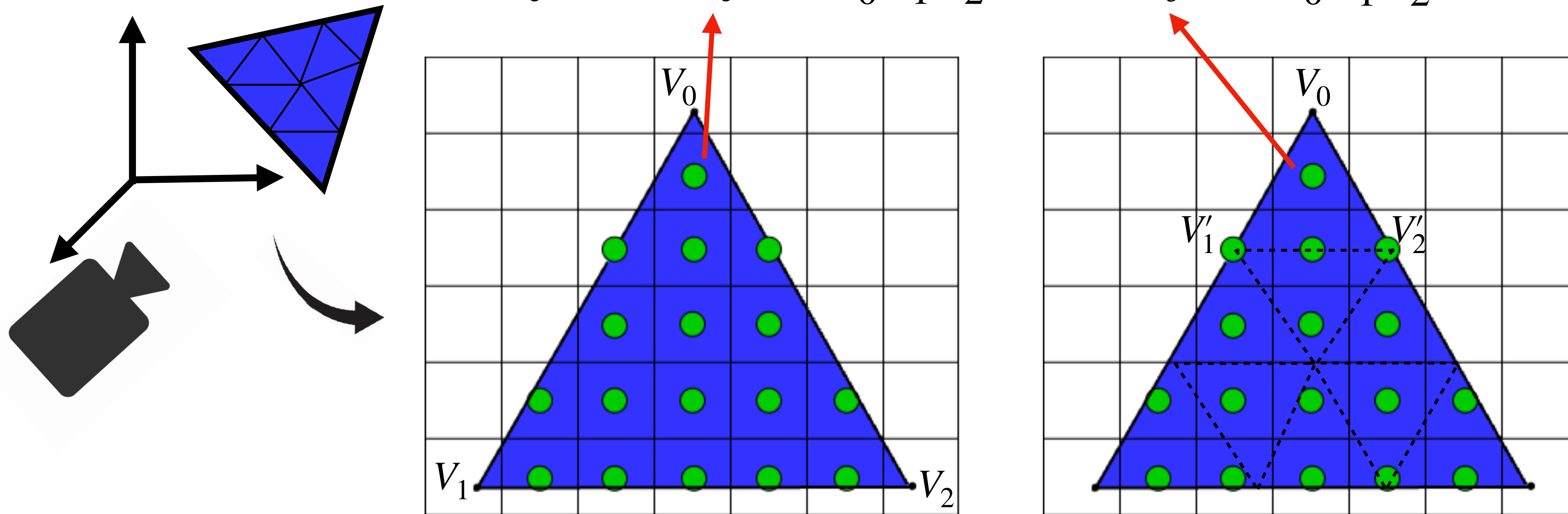
# Pixel to 3D Mapping

$$\mathbb{R}^3 \ni P(i,j) = \lambda(i,j) \cdot [V_0 V_1 V_2] = \lambda'(i,j) \cdot [V_0 V'_1 V'_2]$$



# Pixel to 3D Mapping

$$\mathbb{R}^3 \ni P(i,j) = \lambda(i,j) \cdot [V_0 V_1 V_2] = \lambda'(i,j) \cdot [V_0 V'_1 V'_2]$$

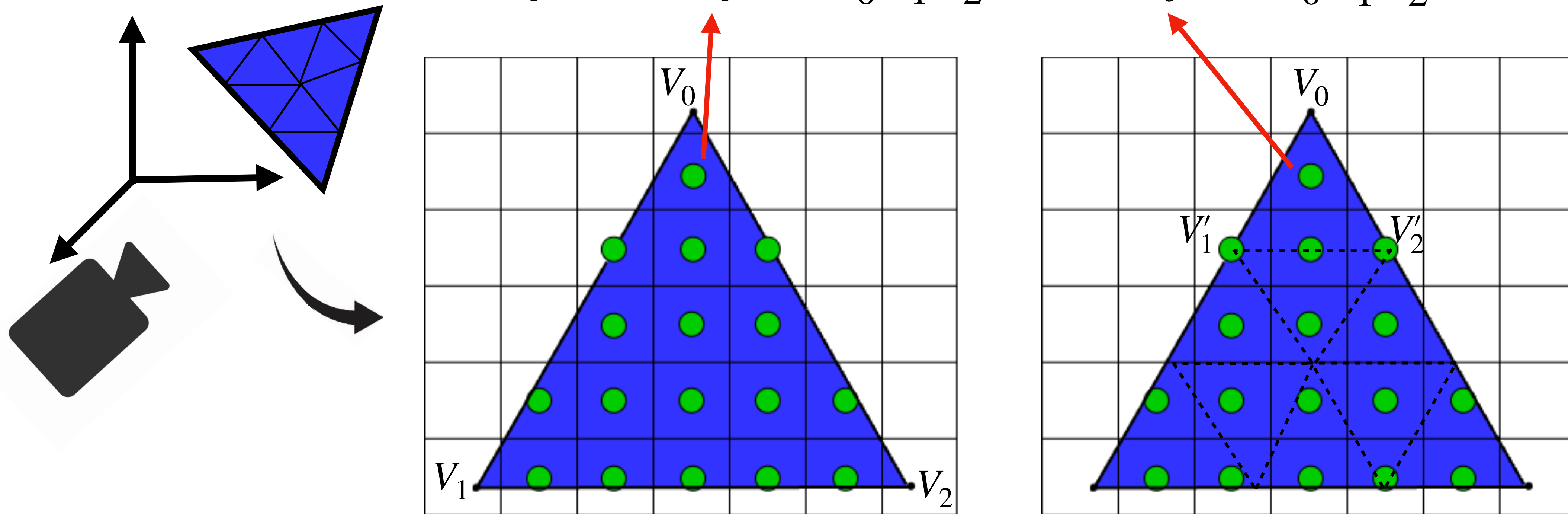


**Render is a surface resampling method**



# Pixel to 3D Mapping

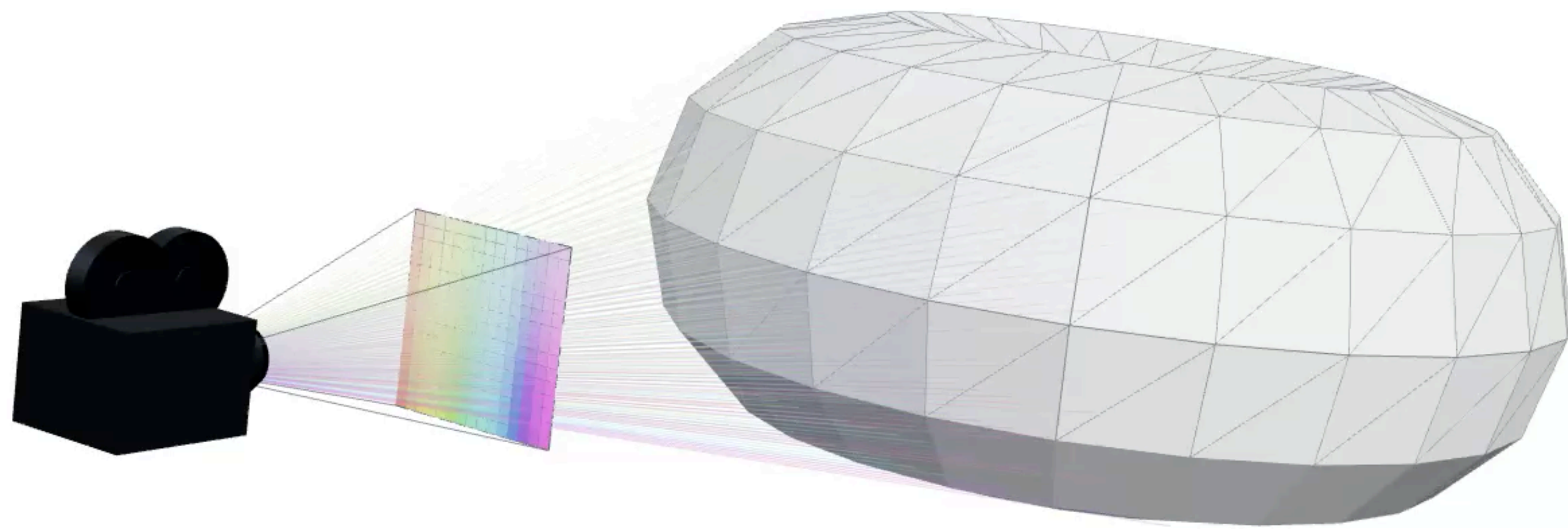
$$\mathbb{R}^3 \ni P(i,j) = \lambda(i,j) \cdot [V_0 V_1 V_2] = \lambda'(i,j) \cdot [V_0 V'_1 V'_2]$$



**Render is a surface resampling method**

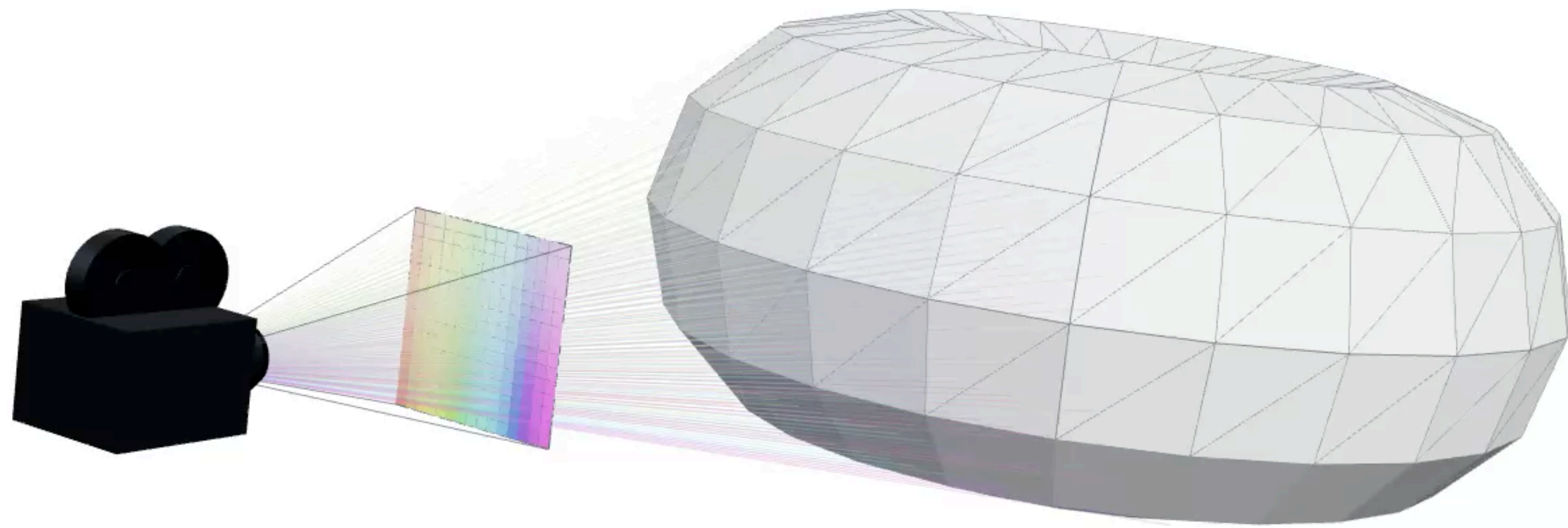
**# samples determined by render resolution, NOT surface**

# Pixel to 3D Mapping



**Barycentric coordinates from rasterization  
give pixel to 3D mapping for free**

# Pixel to 3D Mapping



**Barycentric coordinates from rasterization  
give pixel to 3D mapping for free**

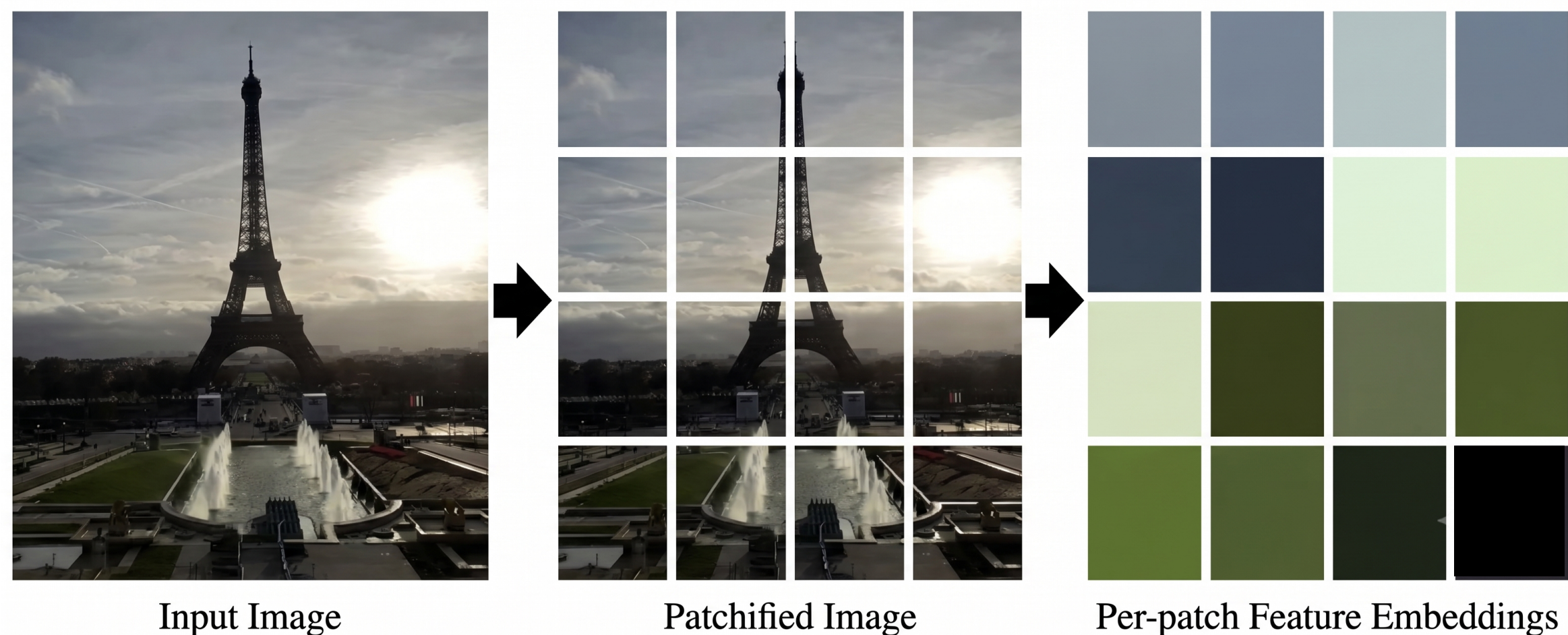


**What if you have features not at the pixel level?**

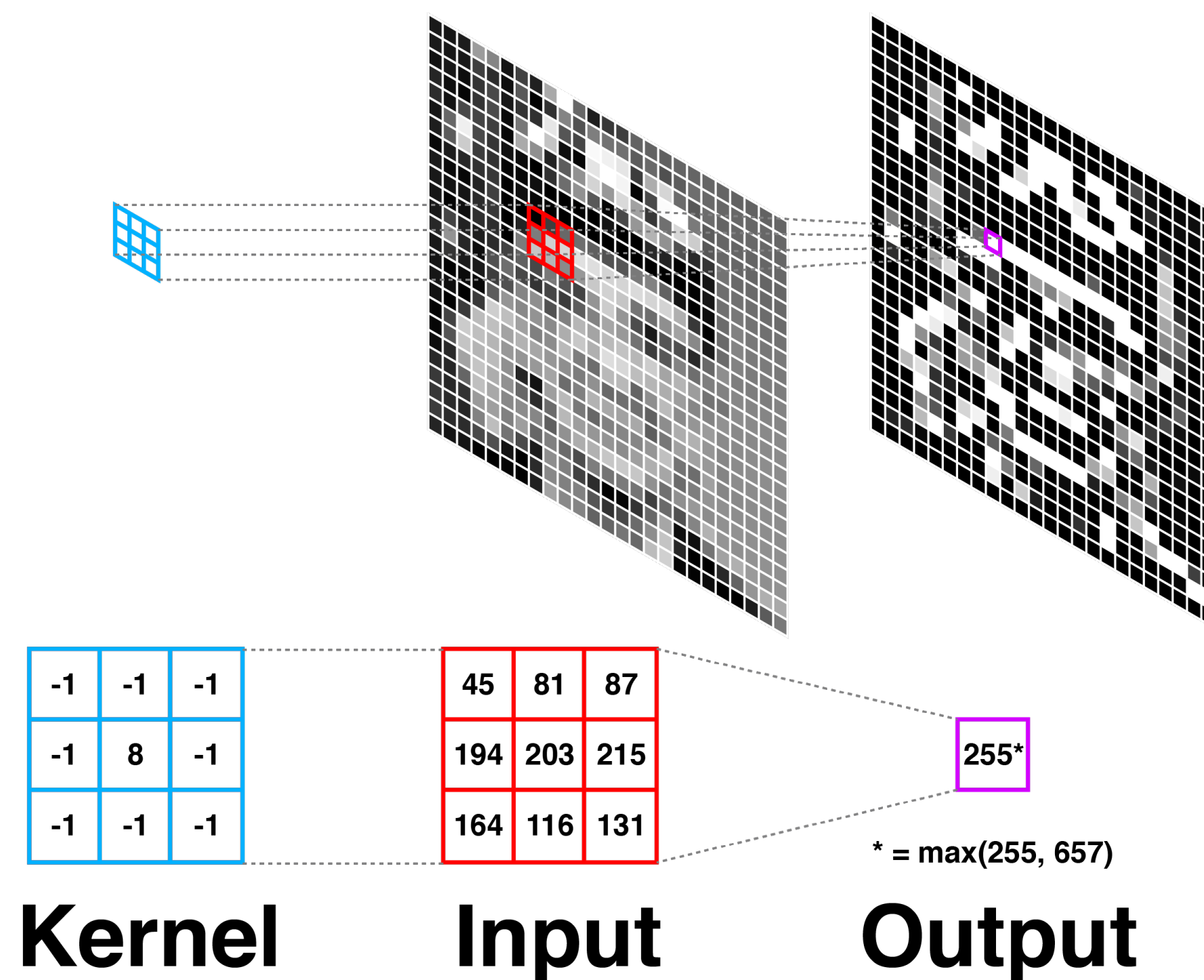
# Image Feature Aliasing Problem

Most modern image encoders do not give pixel-level features

*Patchification*



*Convolution*

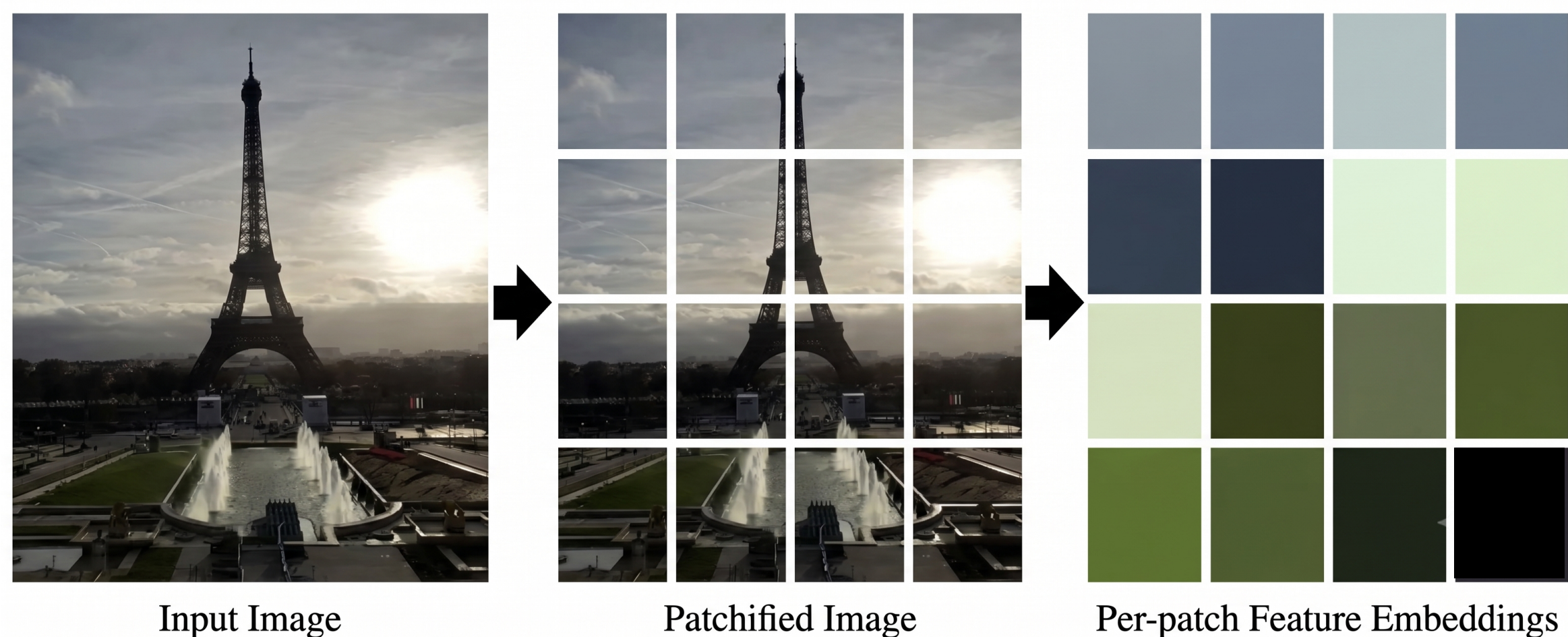




# Image Feature Aliasing Problem

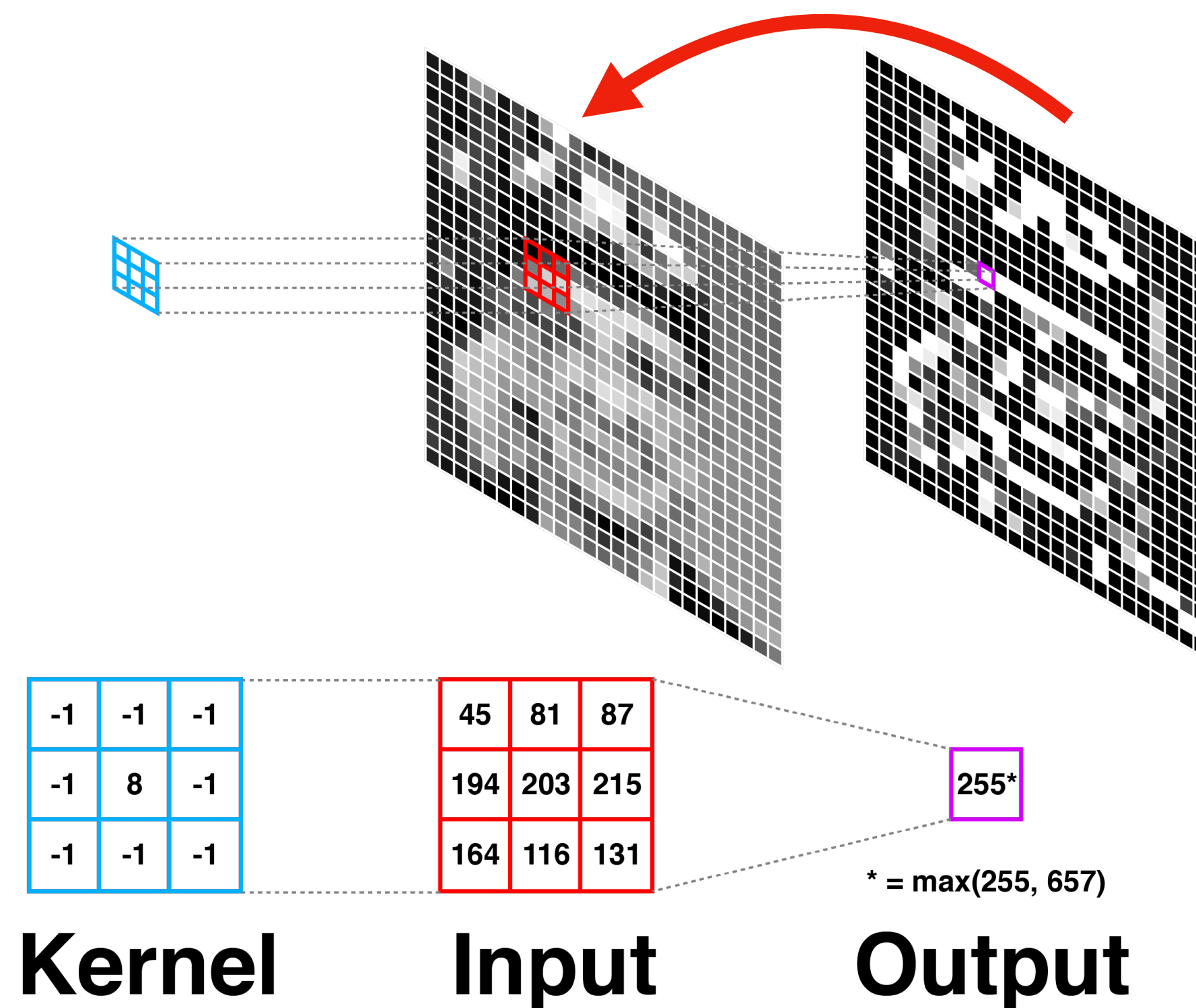
Most modern image encoders do not give pixel-level features

*Patchification*



Aliasing occurs in the upsampling

*Convolution*





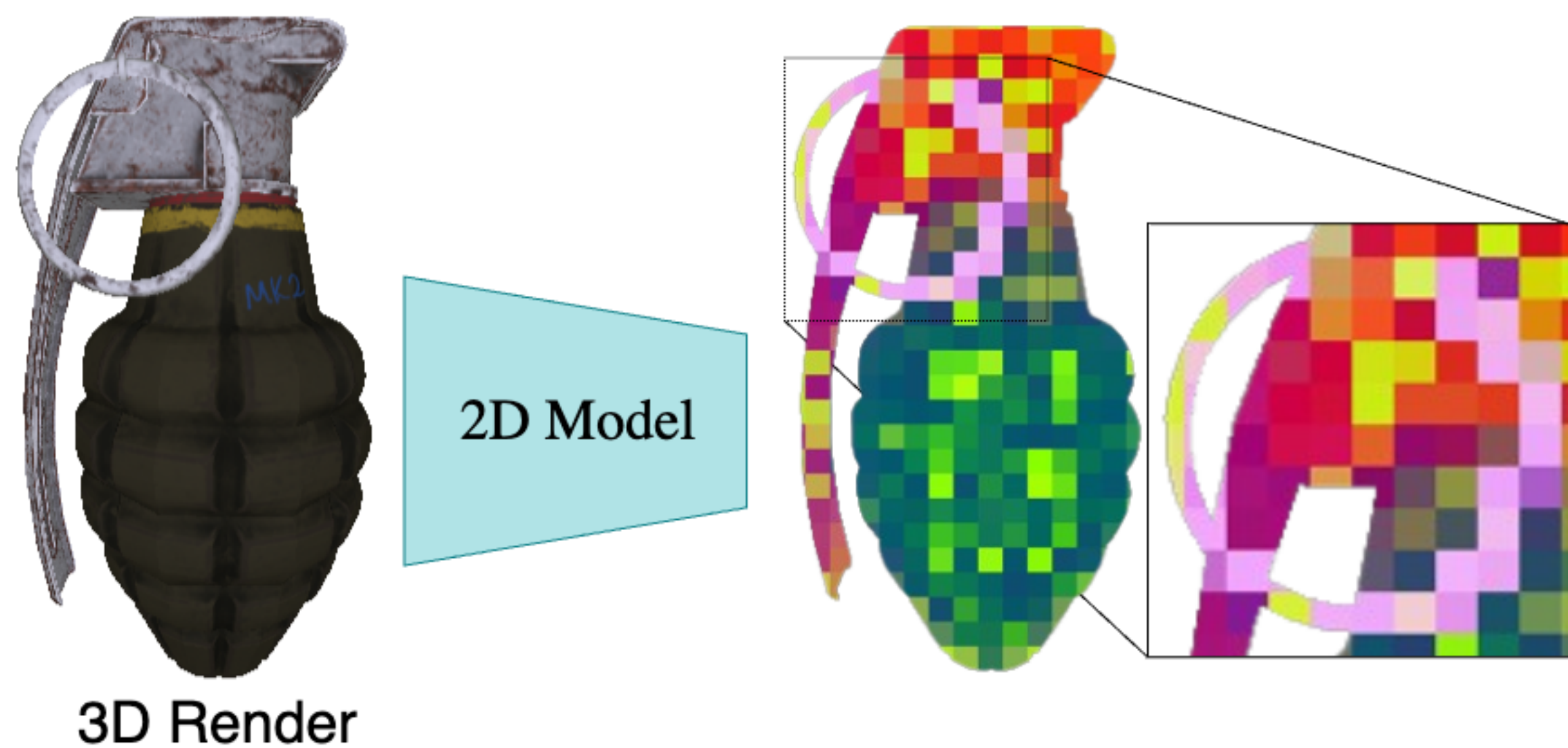
# Segmentation-Based Feature Antialiasing



3D Render

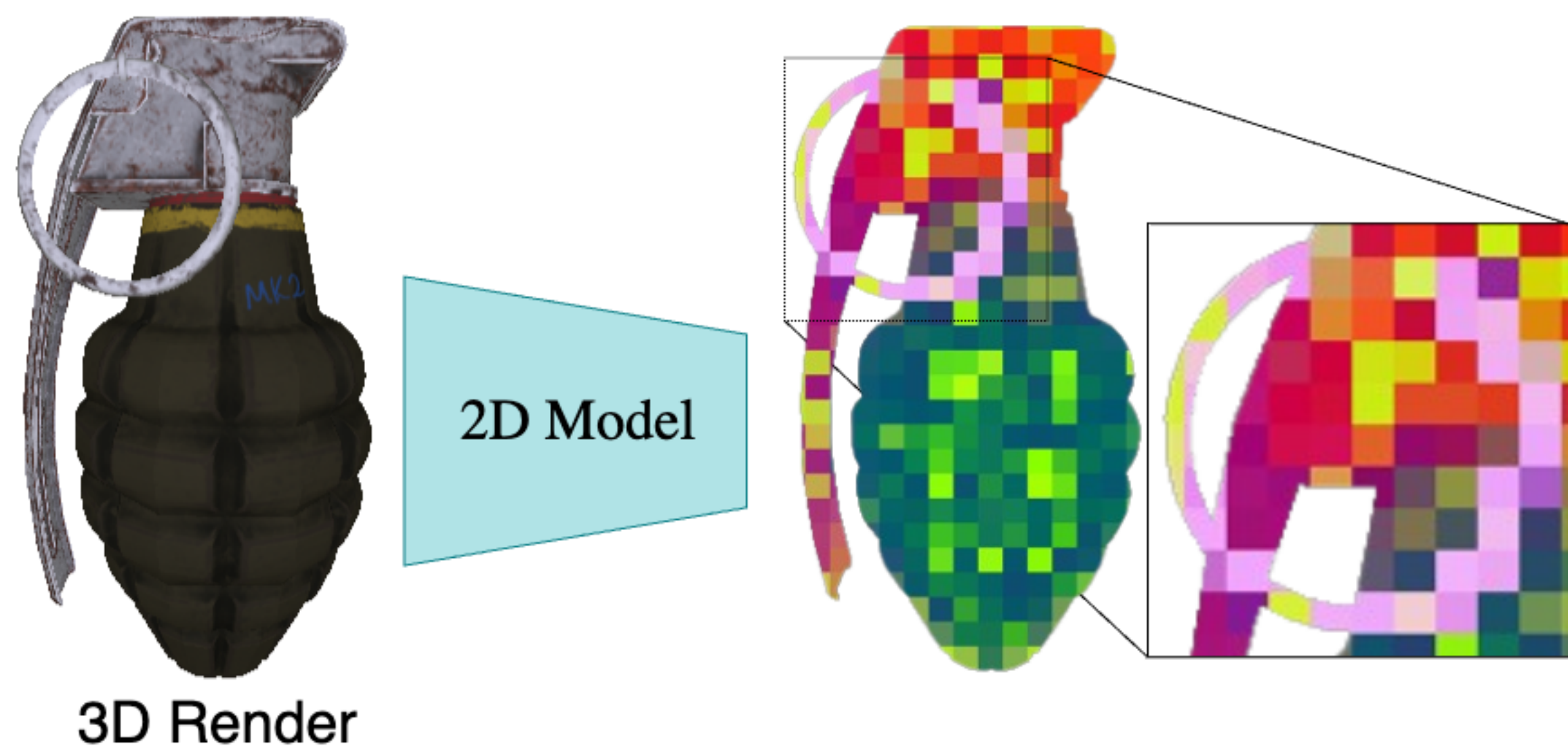
**“MeshFM: 2D Features Are All You  
Need for 3D Shape Understanding”**  
Zhou et al. 2026

# Segmentation-Based Feature Antialiasing



**“MeshFM: 2D Features Are All You Need for 3D Shape Understanding”**  
Zhou et al. 2026

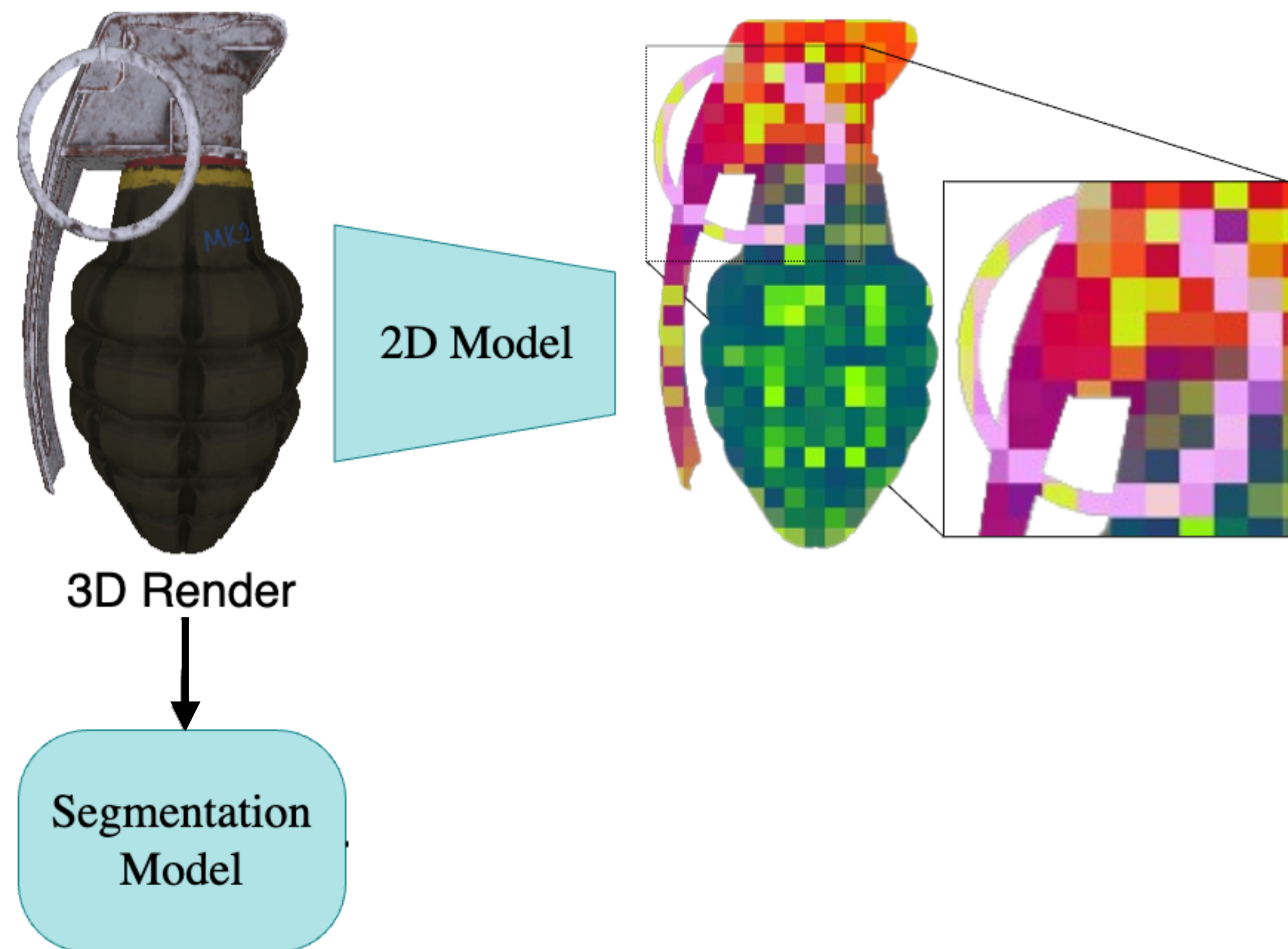
# Segmentation-Based Feature Antialiasing



**“MeshFM: 2D Features Are All You Need for 3D Shape Understanding”**  
Zhou et al. 2026

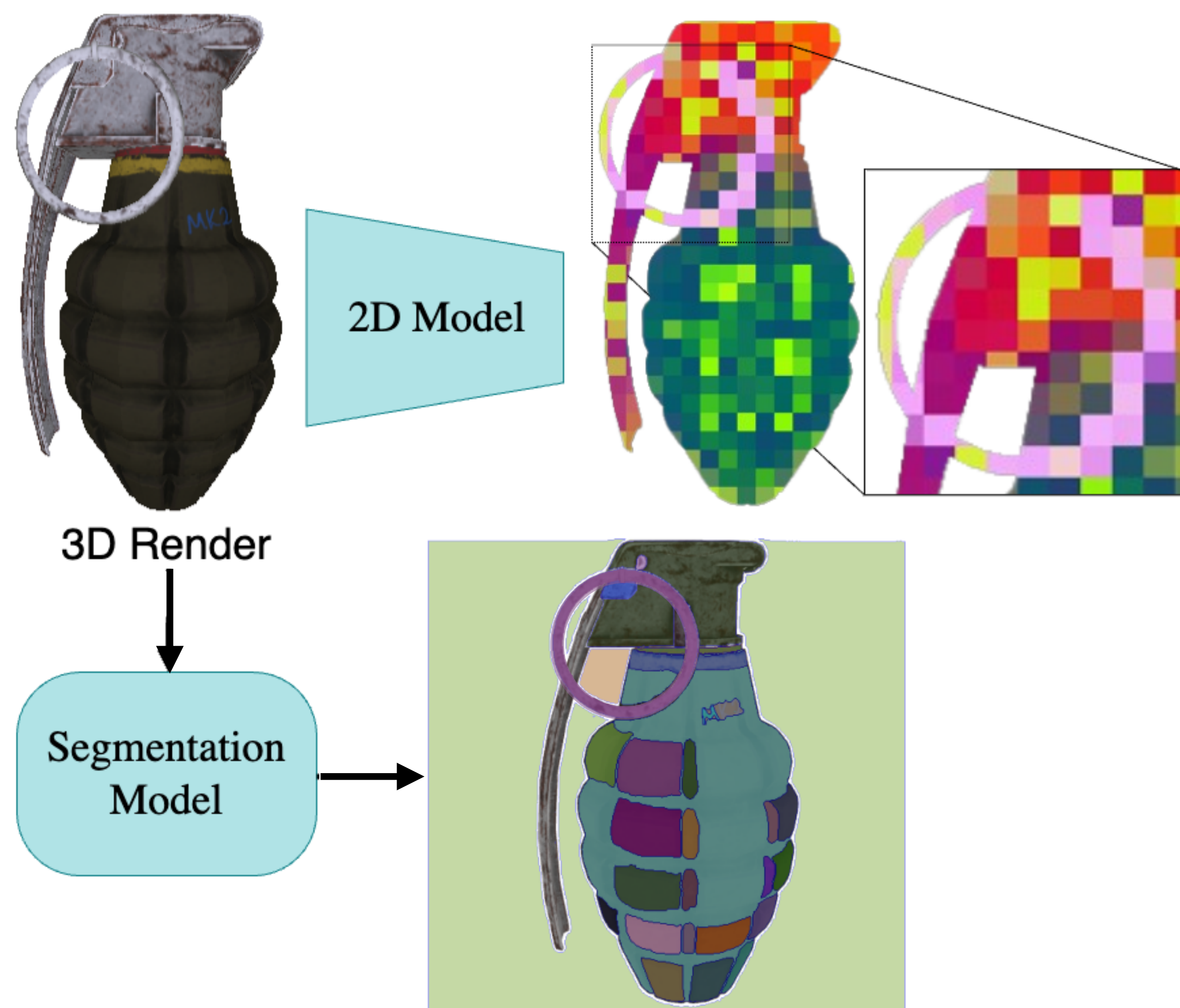


# Segmentation-Based Feature Antialiasing



**“MeshFM: 2D Features Are All You Need for 3D Shape Understanding”**  
Zhou et al. 2026

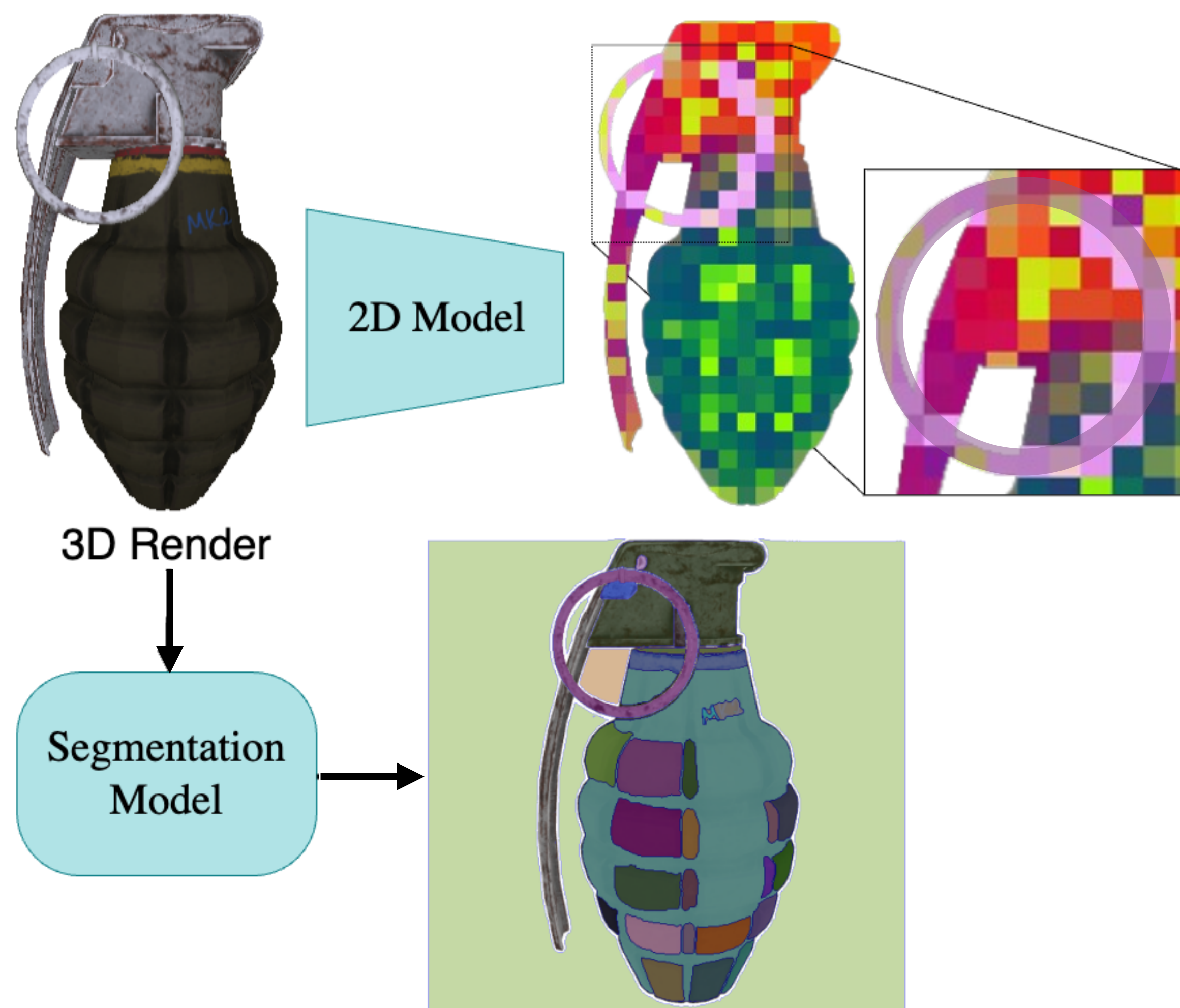
# Segmentation-Based Feature Antialiasing



**“MeshFM: 2D Features Are All You Need for 3D Shape Understanding”**  
Zhou et al. 2026



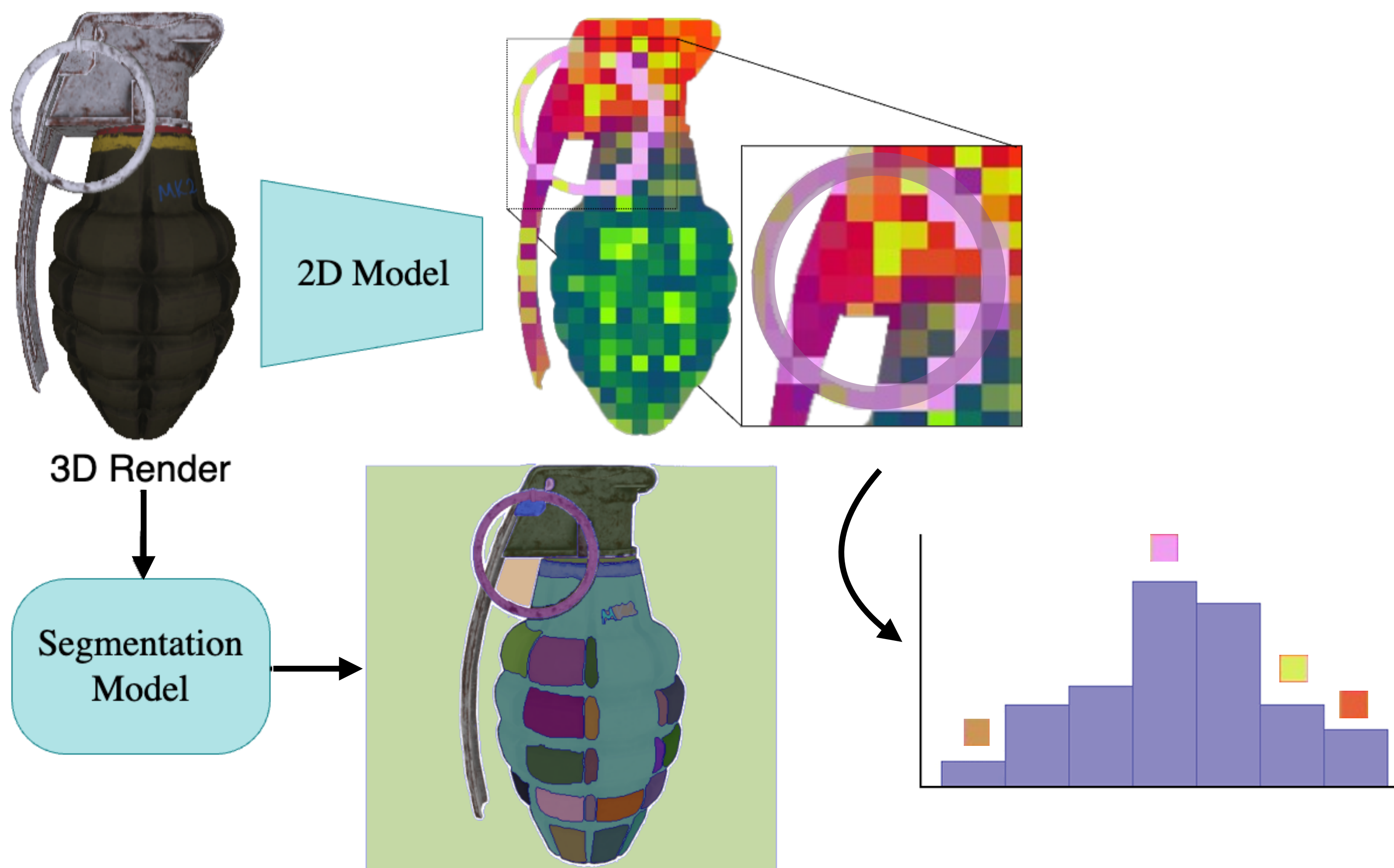
# Segmentation-Based Feature Antialiasing



**“MeshFM: 2D Features Are All You Need for 3D Shape Understanding”**  
Zhou et al. 2026

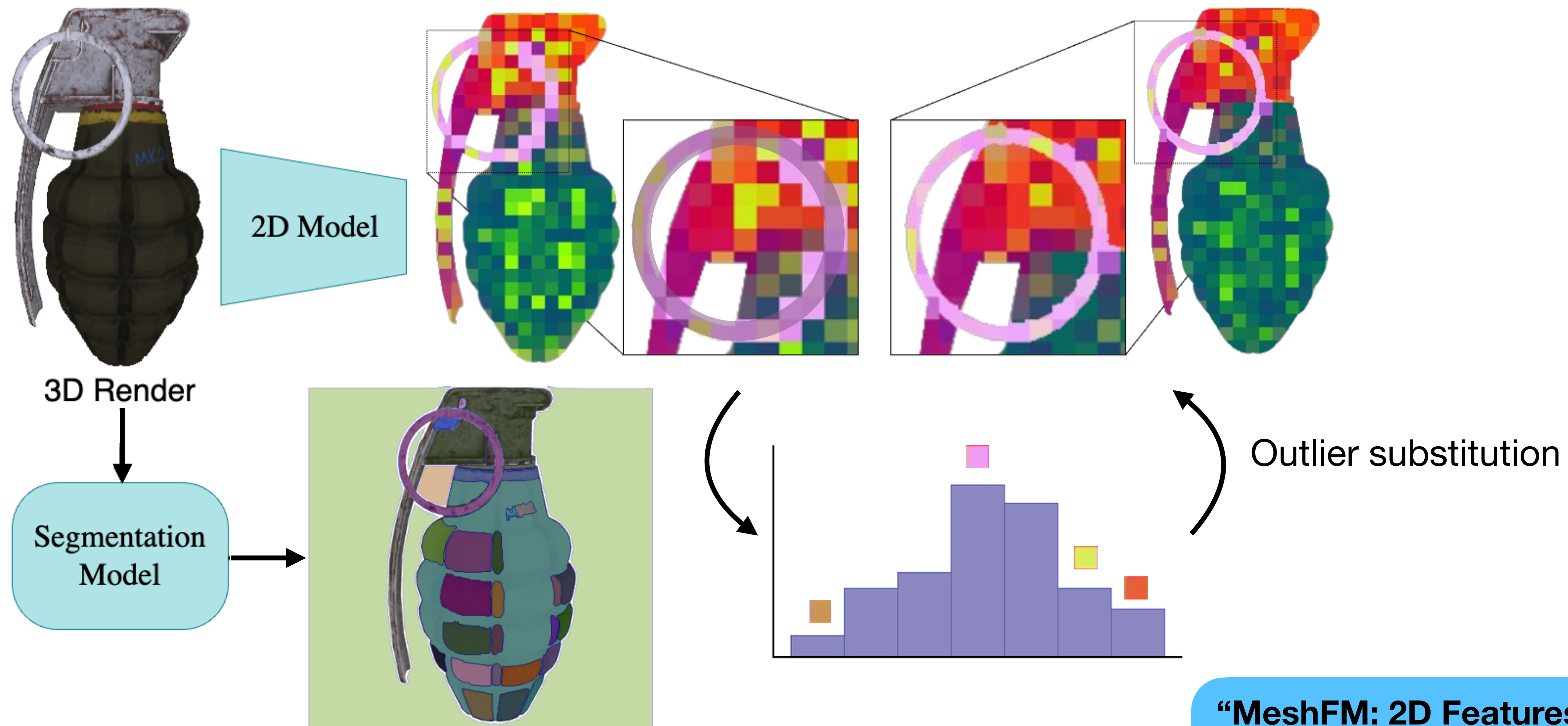


# Segmentation-Based Feature Antialiasing



**“MeshFM: 2D Features Are All You Need for 3D Shape Understanding”**  
Zhou et al. 2026

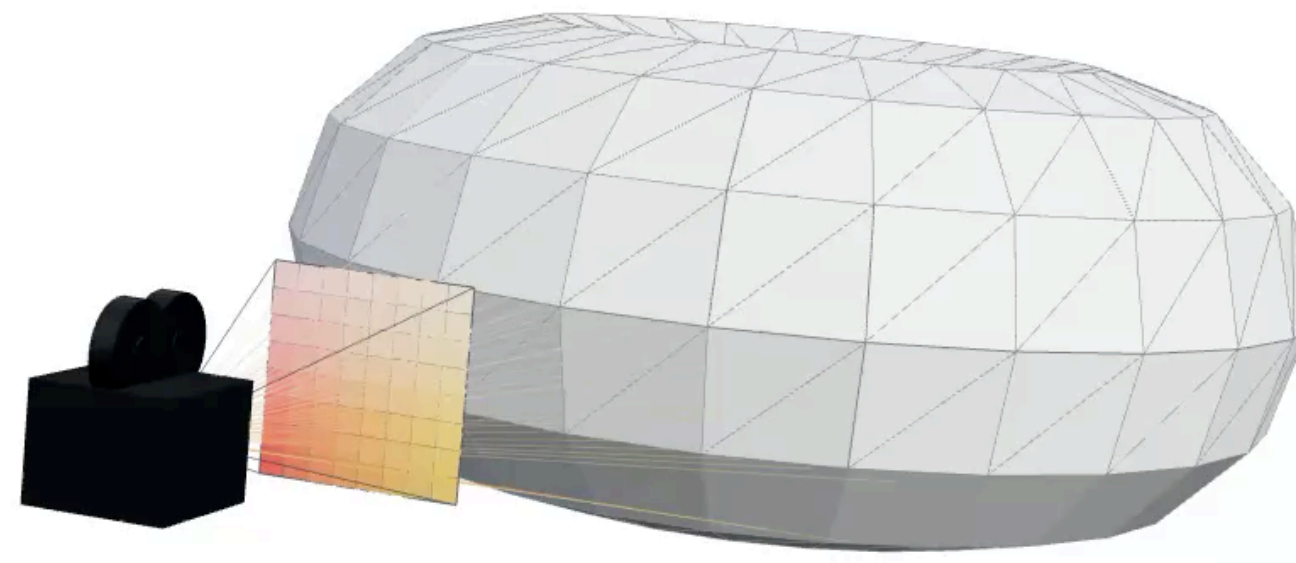
# Segmentation-Based Feature Antialiasing



**“MeshFM: 2D Features Are All You Need for 3D Shape Understanding”**  
Zhou et al. 2026



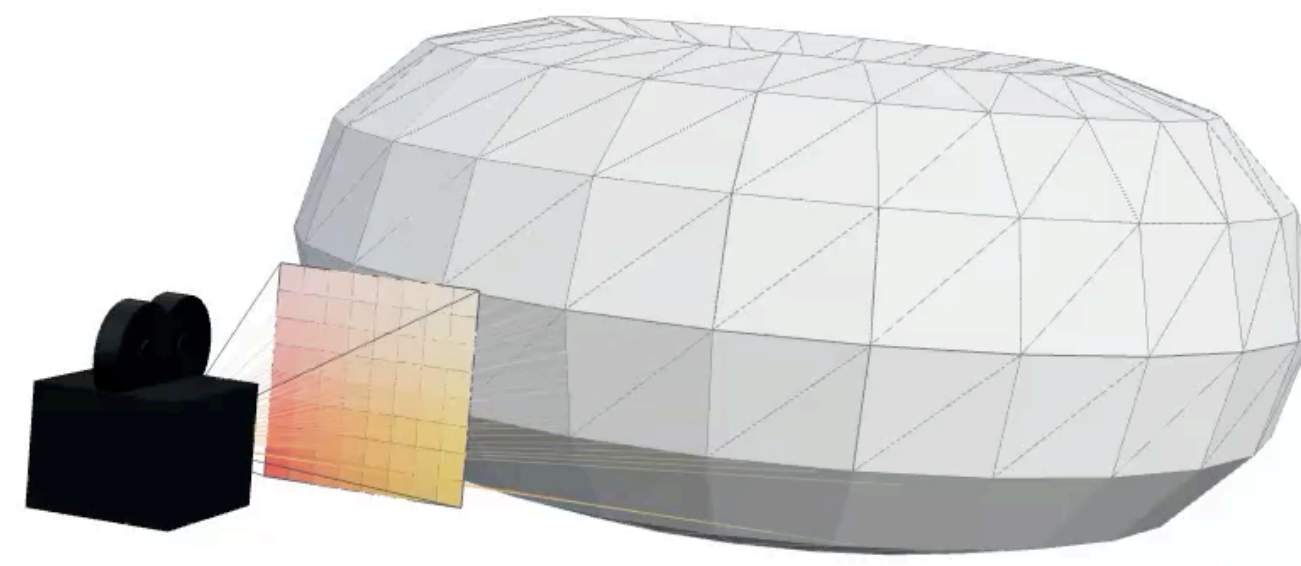
# Multi-view Backprojection



**How to aggregate inconsistent signal from multiple views?**

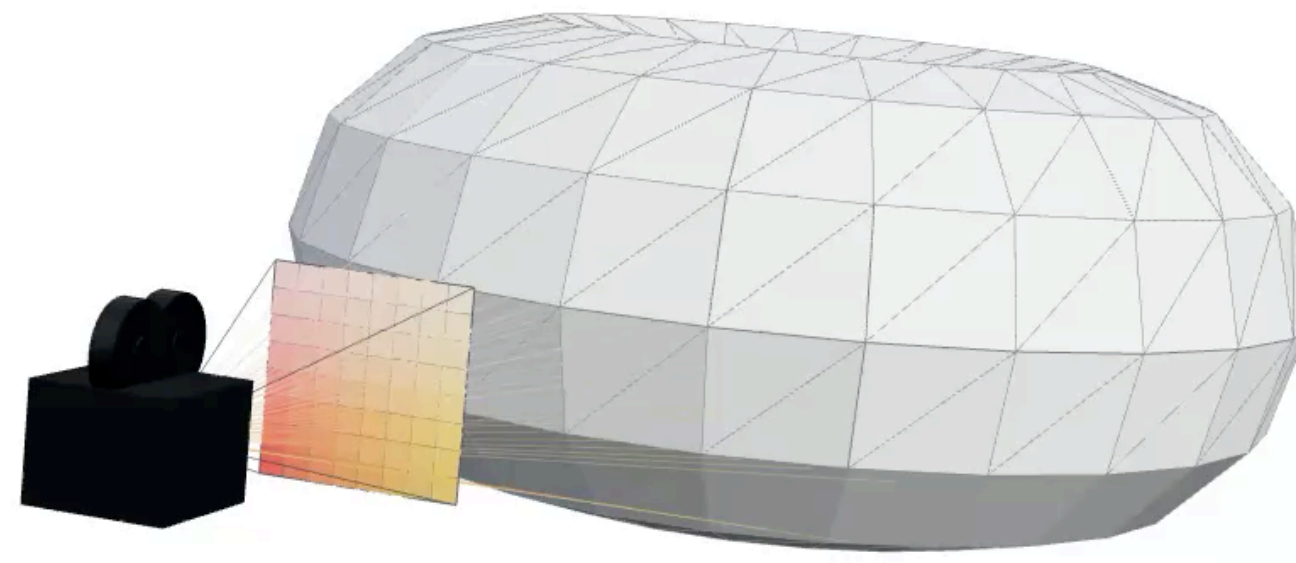


# Multi-view Backprojection

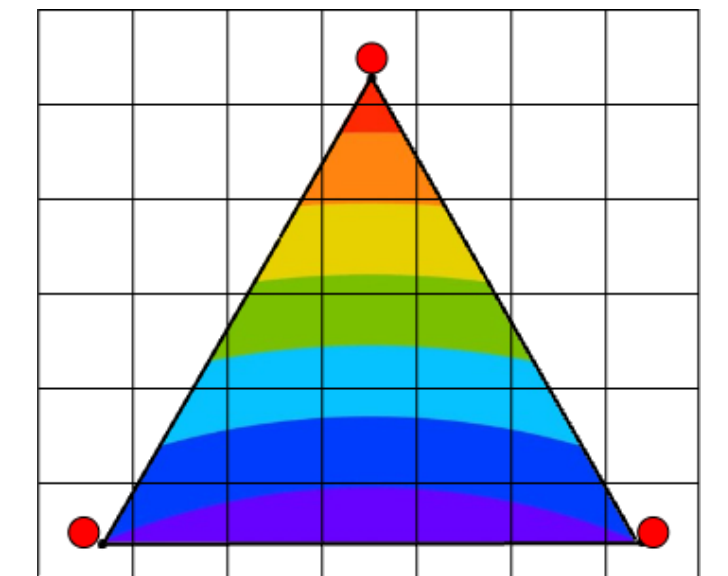


**How to aggregate inconsistent signal from multiple views?**

# Multi-view Backprojection

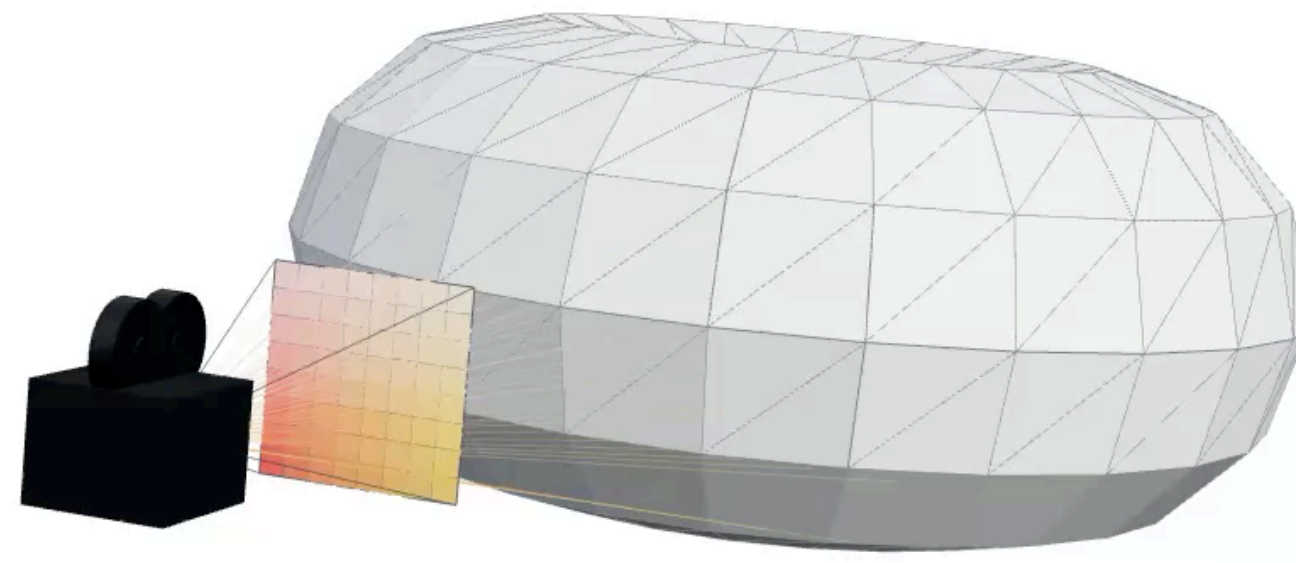


**Vertices**

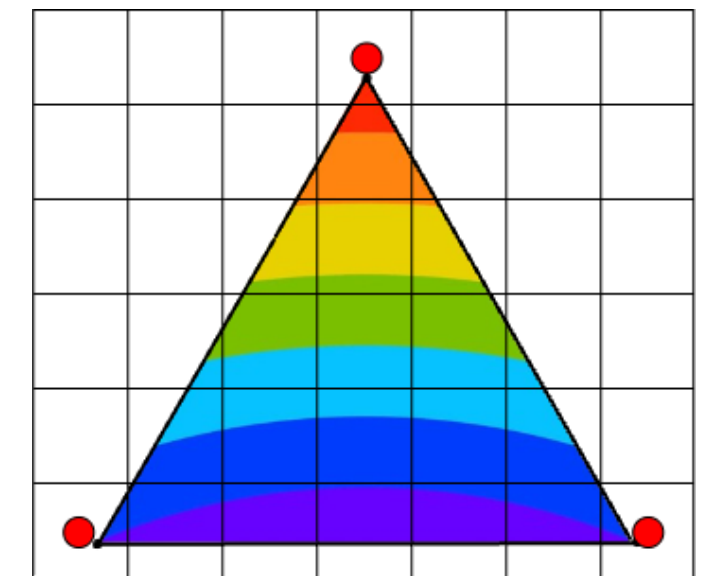


**How to aggregate inconsistent signal from multiple views?**

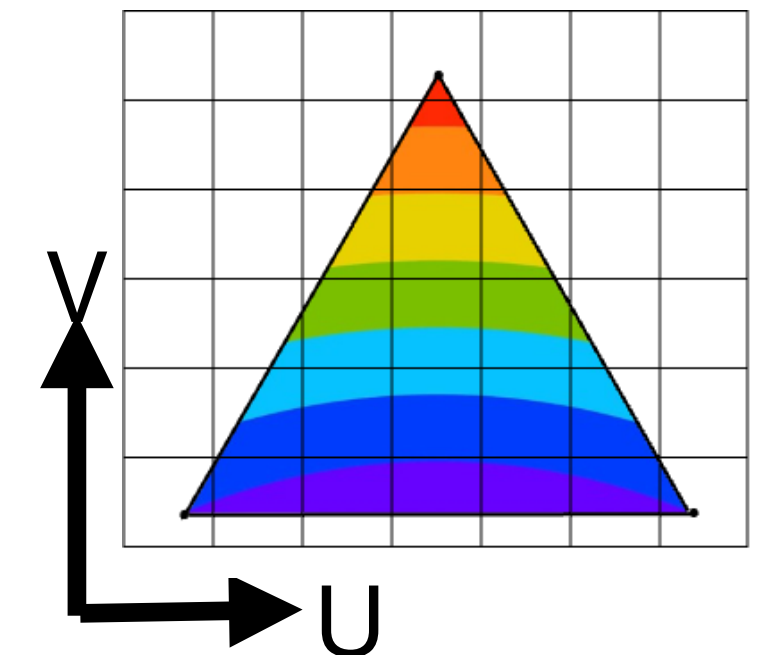
# Multi-view Backprojection



Vertices



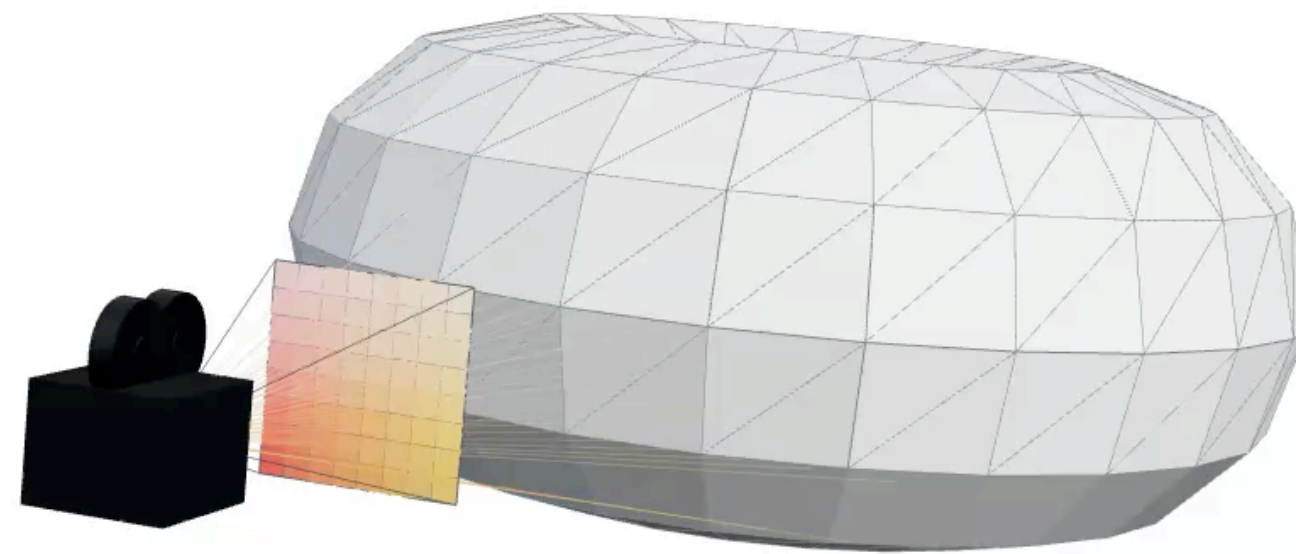
UV Map



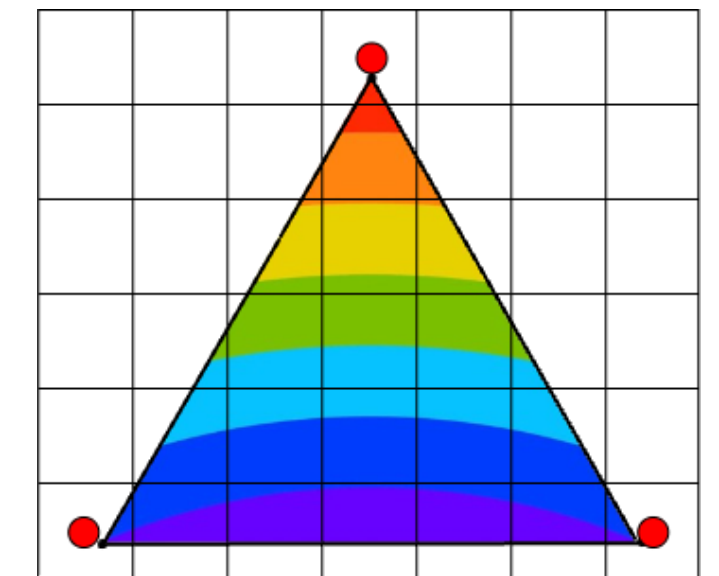
How to aggregate inconsistent signal from multiple views?



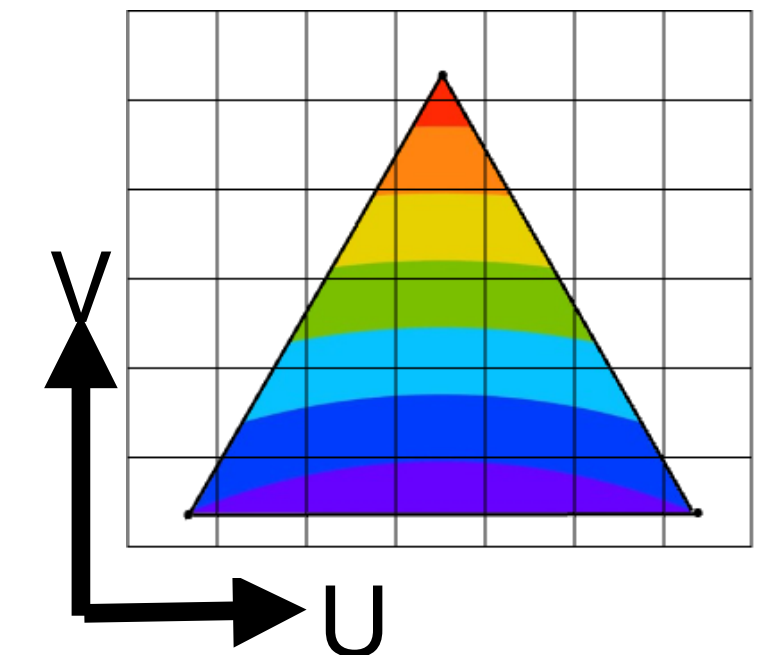
# Multi-view Backprojection



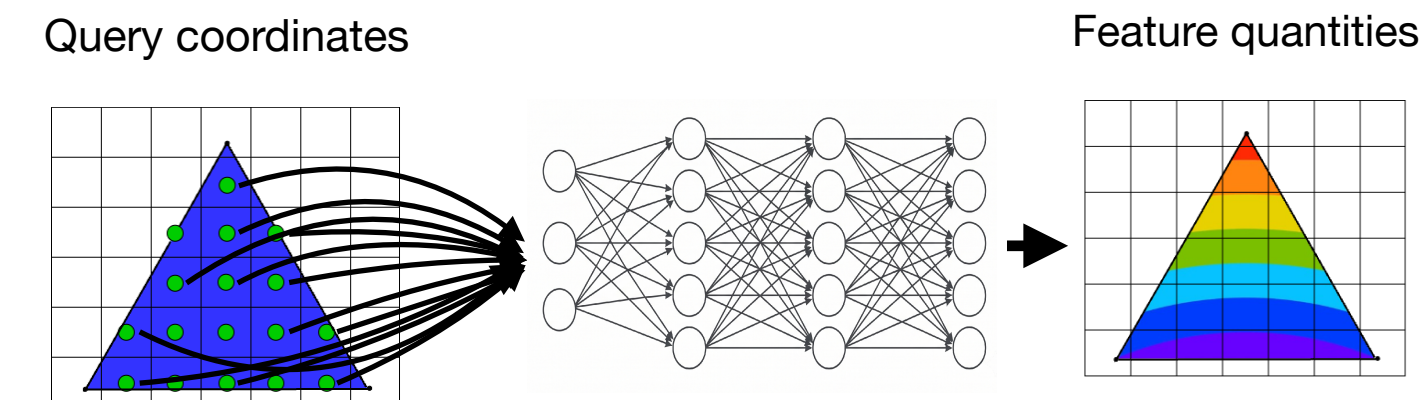
**Vertices**



**UV Map**

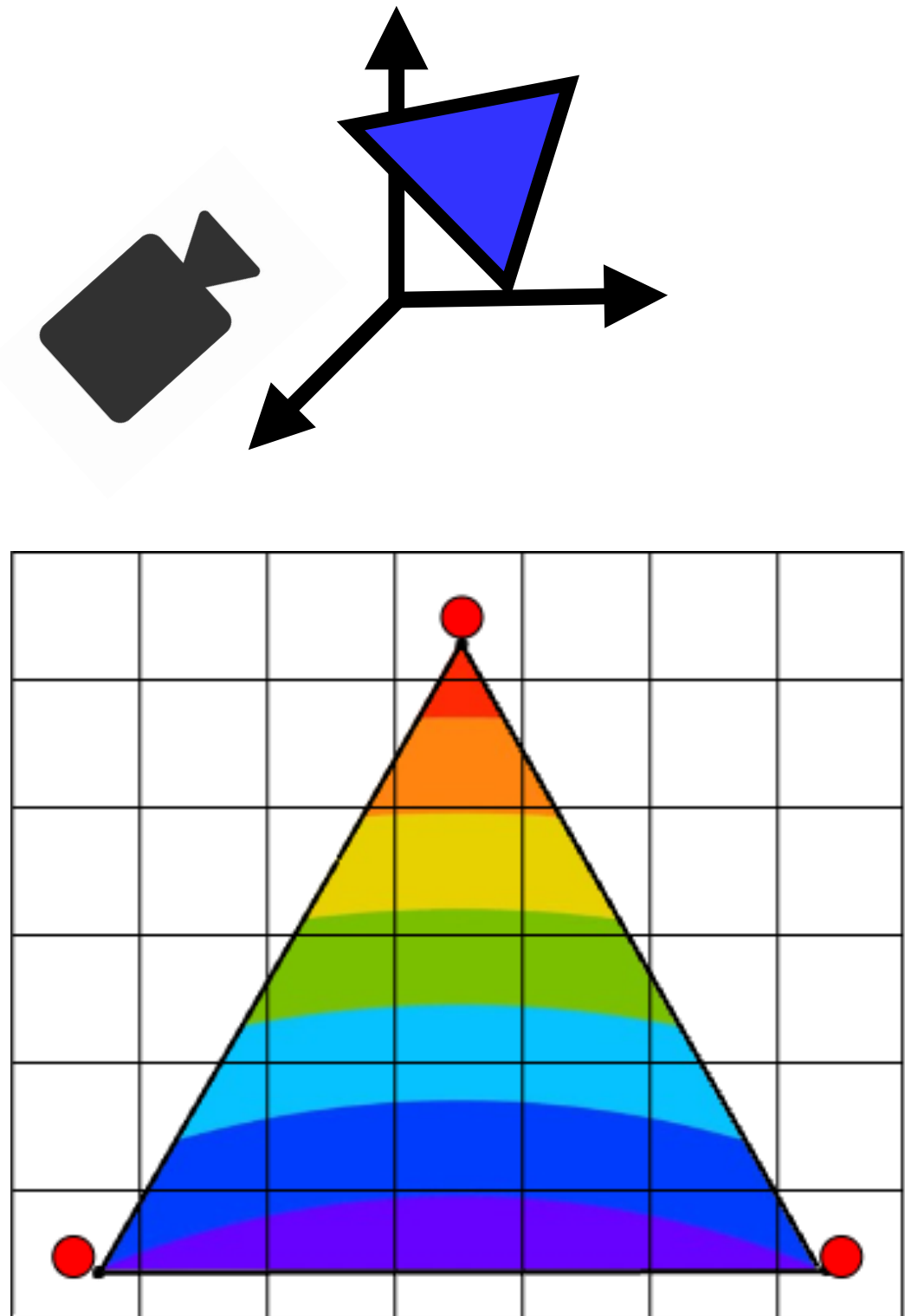


**Neural  
Field**



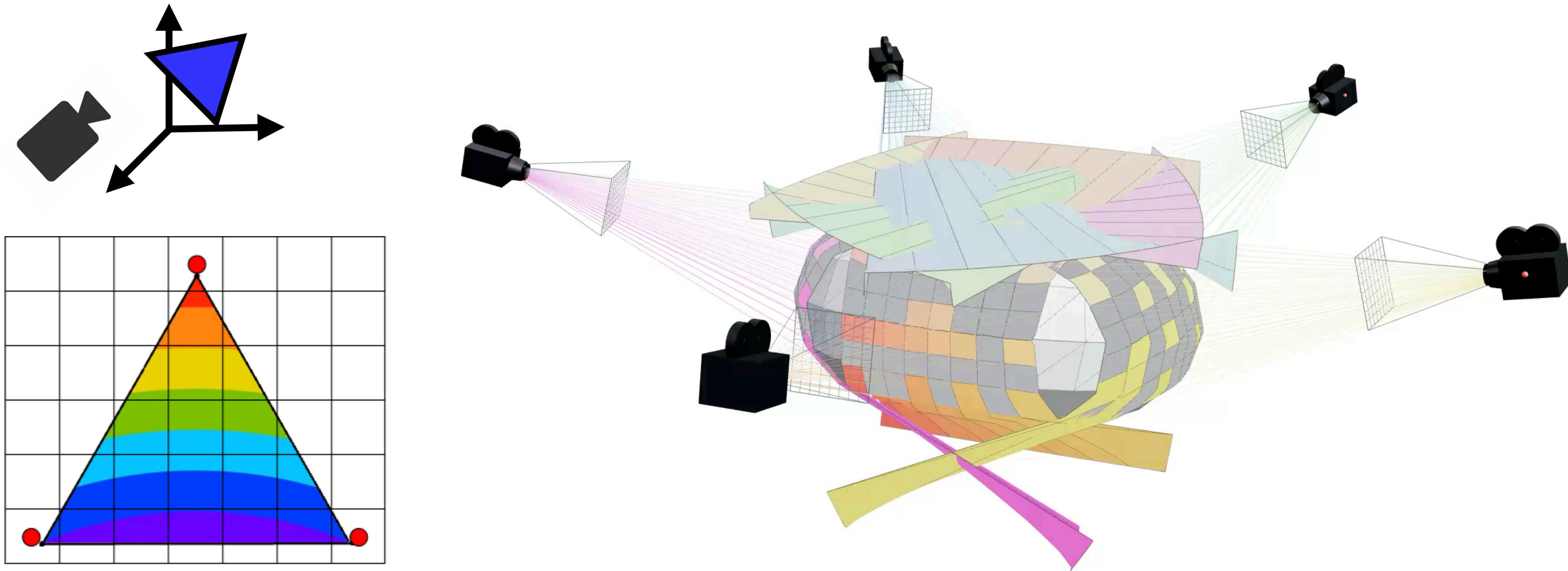
**How to aggregate inconsistent signal from multiple views?**

# Vertex Aggregation



Aggregate into vertices: lossy & dependent on mesh resolution

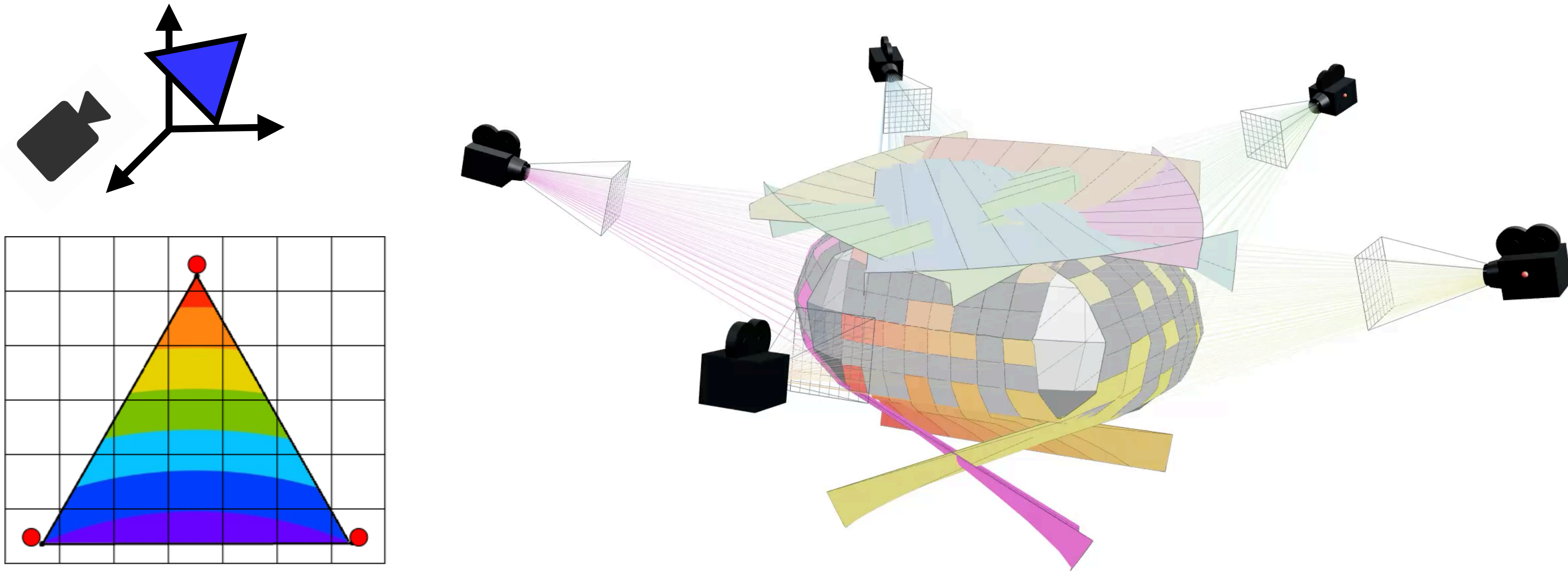
# Vertex Aggregation



Aggregate into vertices: lossy & dependent on mesh resolution

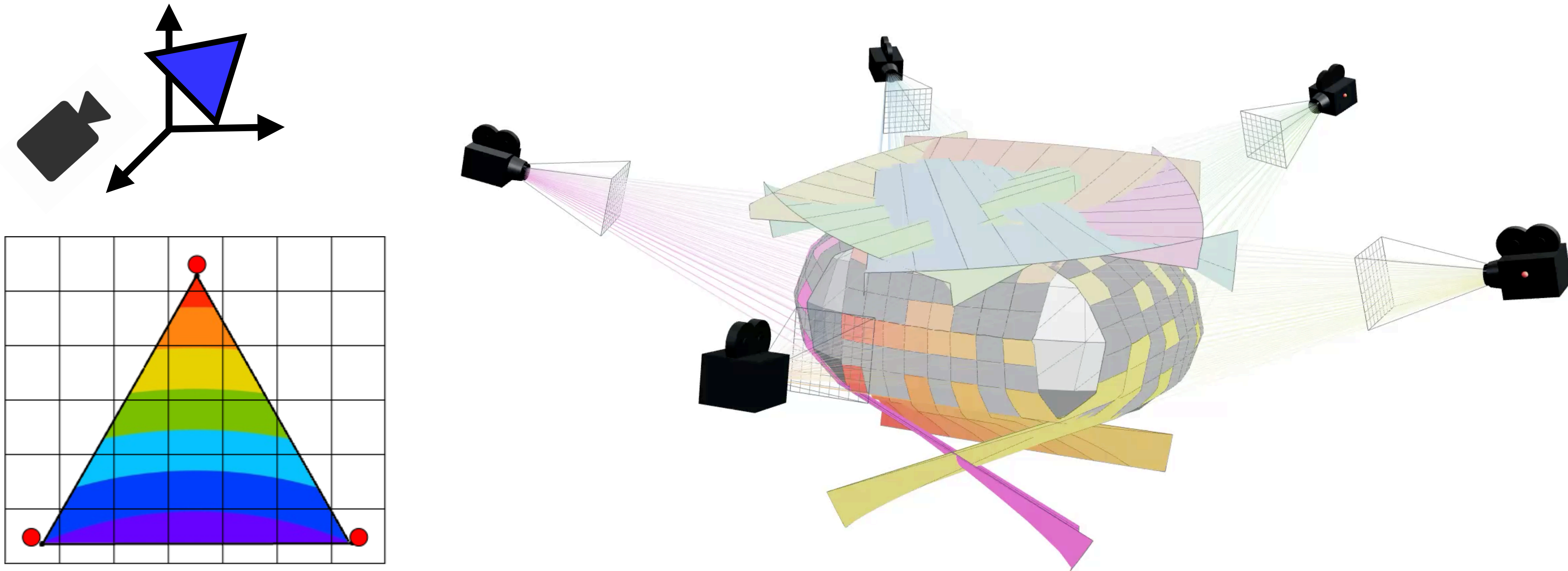


# Vertex Aggregation



Aggregate into vertices: lossy & dependent on mesh resolution

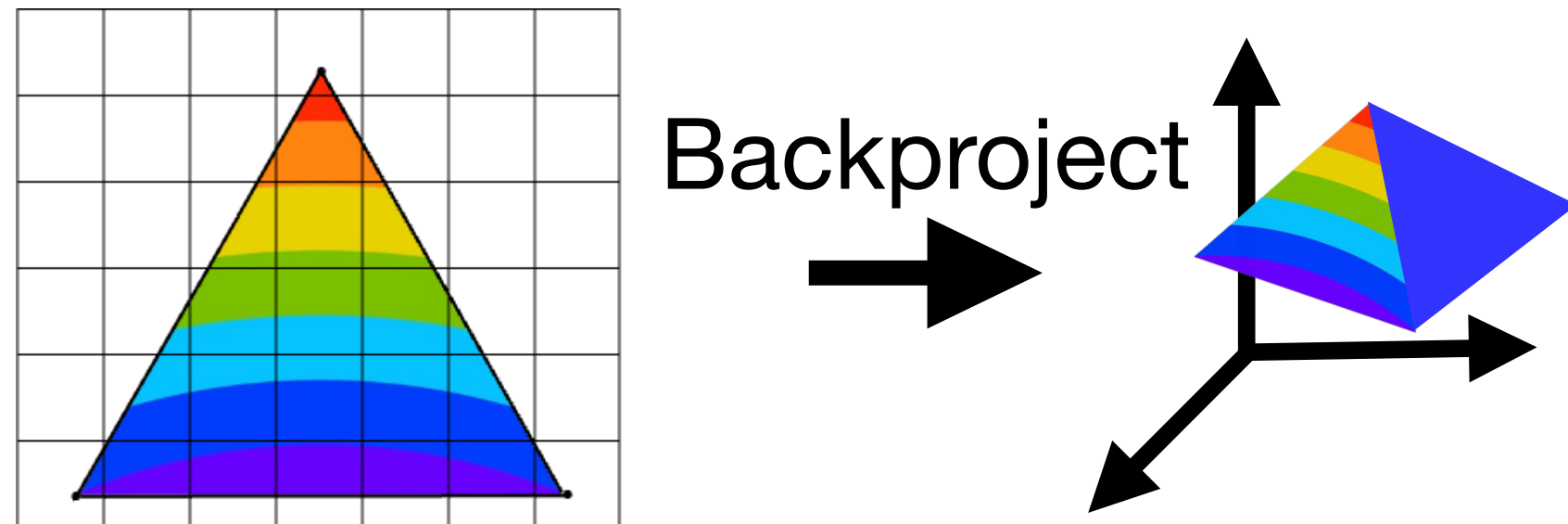
# Vertex Aggregation



Aggregate into vertices: lossy & dependent on mesh resolution

>50% surface pixels do not contribute at  $8^2$  render resolution

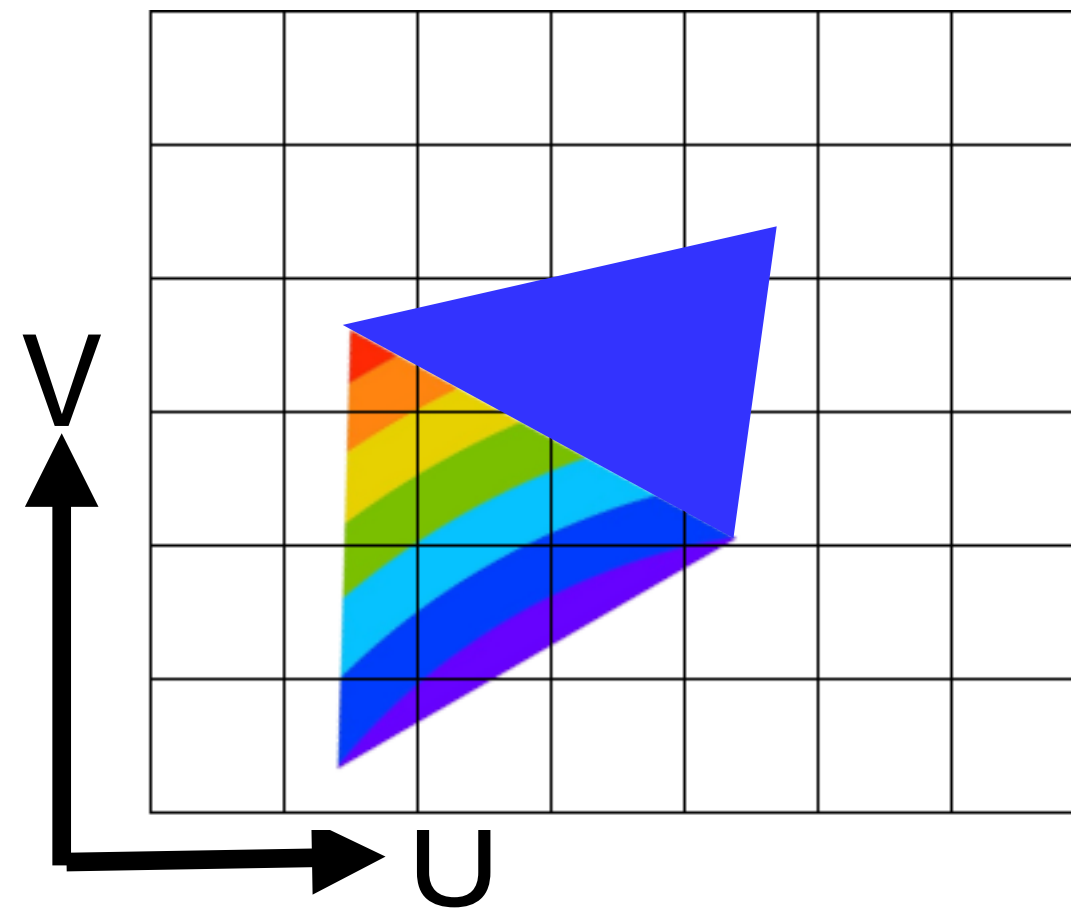
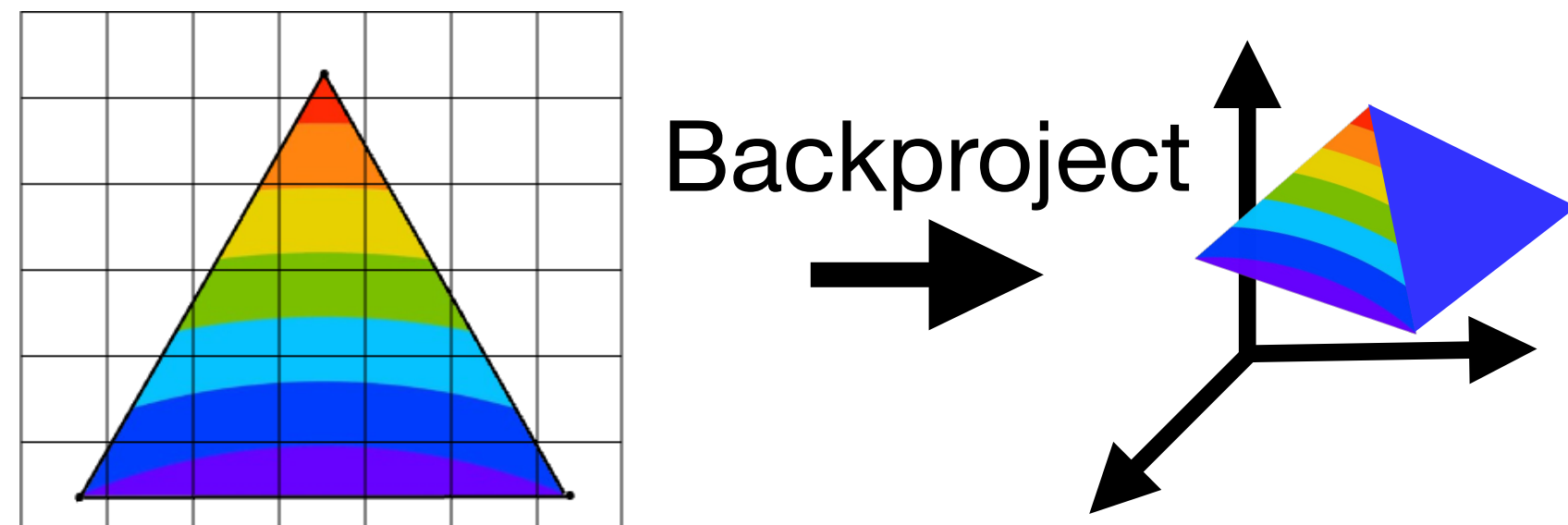
# UV Aggregation



Aggregate into UV map: requires UVs & quality depends on texel density

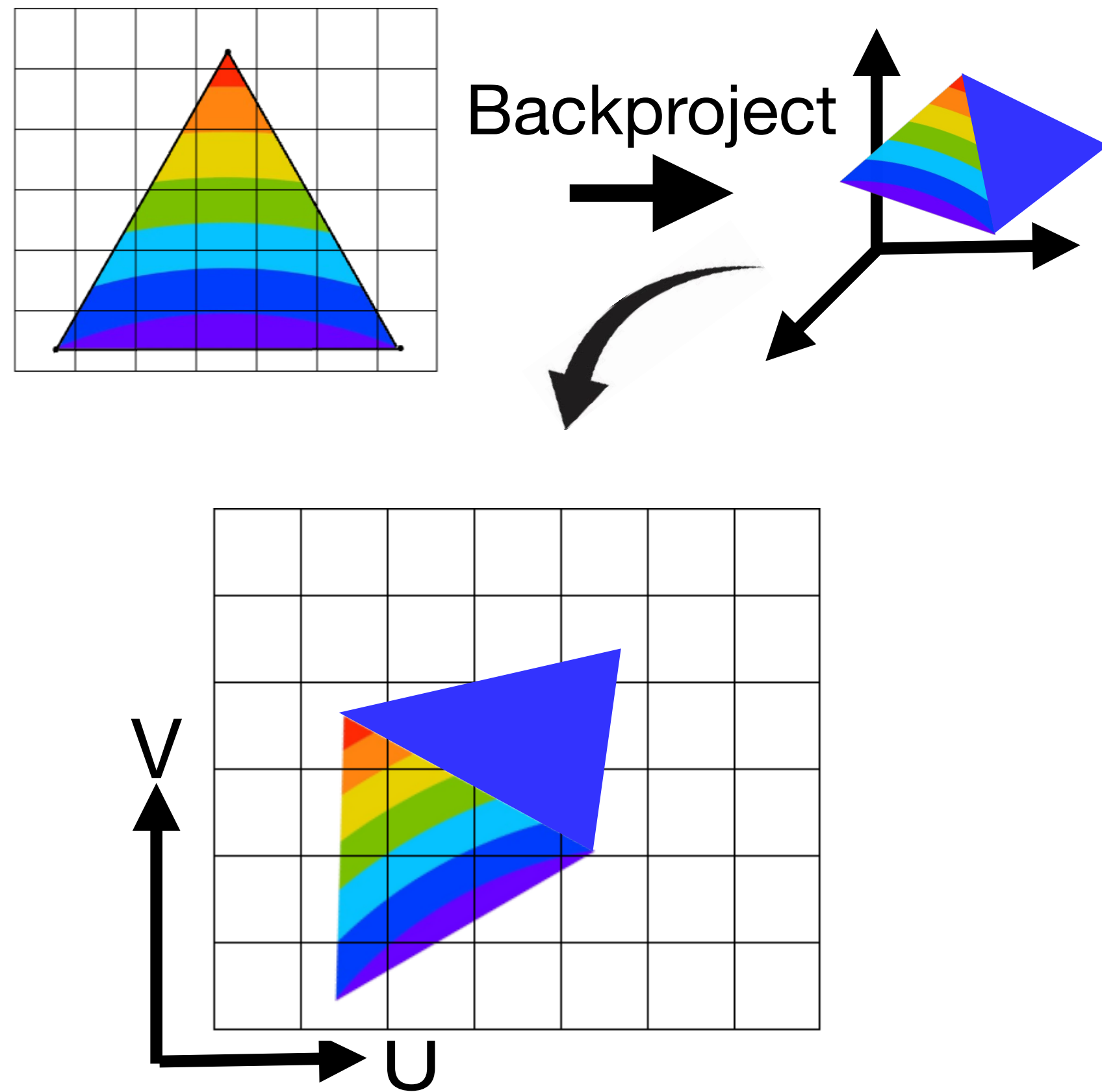


# UV Aggregation



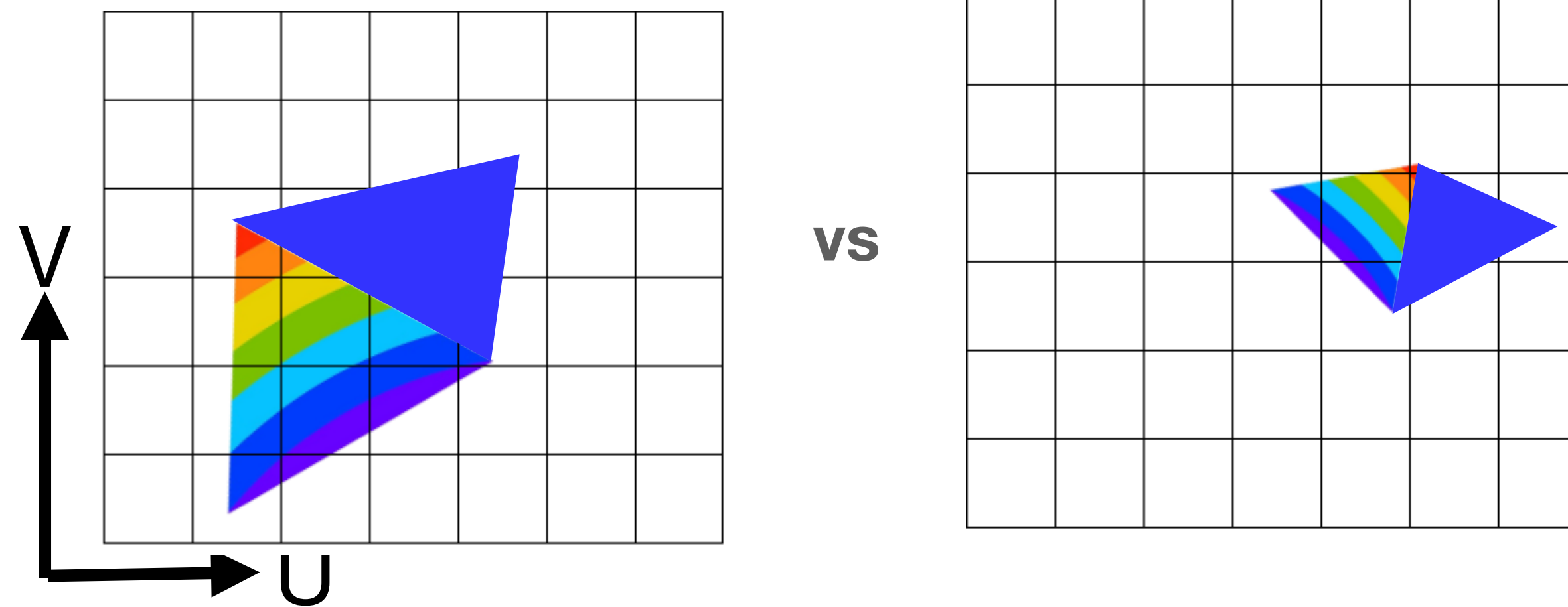
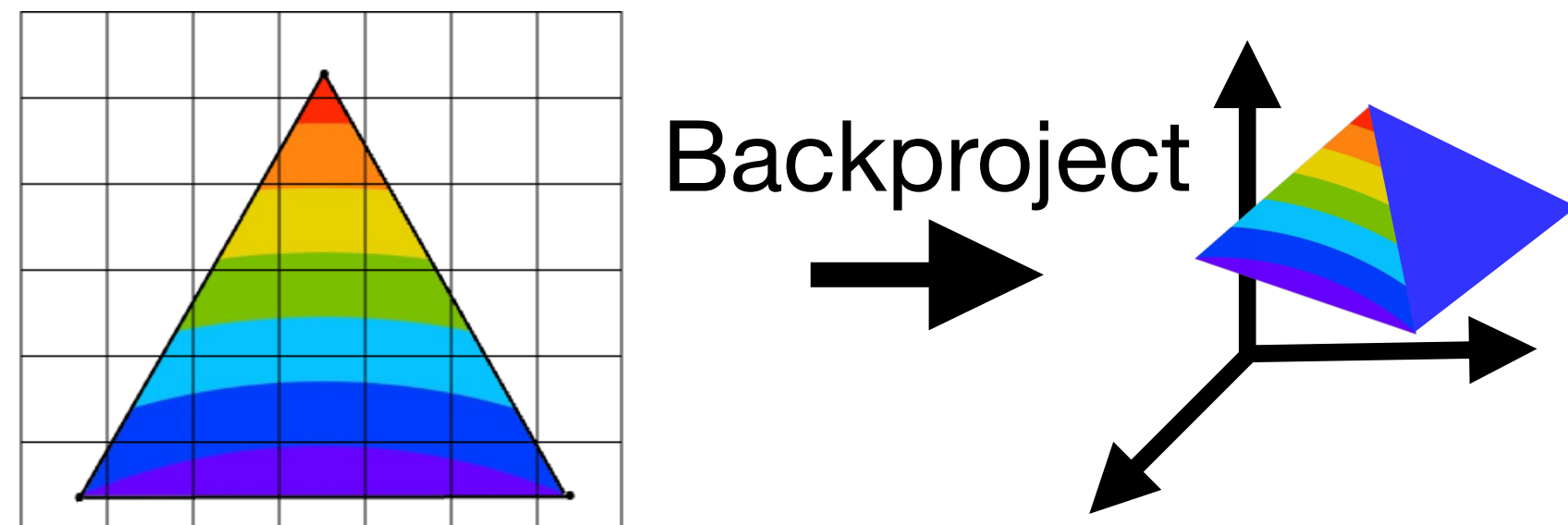
Aggregate into UV map: requires UVs & quality depends on texel density

# UV Aggregation



Aggregate into UV map: requires UVs & quality depends on texel density

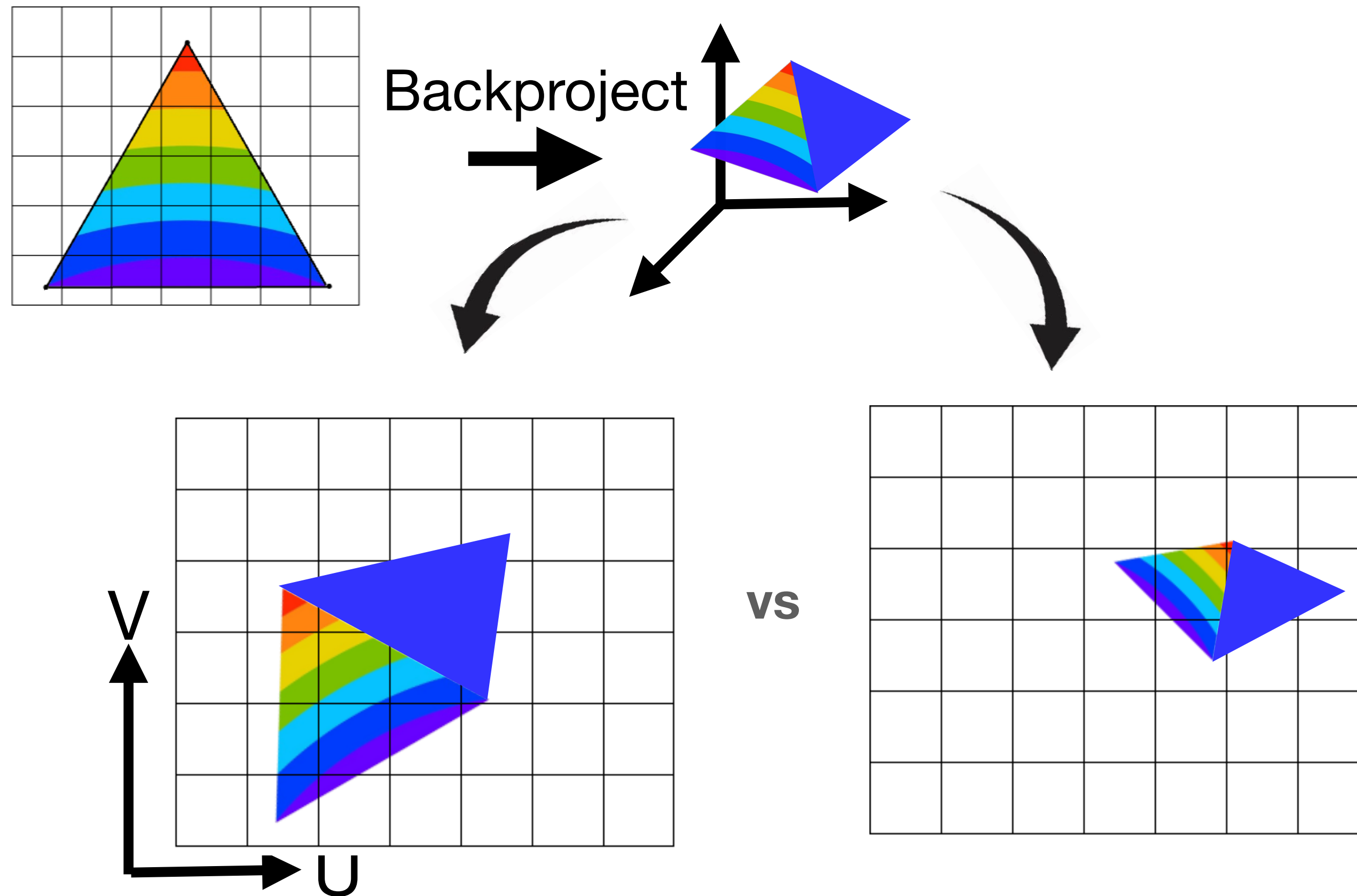
# UV Aggregation



Aggregate into UV map: requires UVs & quality depends on texel density

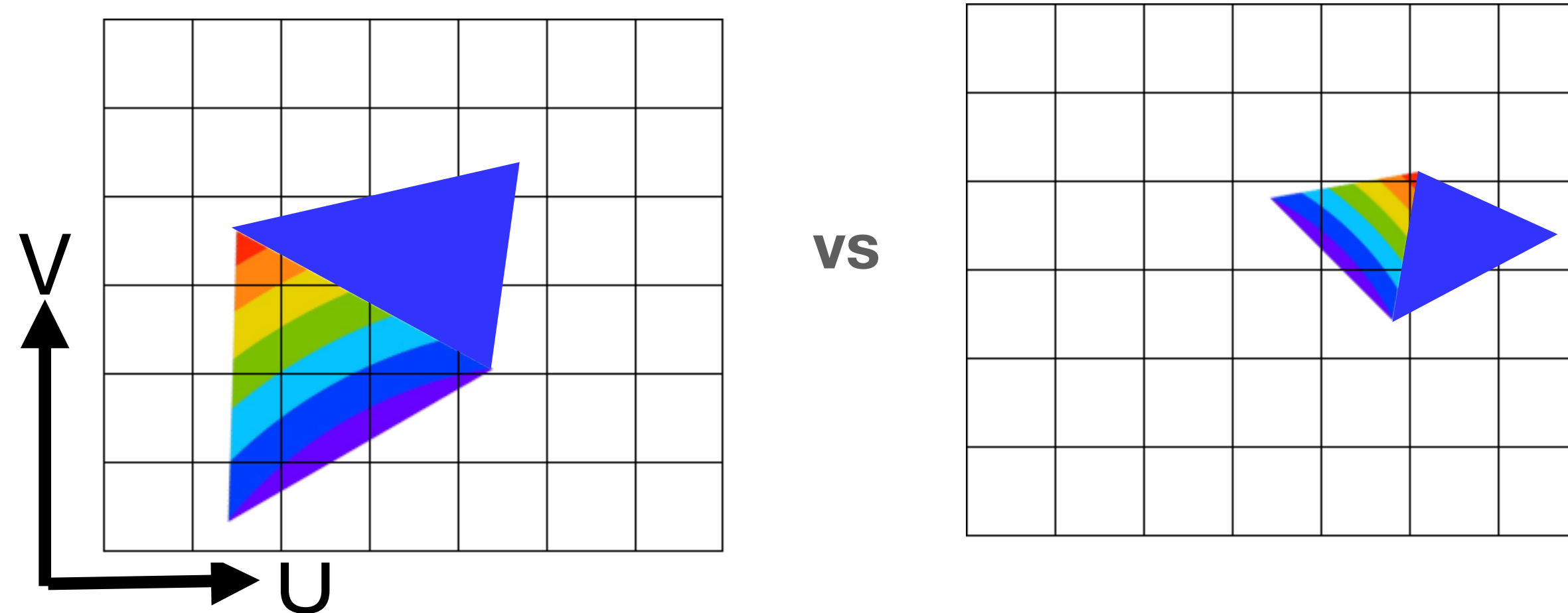
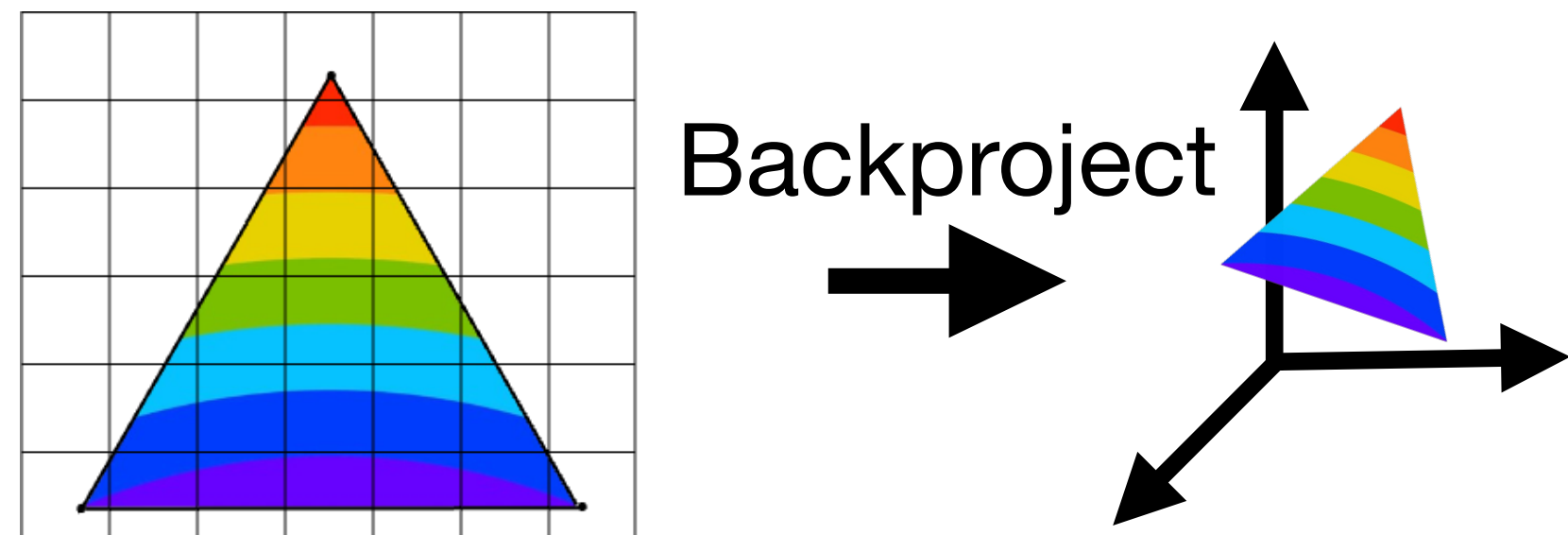


# UV Aggregation

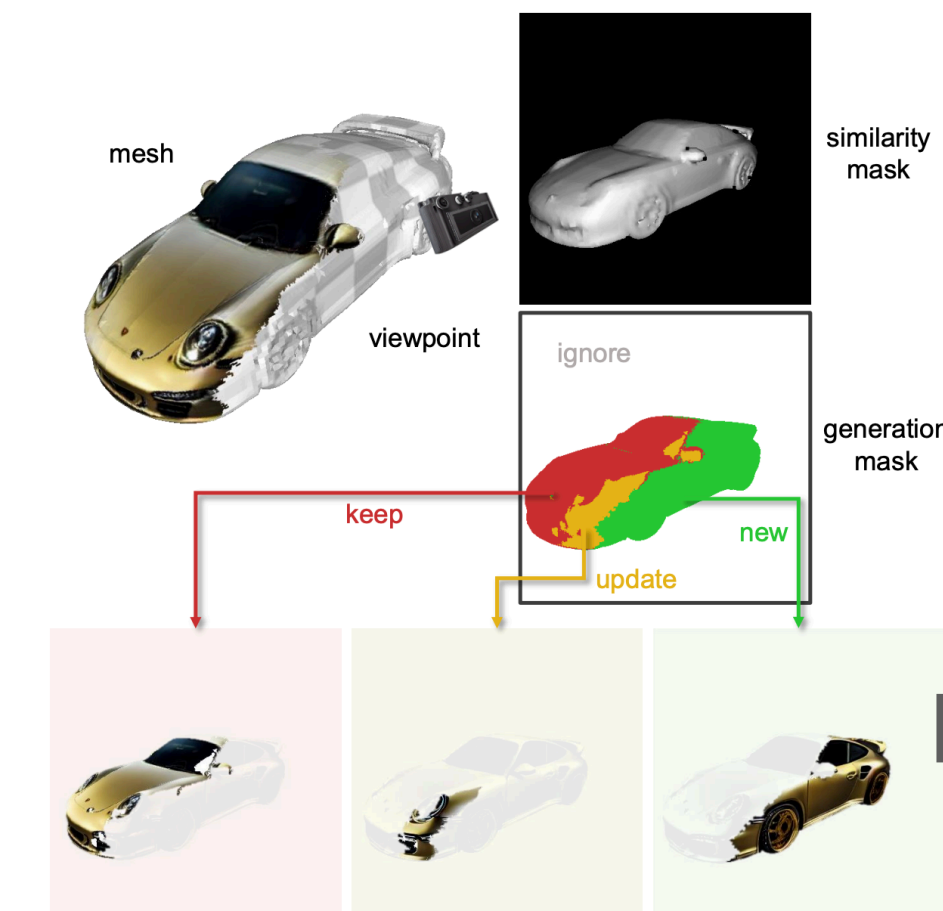
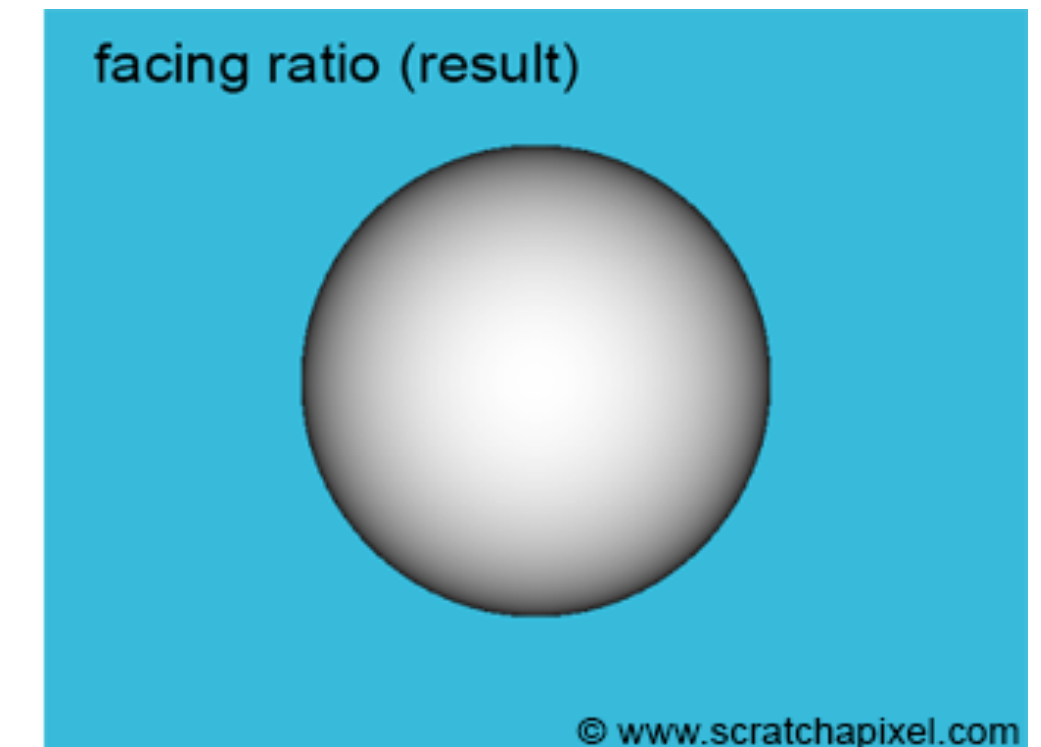
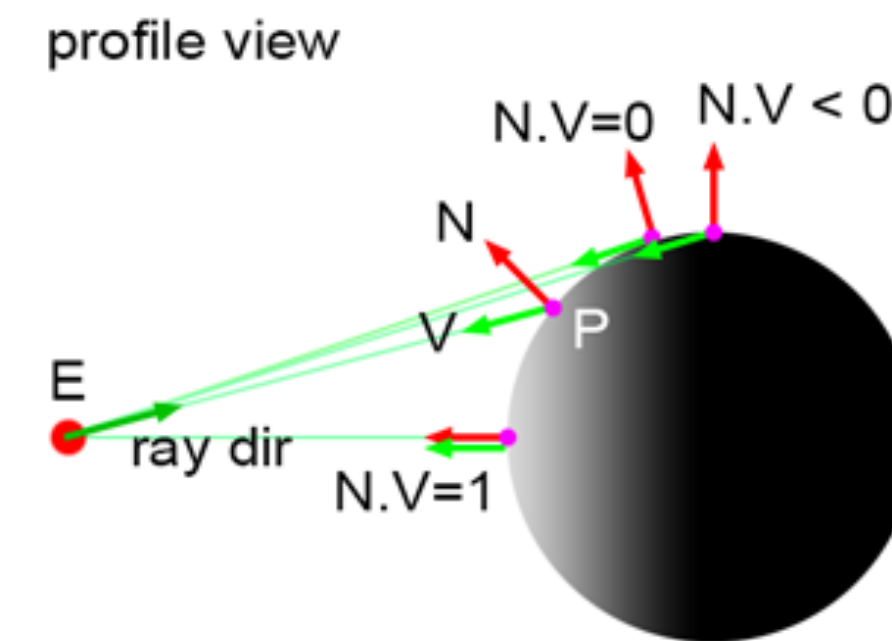


Aggregate into UV map: requires UVs & quality depends on texel density

# UV Aggregation



## Confidence scoring from normal alignment

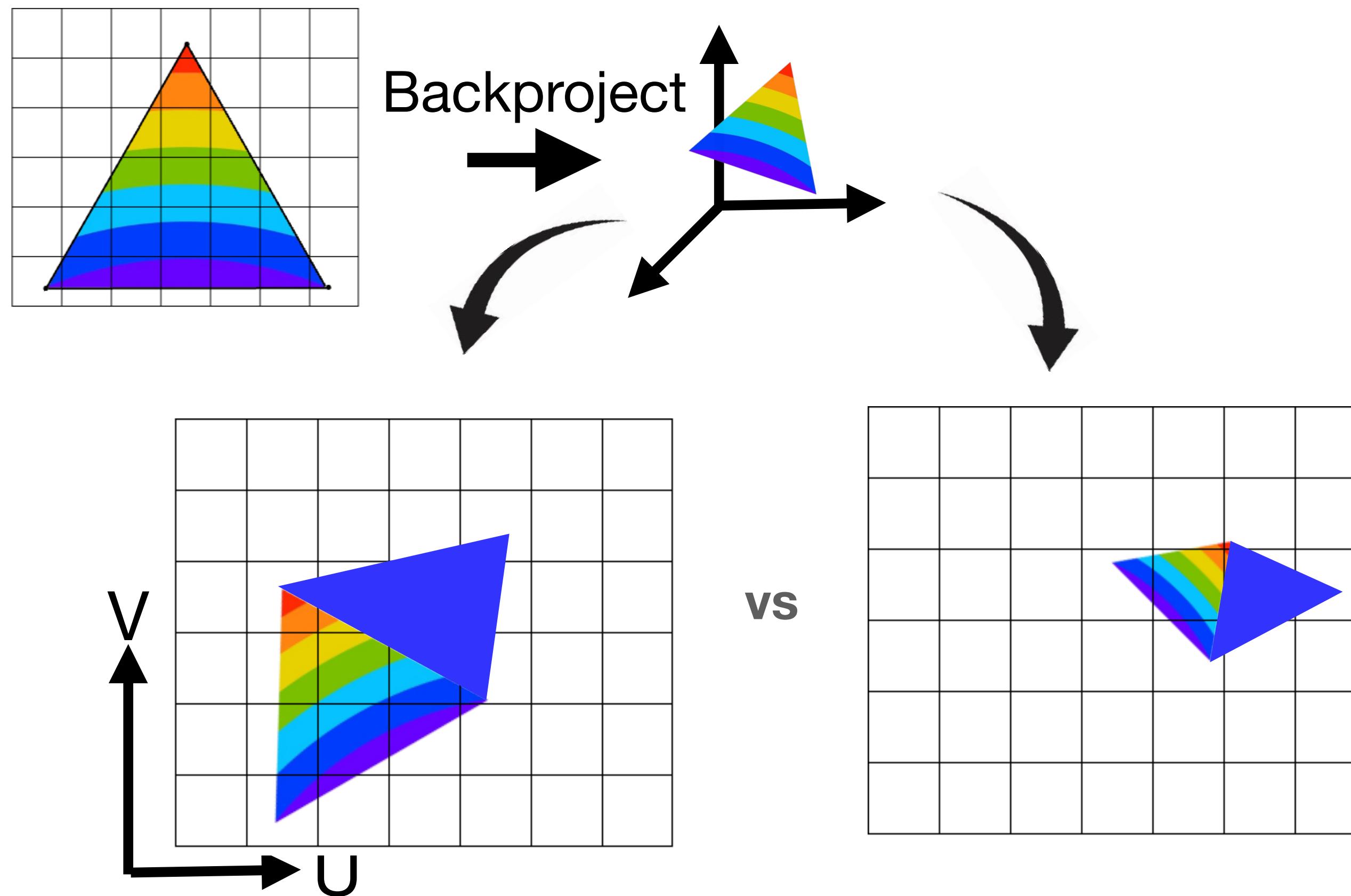


**Text2Tex**  
Chen et al. 2023

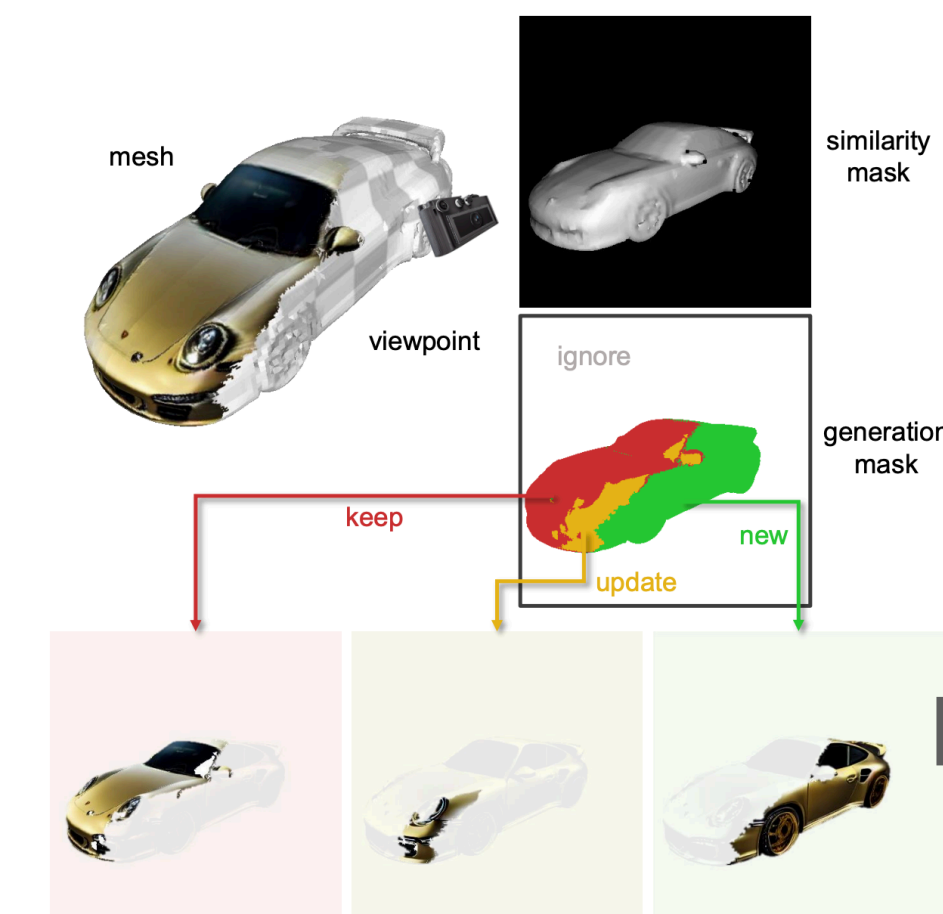
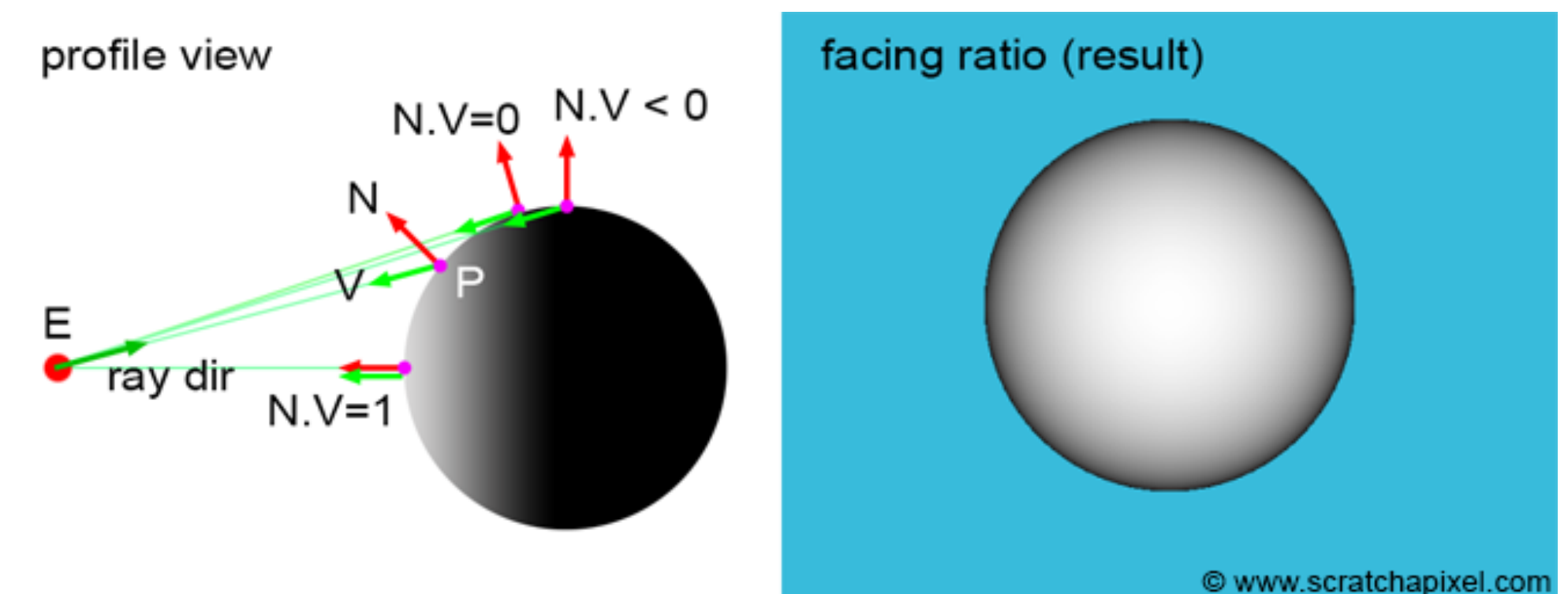
**TEXTure**  
Richardson et al. 2023

Aggregate into UV map: requires UVs & quality depends on texel density

# UV Aggregation



## Confidence scoring from normal alignment



**Text2Tex**  
Chen et al. 2023

**TEXTure**  
Richardson et al. 2023

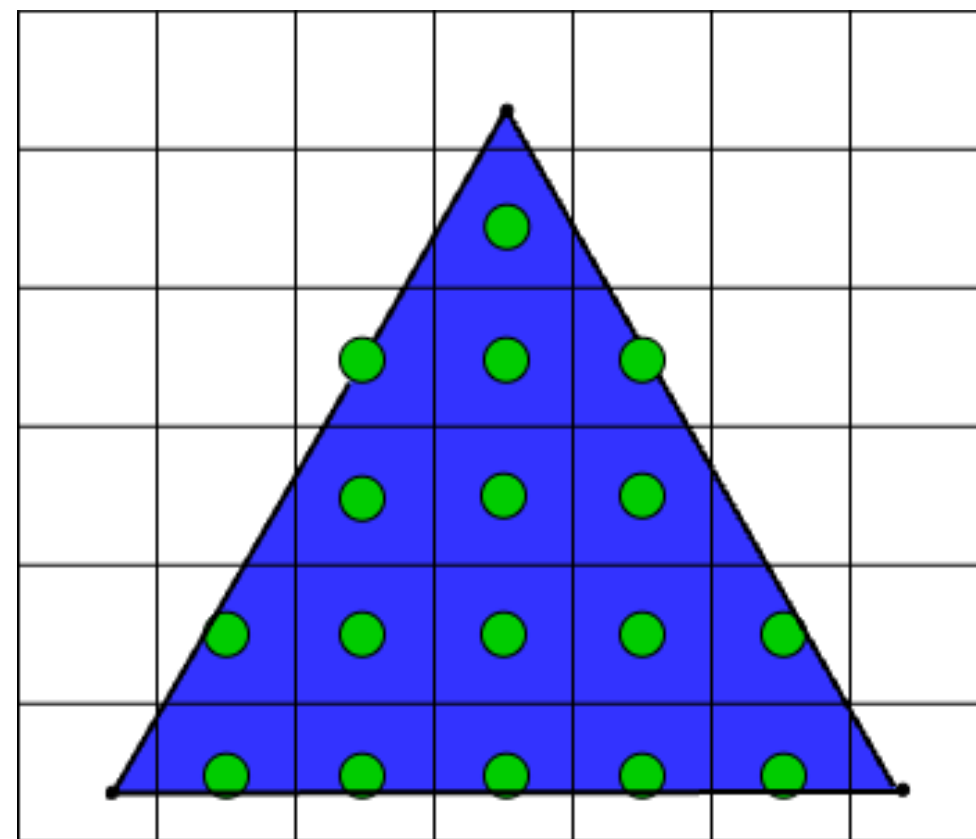
Aggregate into UV map: requires UVs & quality depends on texel density



# Neural Field Aggregation

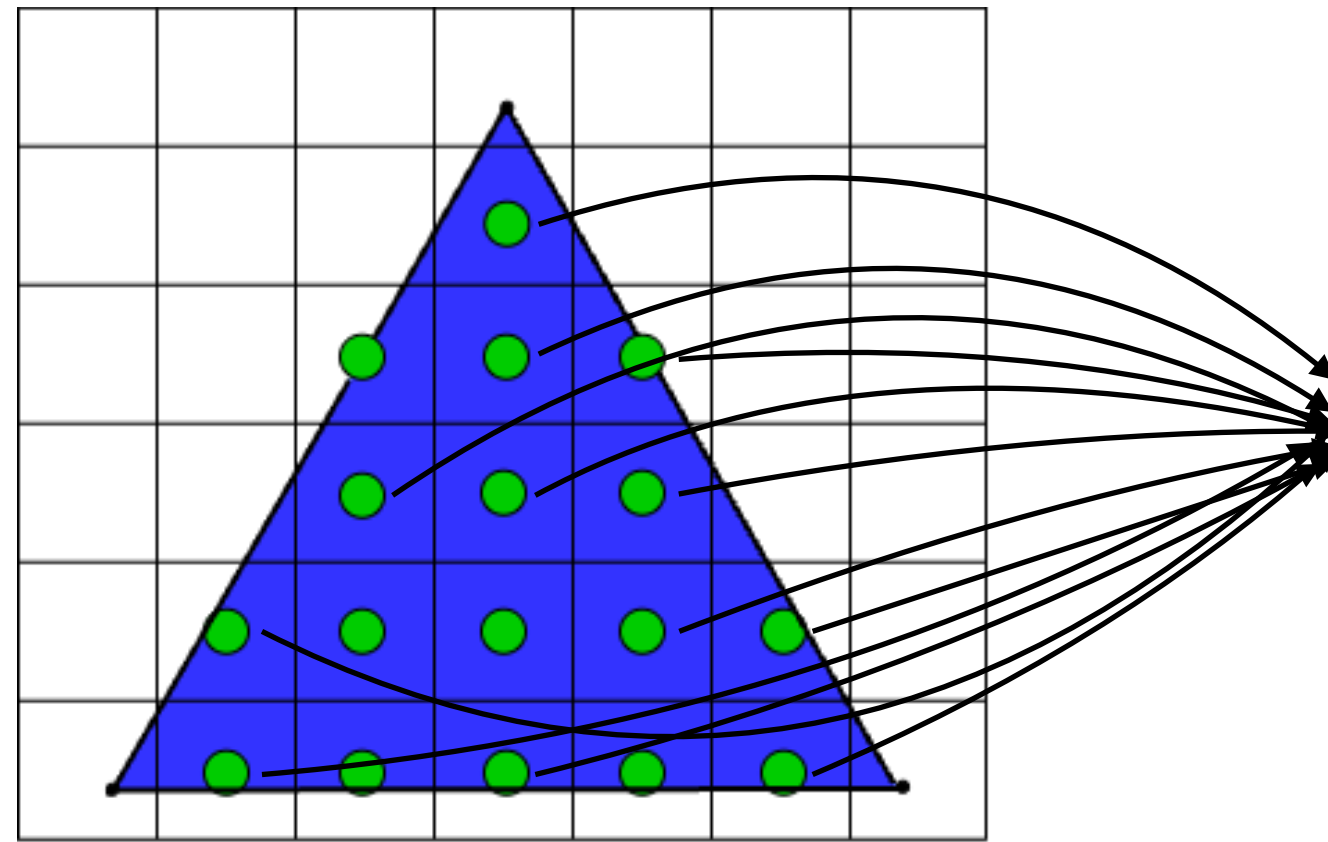
# Neural Field Aggregation

Query  
coordinates

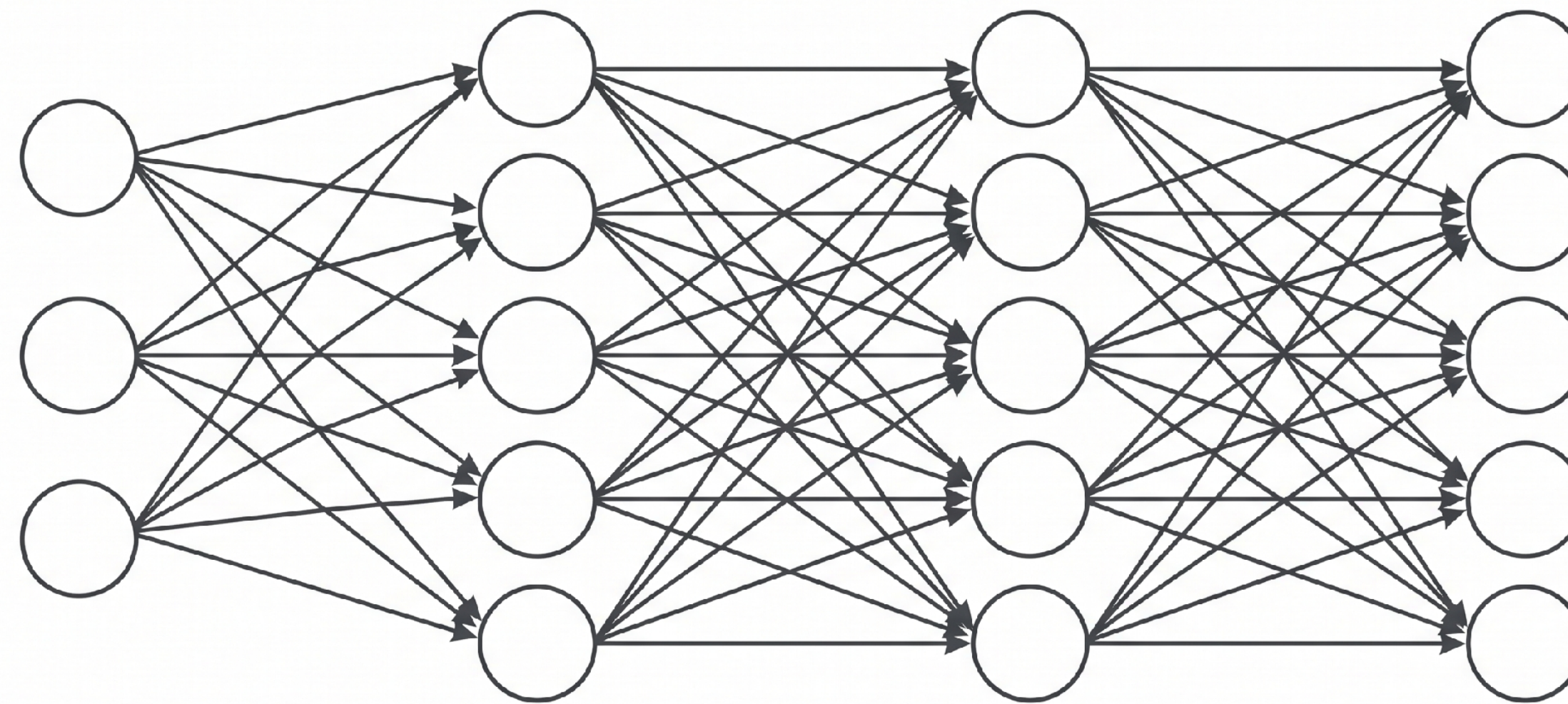


# Neural Field Aggregation

Query  
coordinates



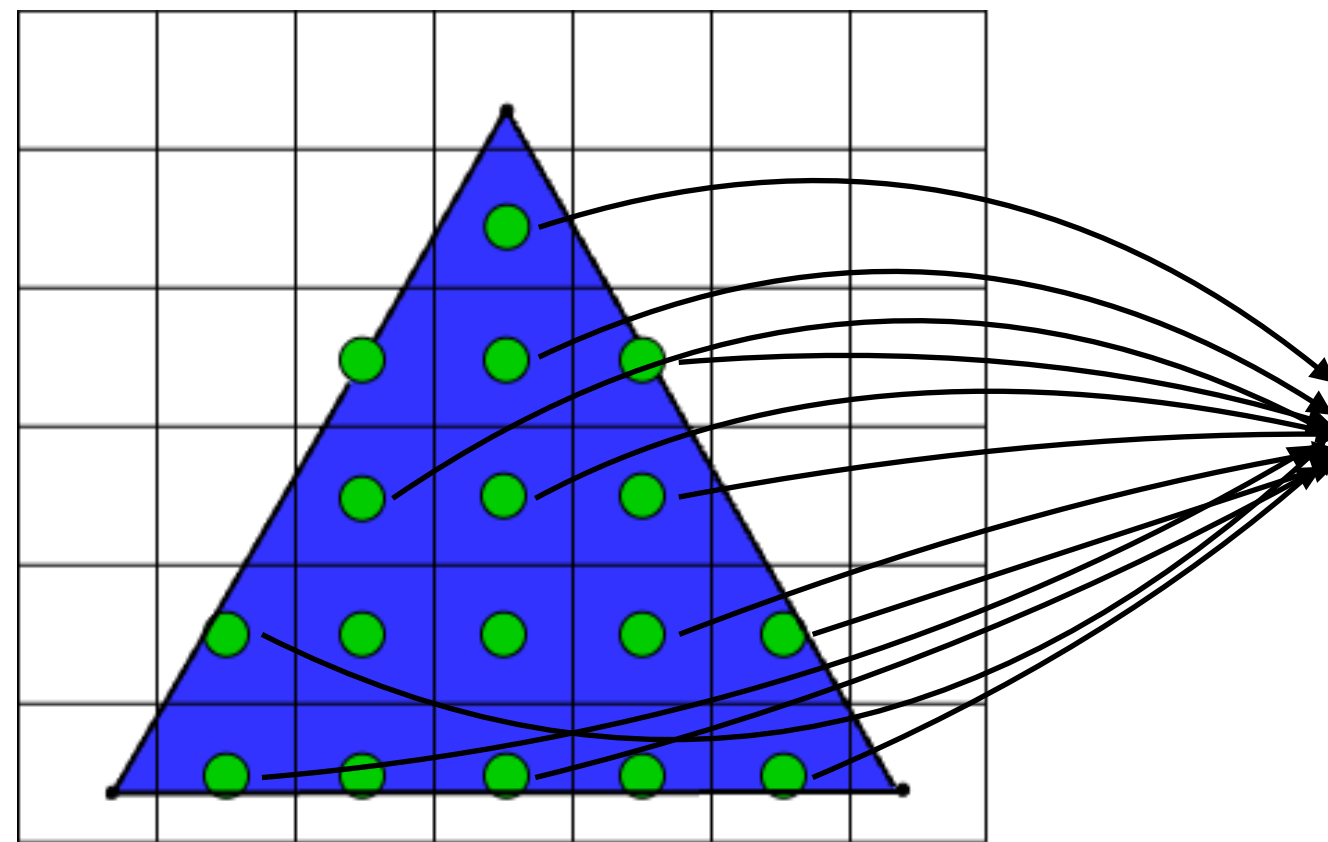
Neural Field



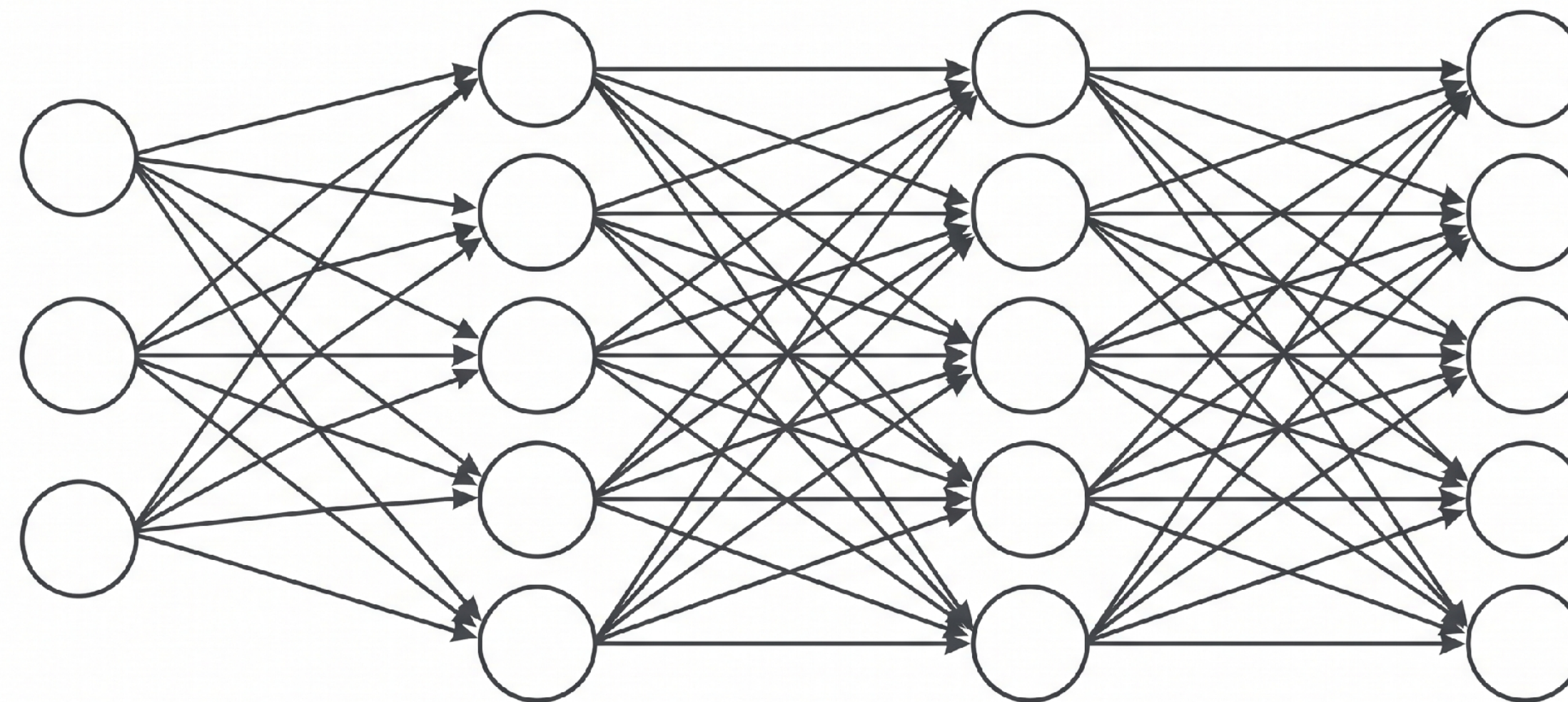


# Neural Field Aggregation

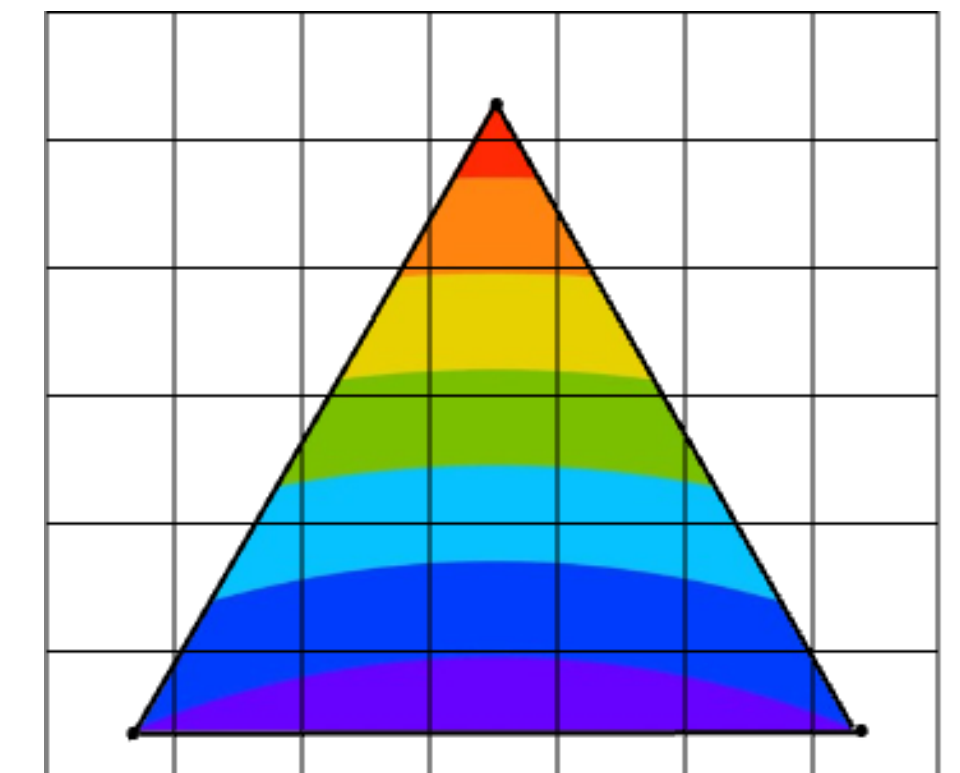
Query  
coordinates



Neural Field



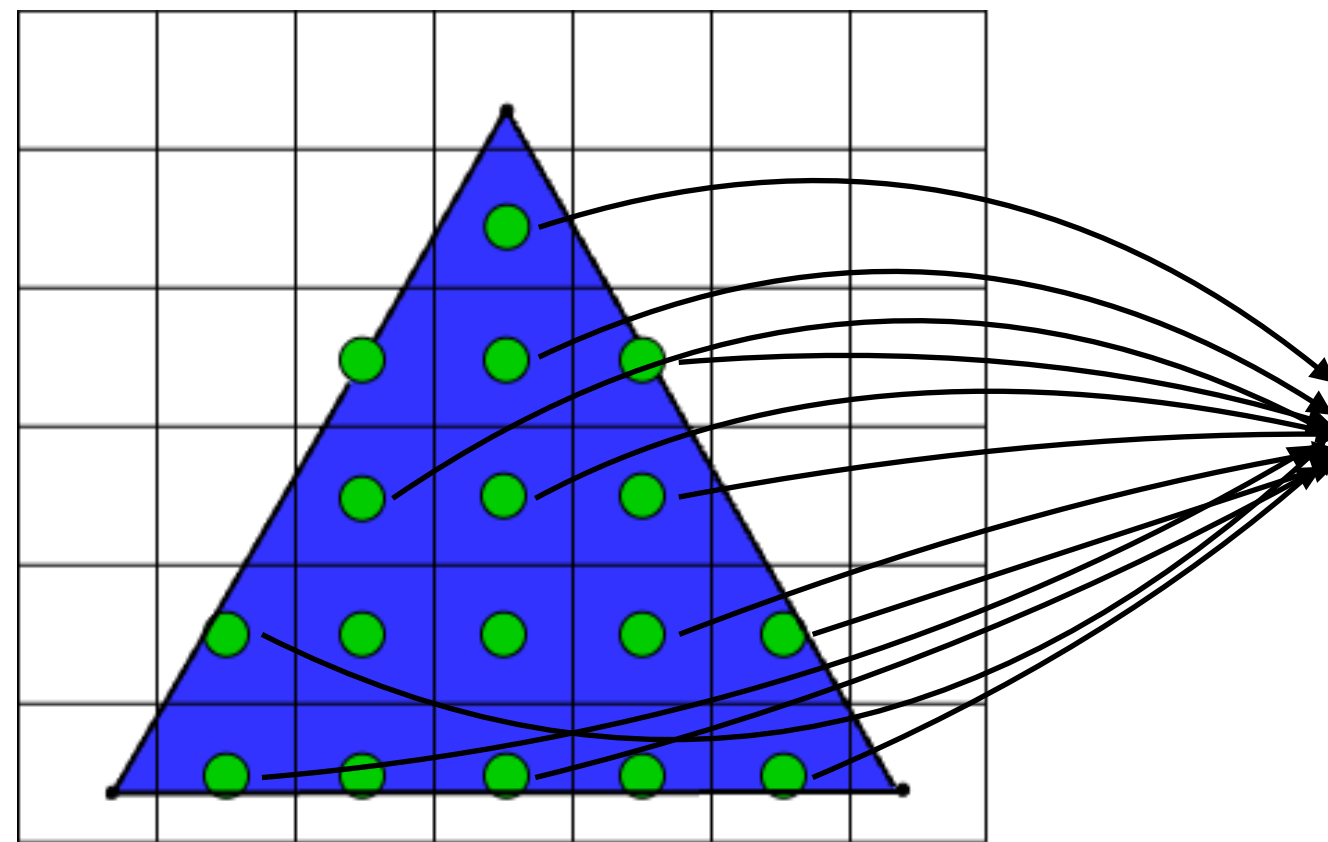
Feature  
quantities



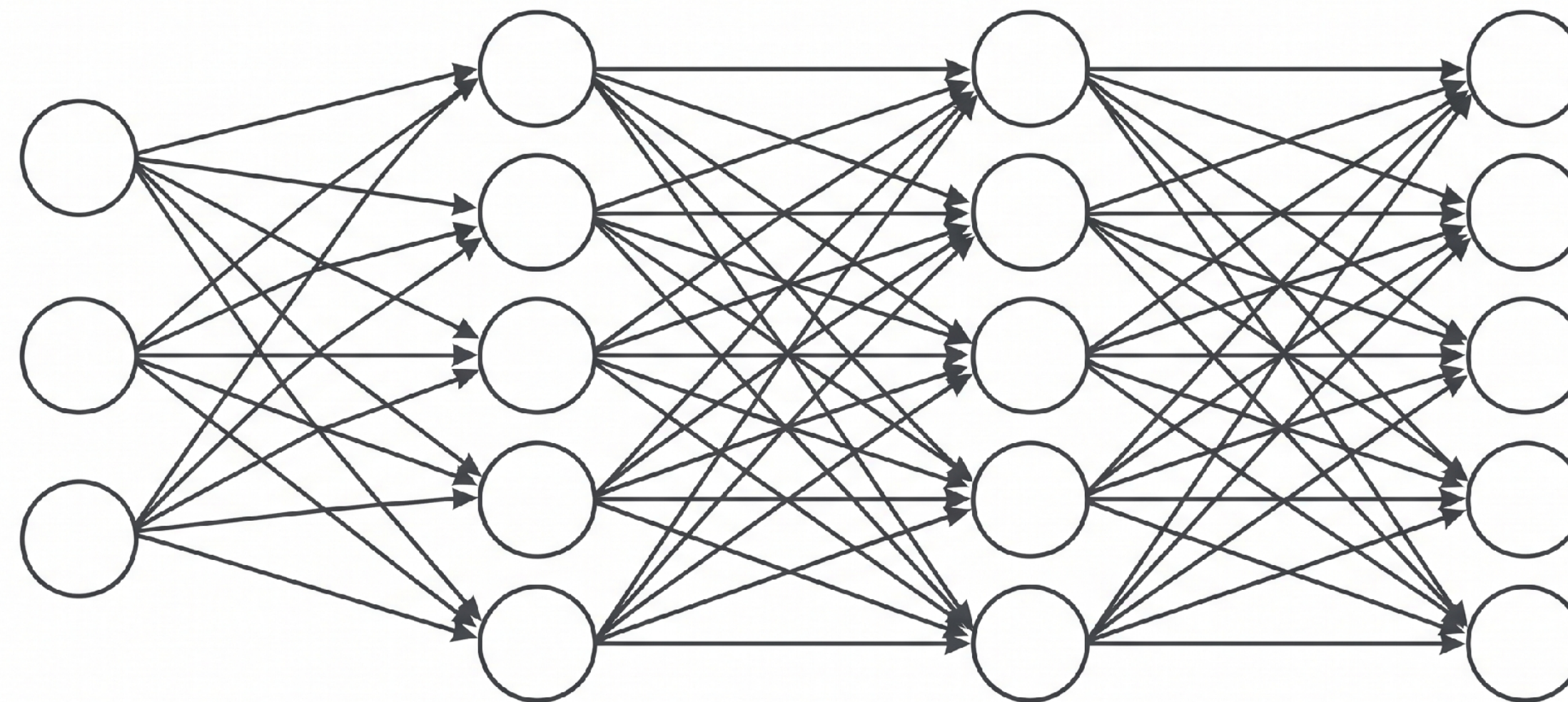


# Neural Field Aggregation

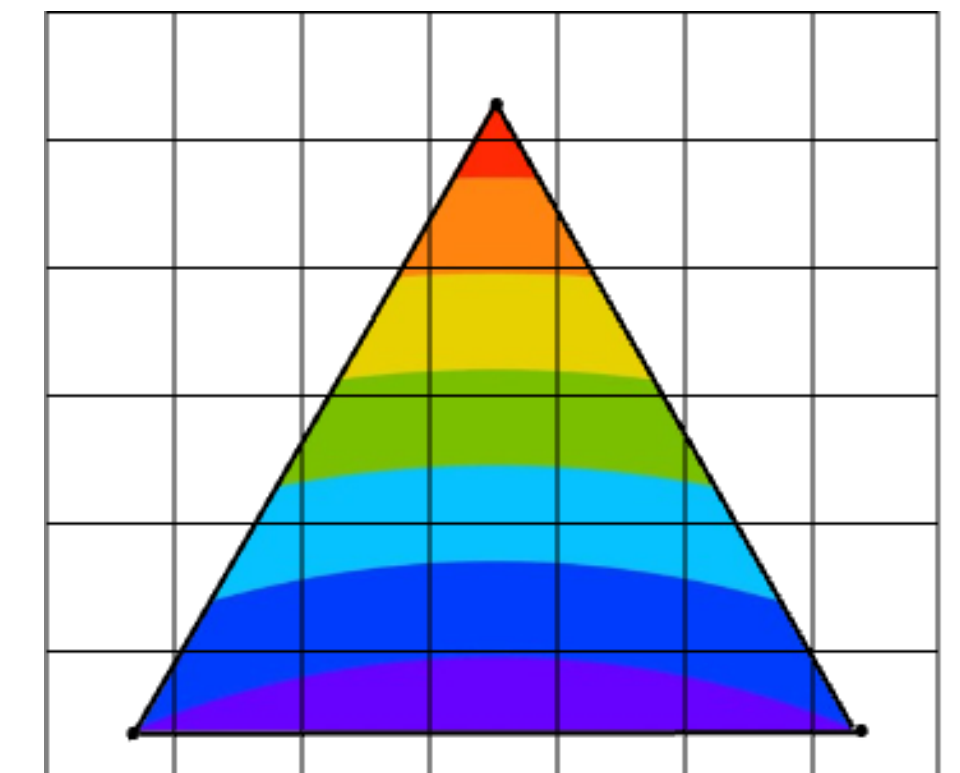
Query  
coordinates



Neural Field



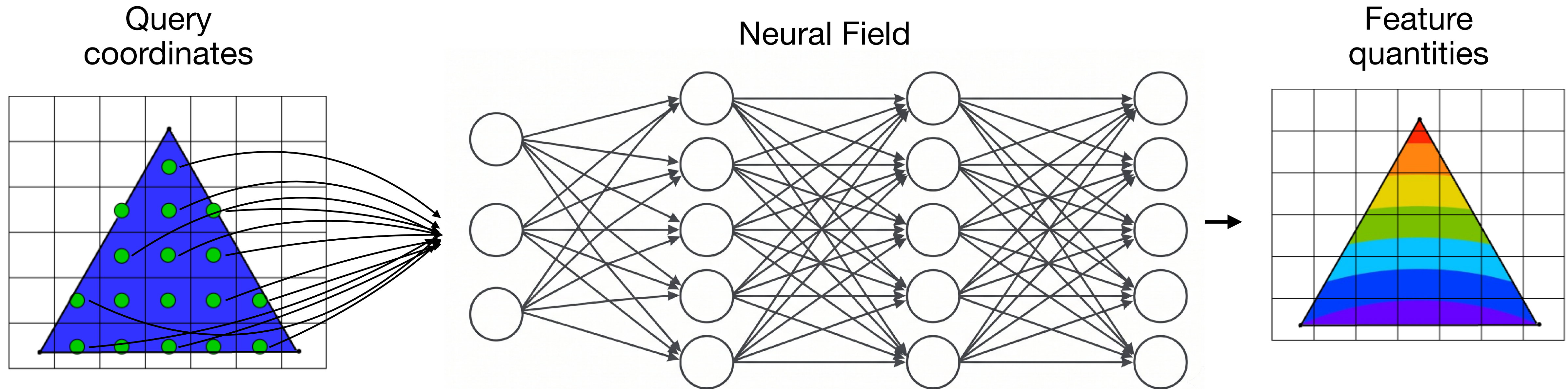
Feature  
quantities



Smoothly interpolates sample data



# Neural Field Aggregation



Smoothly interpolates sample data

Function over ambient space (not restricted to discretization)



# Feature Fields: Representation Agnostic

**“Deep Feature  
Deformation  
Weights”  
Liu et al. 2026**

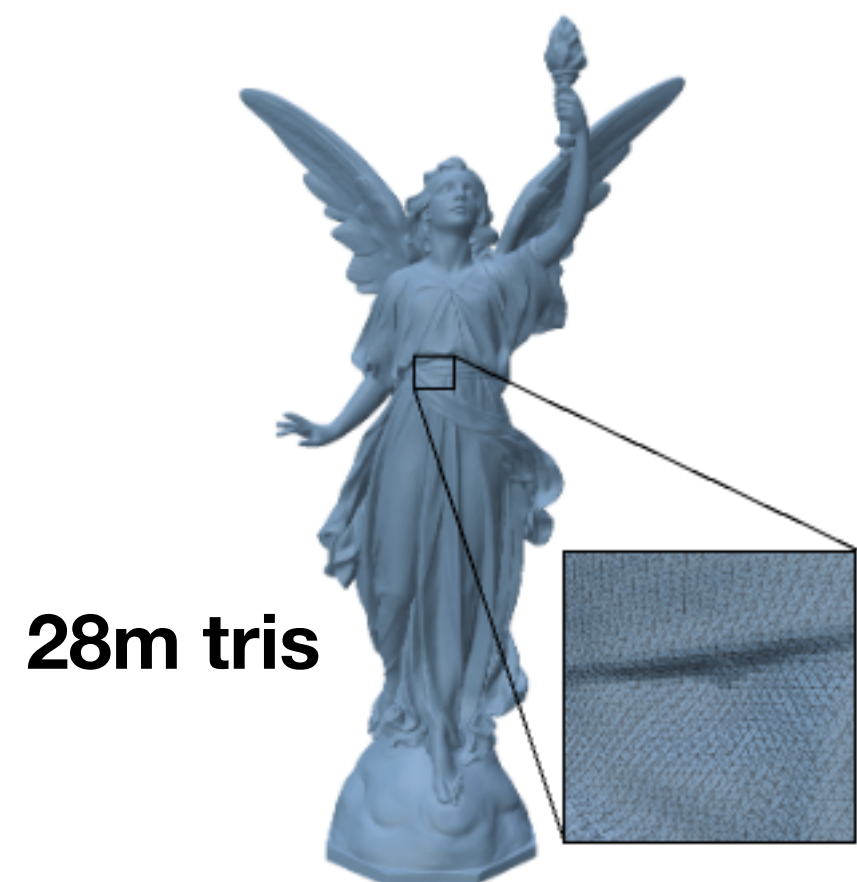
# Feature Fields: Representation Agnostic

Transfer across remeshing

**“Deep Feature  
Deformation  
Weights”  
Liu et al. 2026**

# Feature Fields: Representation Agnostic

Transfer across remeshing

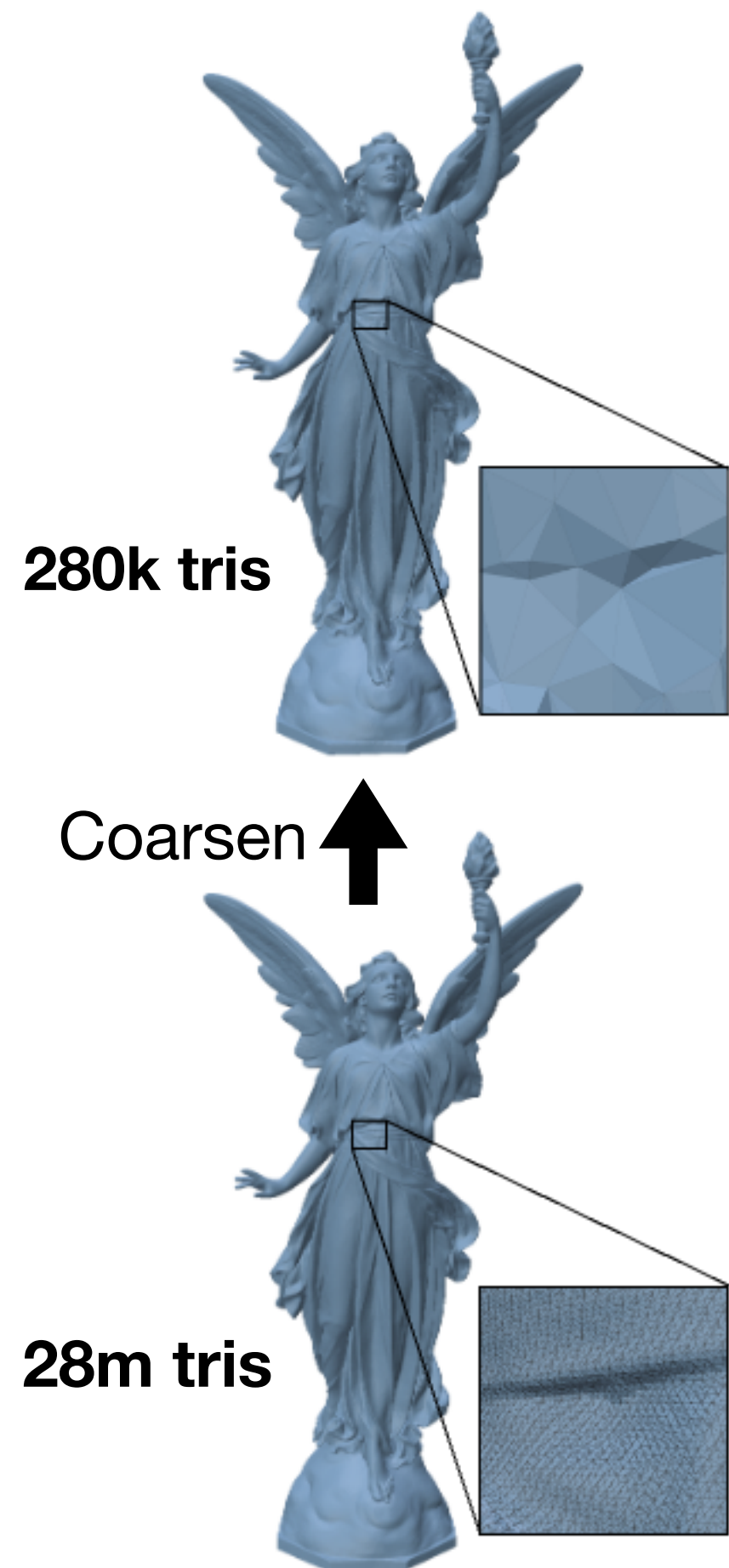


**“Deep Feature  
Deformation  
Weights”**  
Liu et al. 2026



# Feature Fields: Representation Agnostic

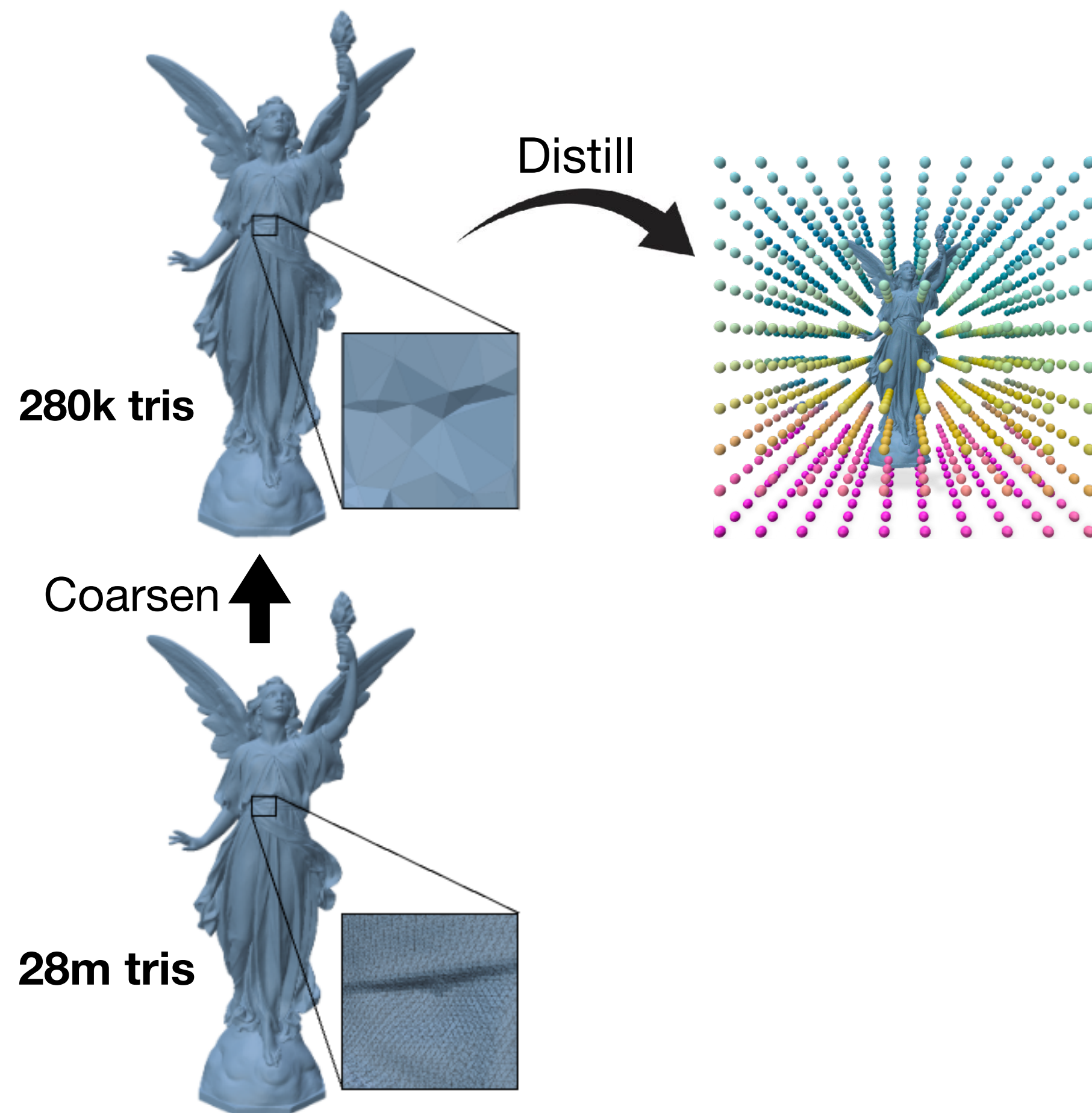
Transfer across remeshing



**“Deep Feature  
Deformation  
Weights”**  
Liu et al. 2026

# Feature Fields: Representation Agnostic

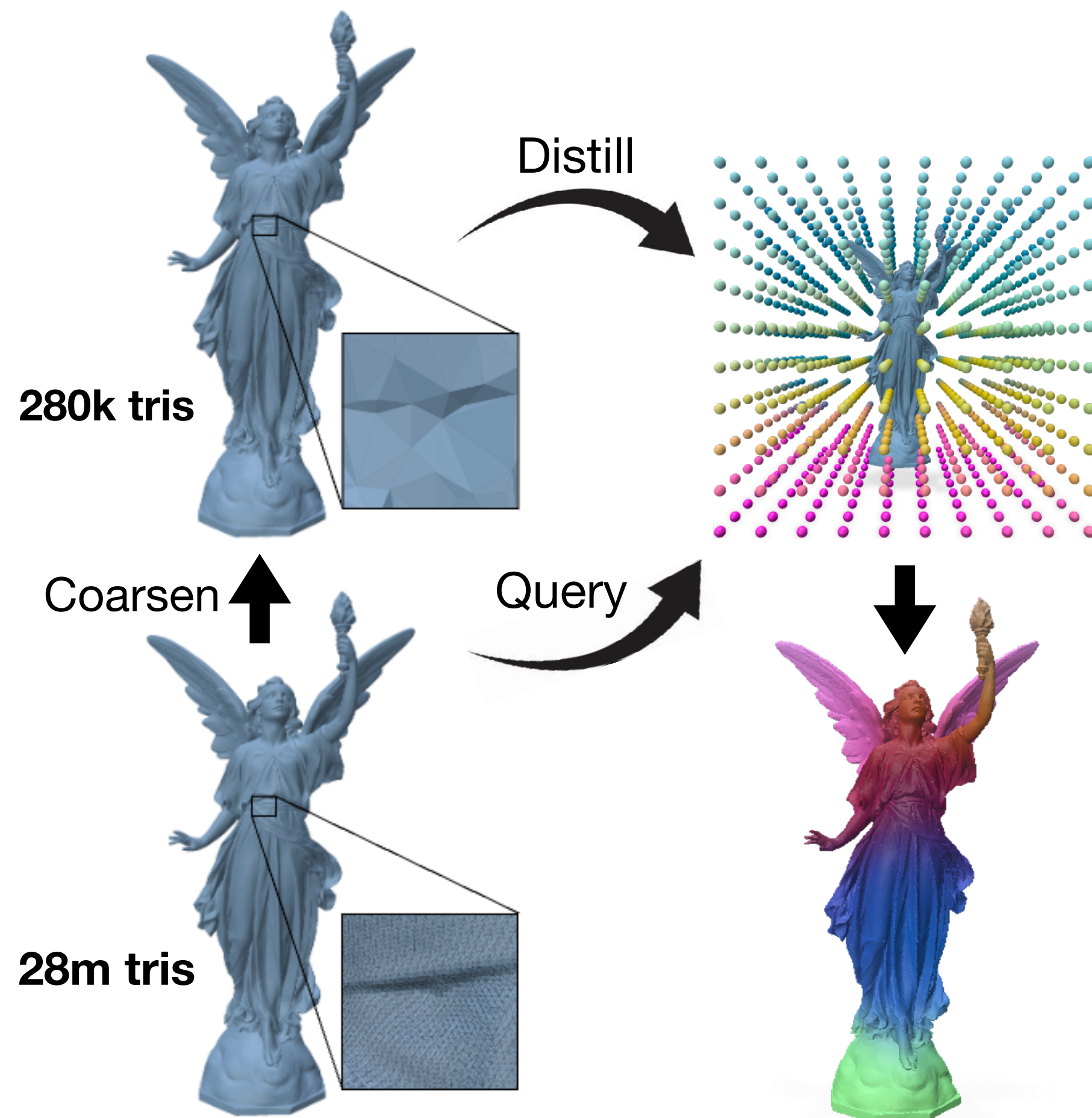
Transfer across remeshing



**“Deep Feature  
Deformation  
Weights”**  
Liu et al. 2026

# Feature Fields: Representation Agnostic

Transfer across remeshing



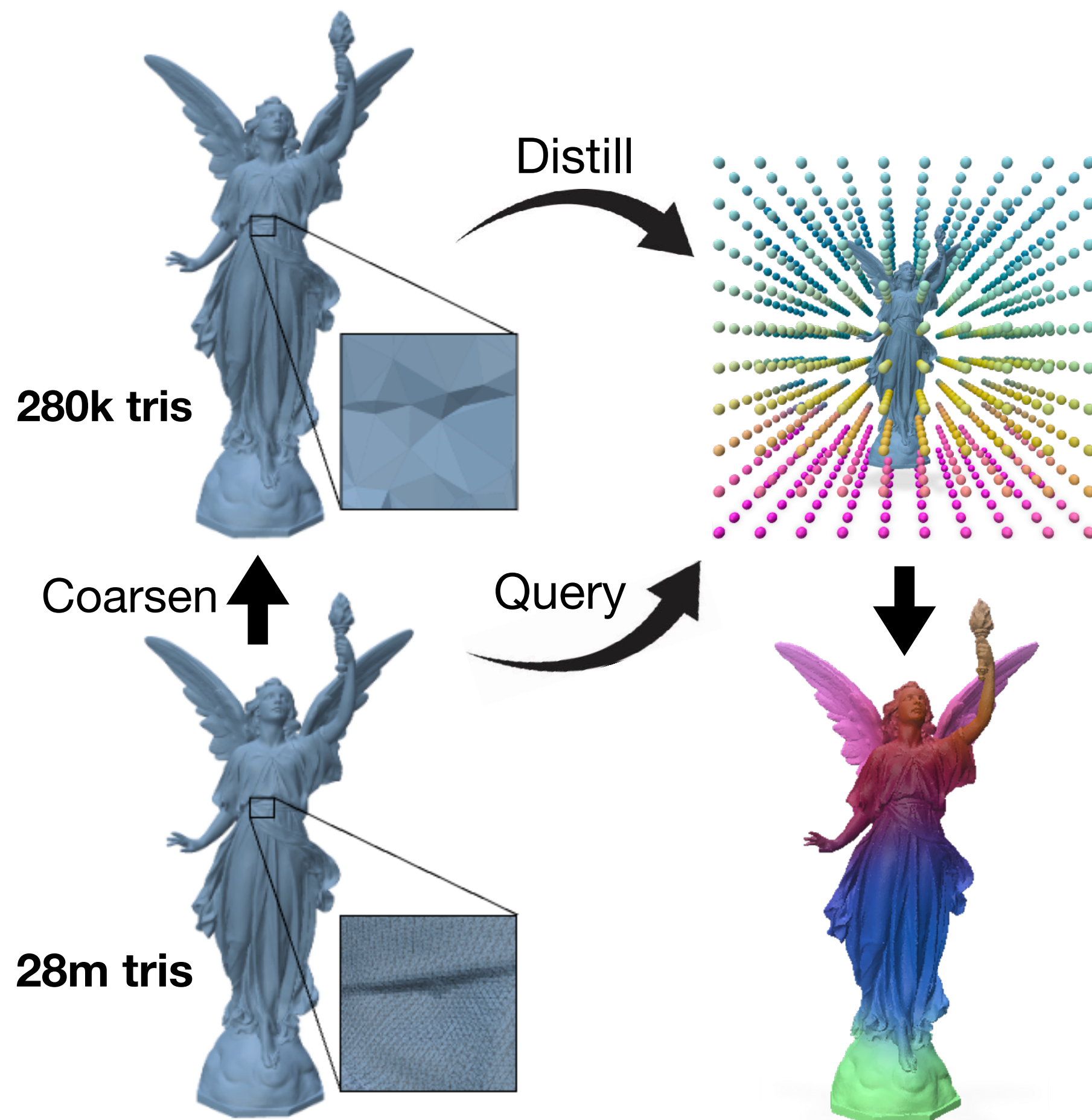
**“Deep Feature  
Deformation  
Weights”  
Liu et al. 2026**



# Feature Fields: Representation Agnostic

Transfer across remeshing

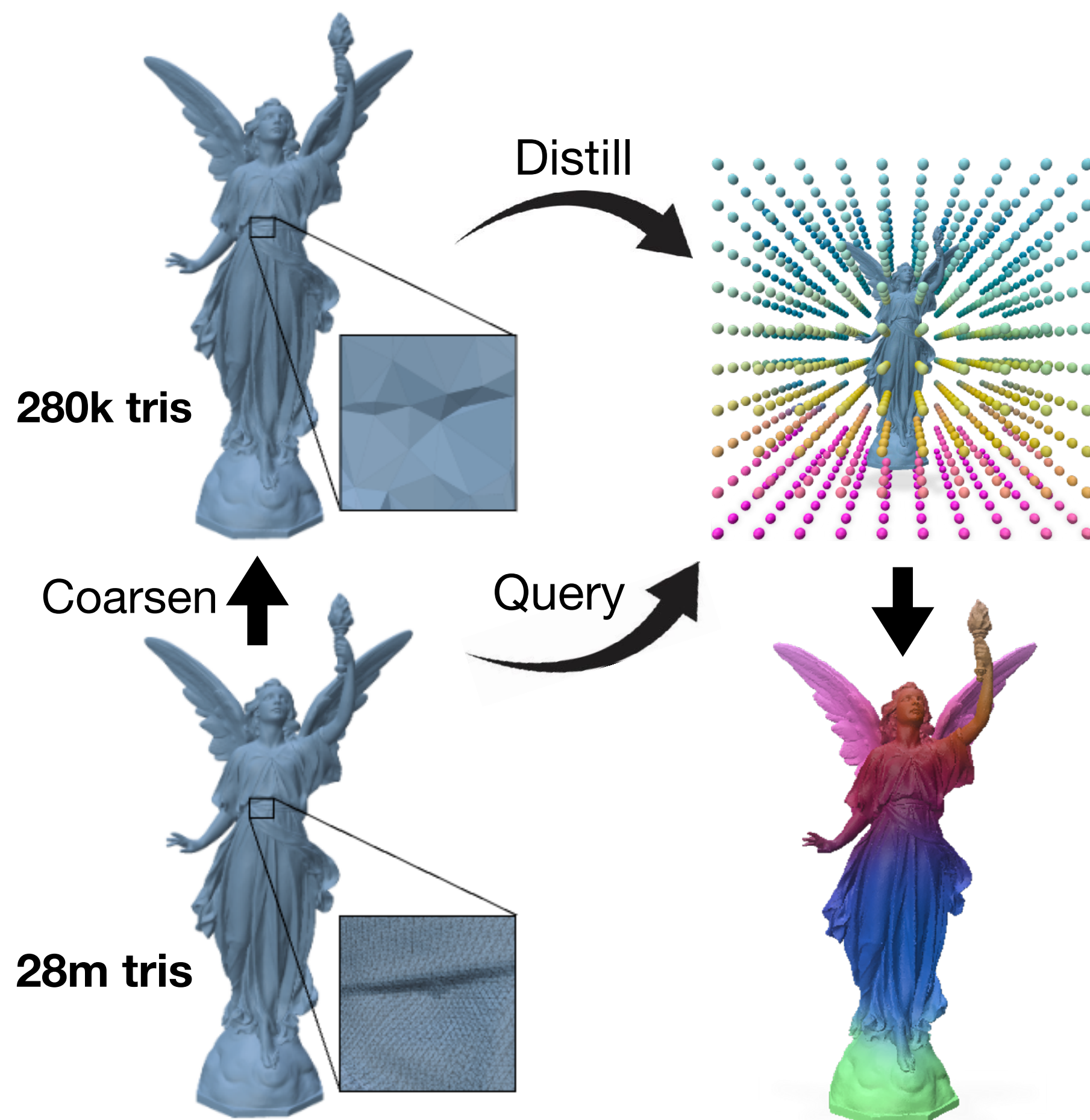
Transfer across representations



**“Deep Feature  
Deformation  
Weights”**  
Liu et al. 2026

# Feature Fields: Representation Agnostic

Transfer across remeshing



Transfer across representations

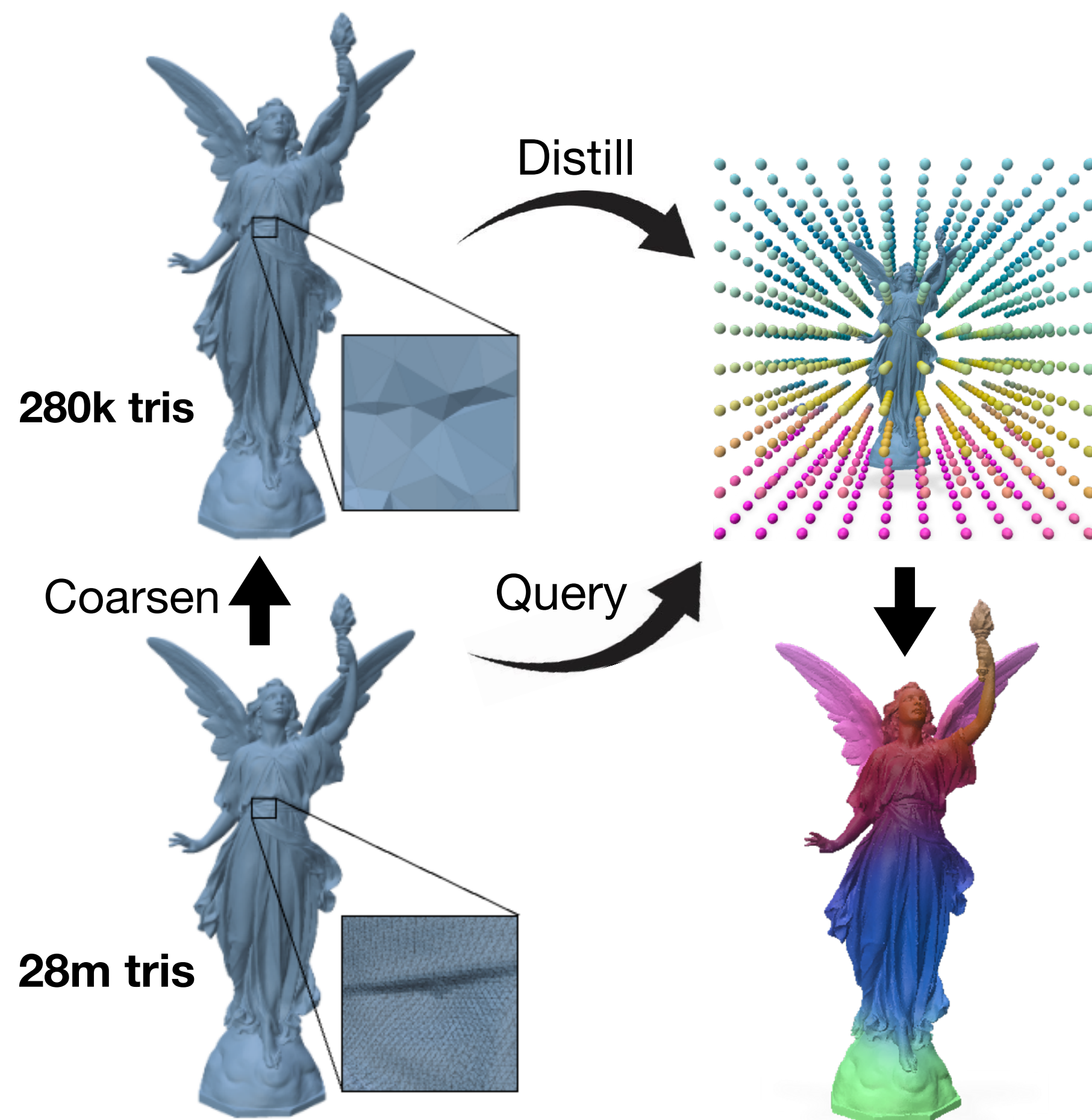


**“Deep Feature  
Deformation  
Weights”**  
Liu et al. 2026

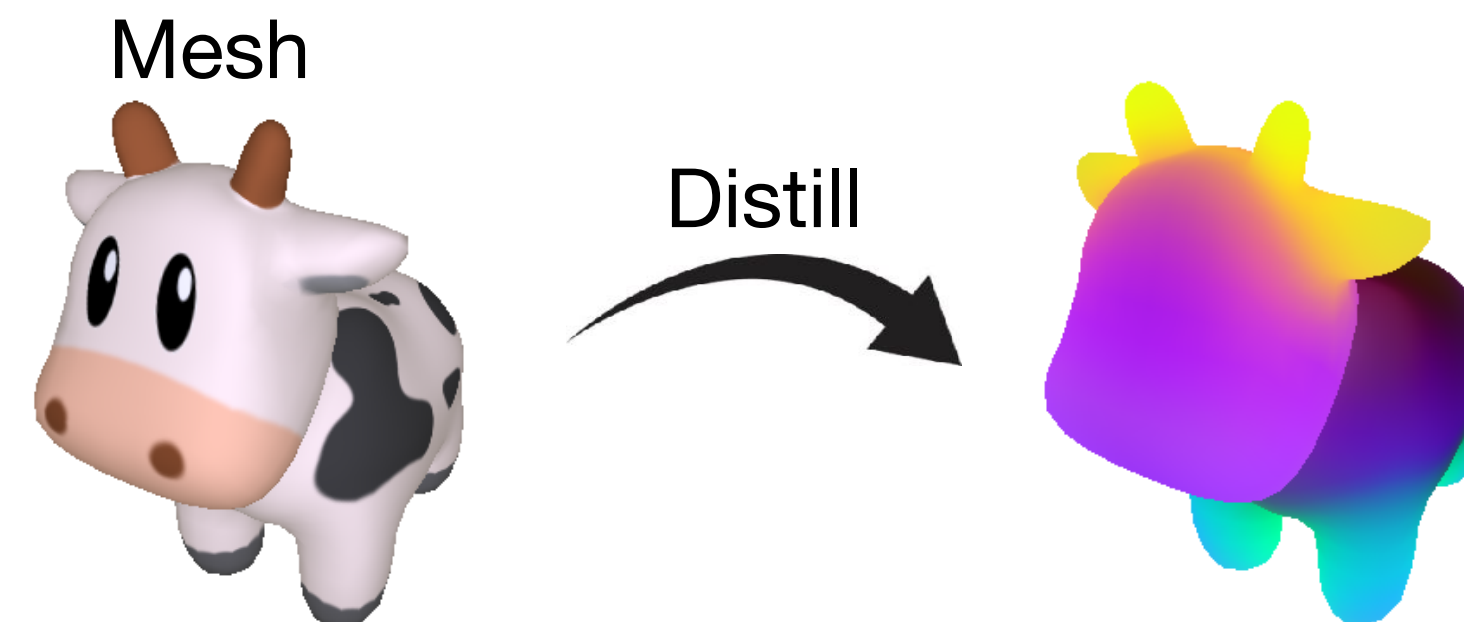


# Feature Fields: Representation Agnostic

## Transfer across remeshing



## Transfer across representations

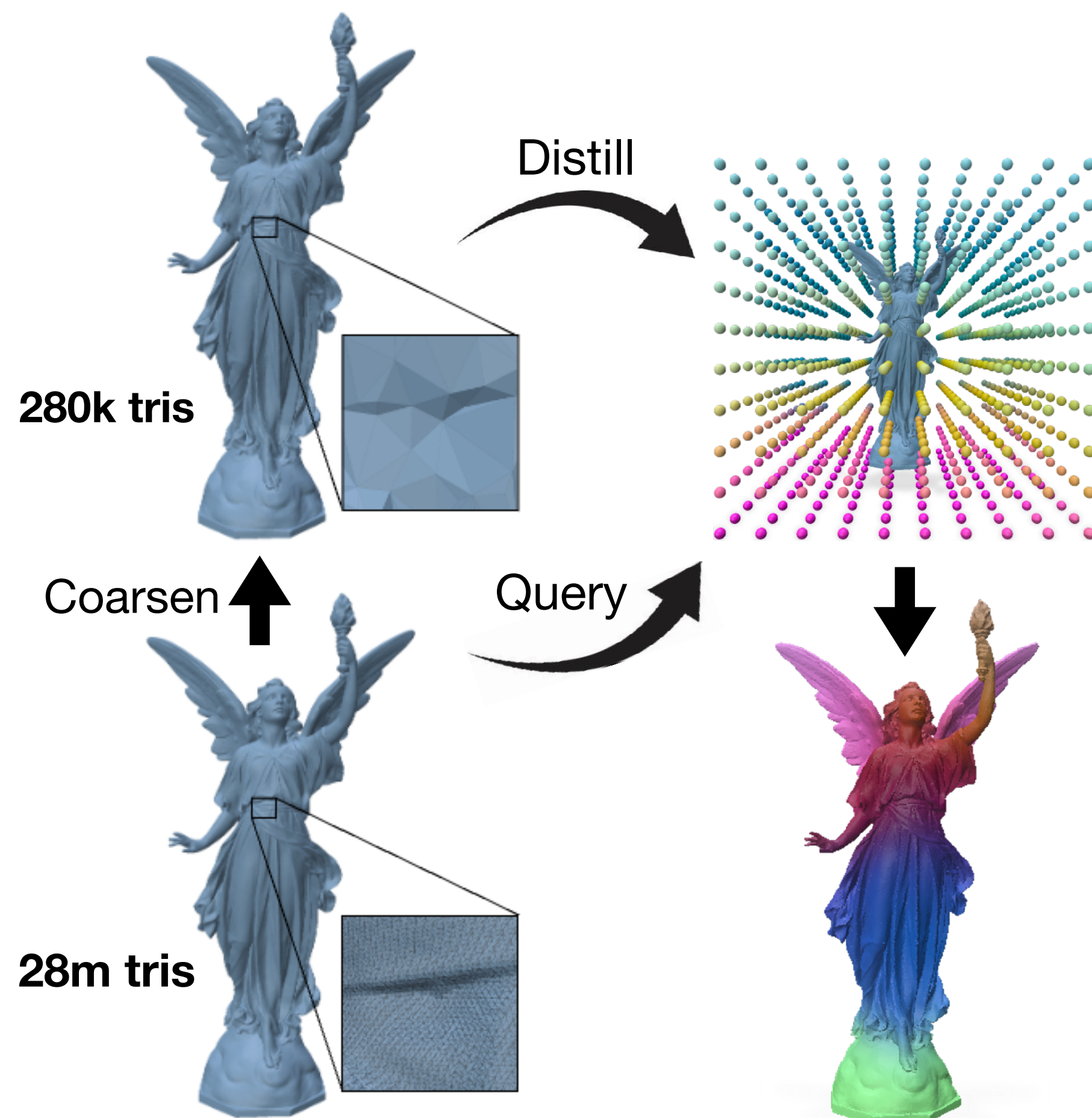


**“Deep Feature  
Deformation  
Weights”  
Liu et al. 2026**

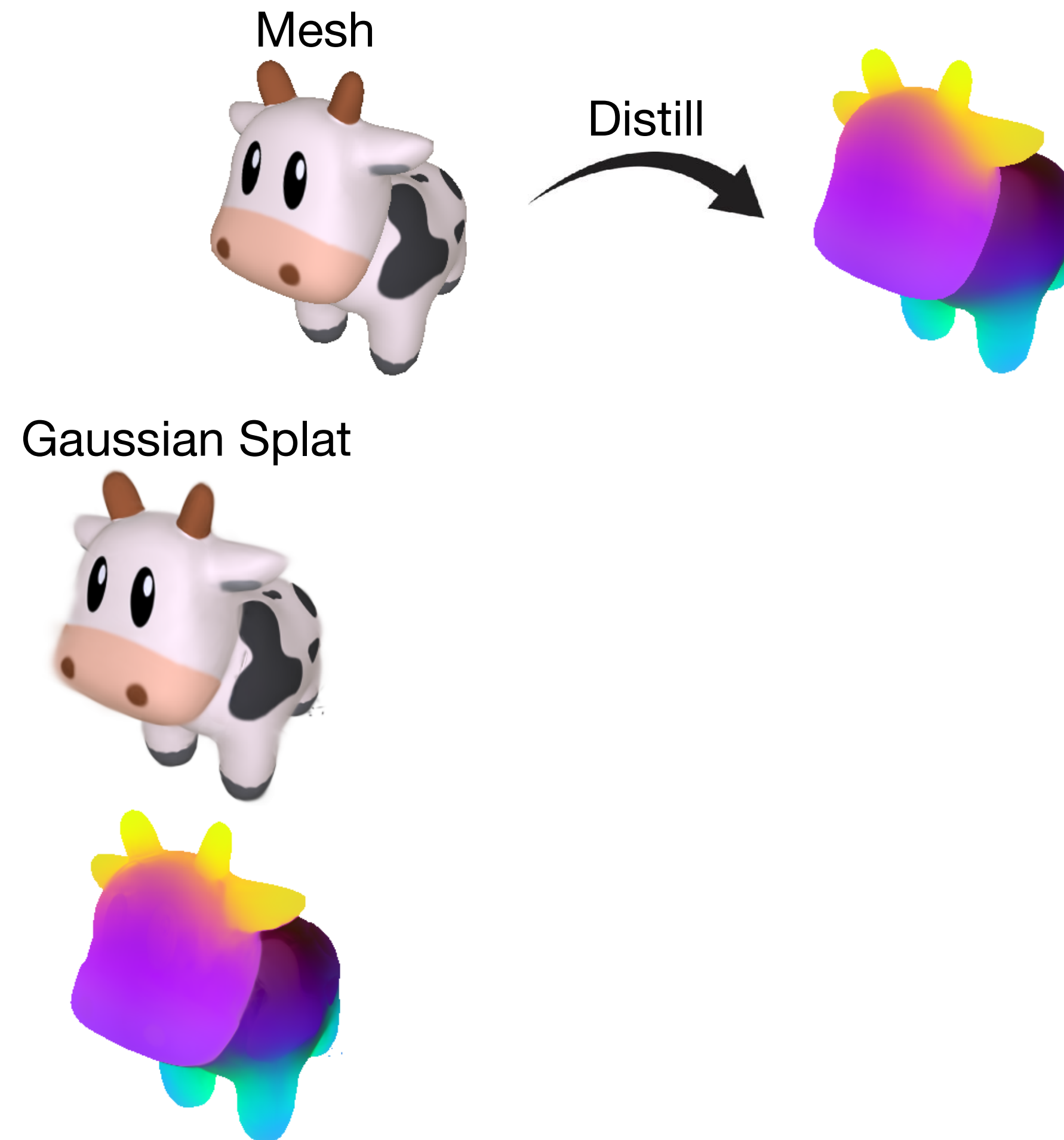


# Feature Fields: Representation Agnostic

## Transfer across remeshing



## Transfer across representations

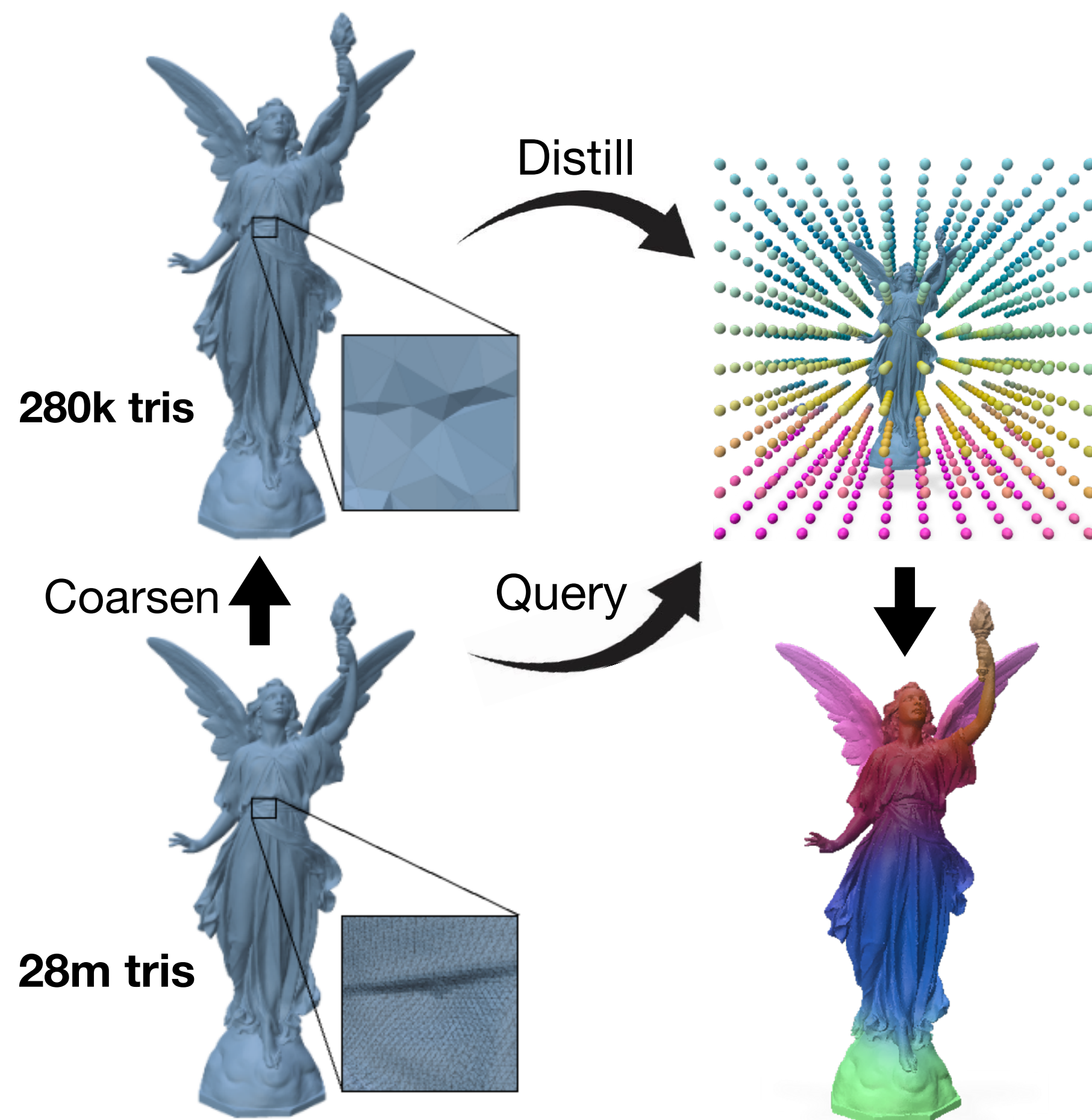


**“Deep Feature  
Deformation  
Weights”  
Liu et al. 2026**

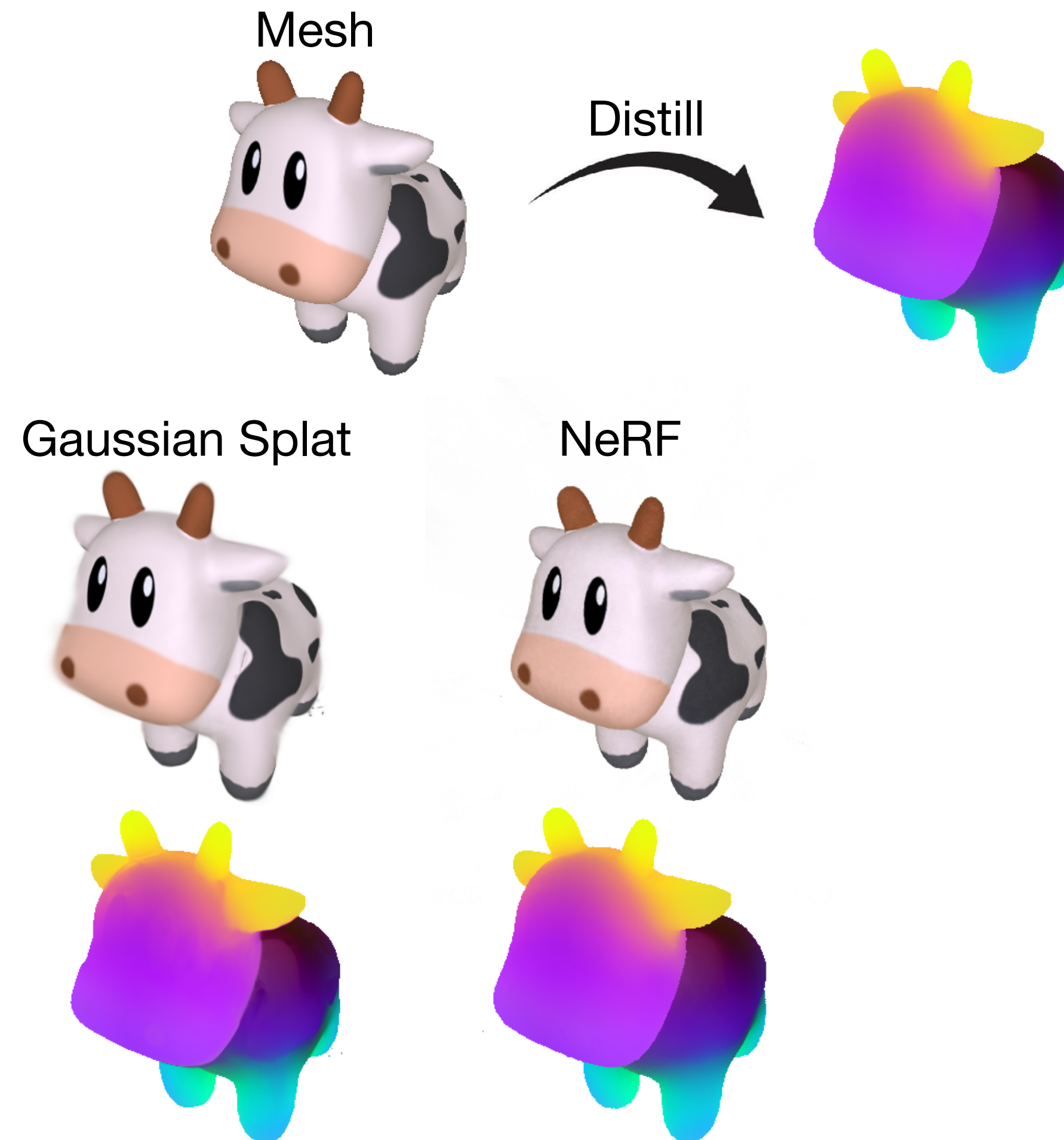


# Feature Fields: Representation Agnostic

## Transfer across remeshing



## Transfer across representations

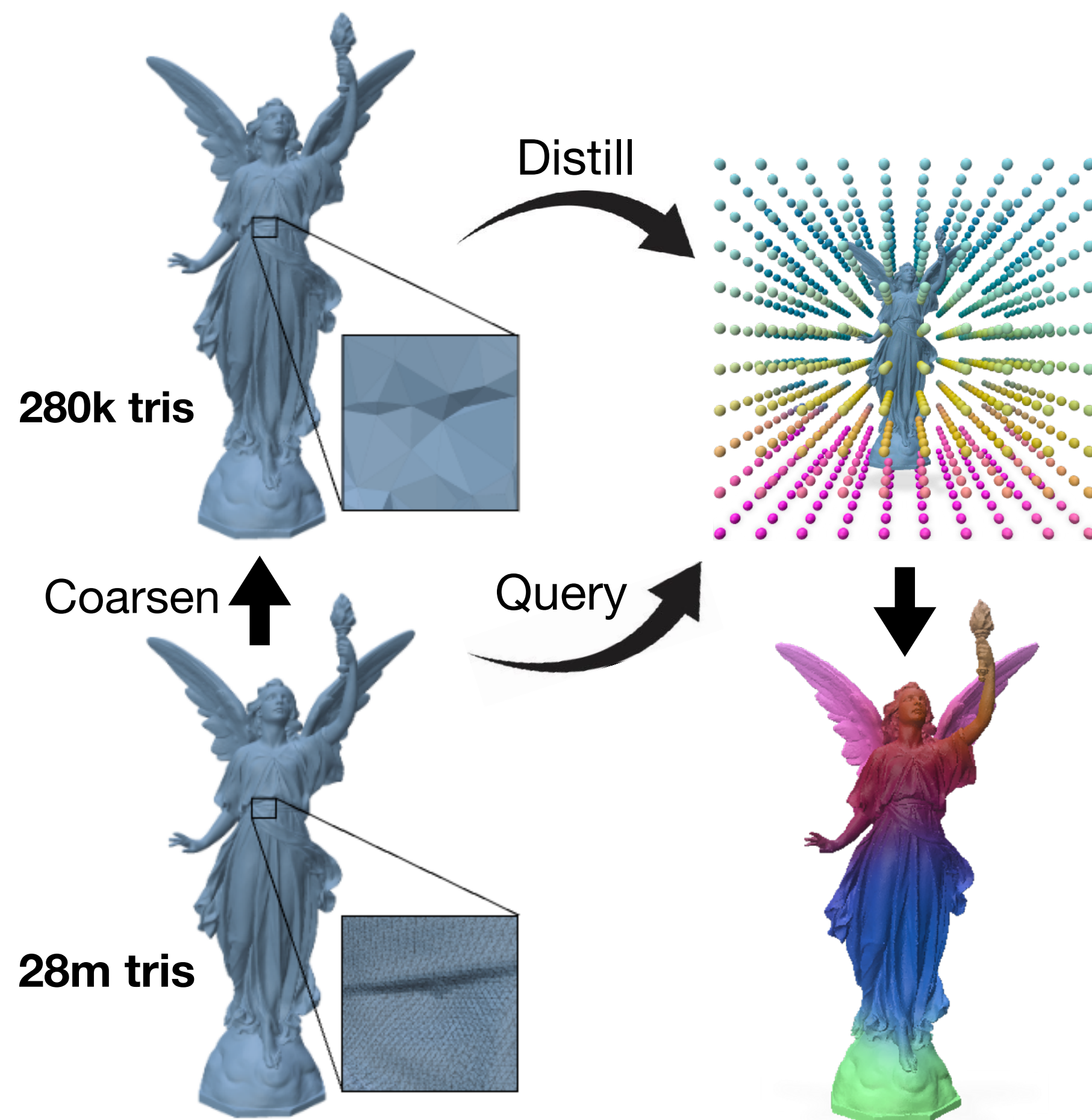


**“Deep Feature  
Deformation  
Weights”  
Liu et al. 2026**

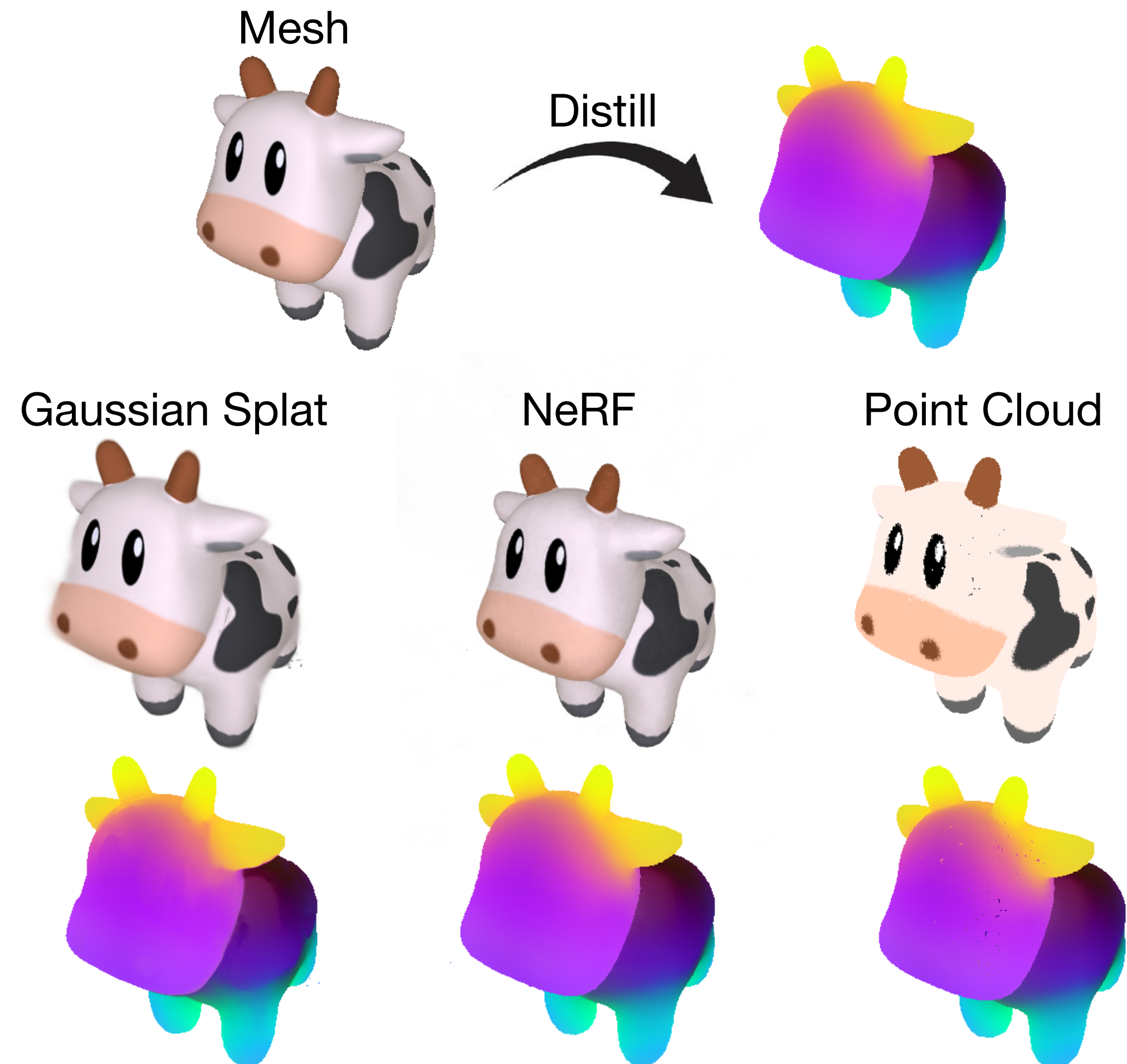


# Feature Fields: Representation Agnostic

## Transfer across remeshing



## Transfer across representations



**“Deep Feature  
Deformation  
Weights”  
Liu et al. 2026**



# Feature Fields: Unrestricted Query Domain

**“Deep Feature  
Deformation  
Weights”  
Liu et al. 2026**

# Feature Fields: Unrestricted Query Domain

Ambient space query: off-surface evaluation for symmetry detection

**“Deep Feature  
Deformation  
Weights”  
Liu et al. 2026**

# Feature Fields: Unrestricted Query Domain

Ambient space query: off-surface evaluation for symmetry detection

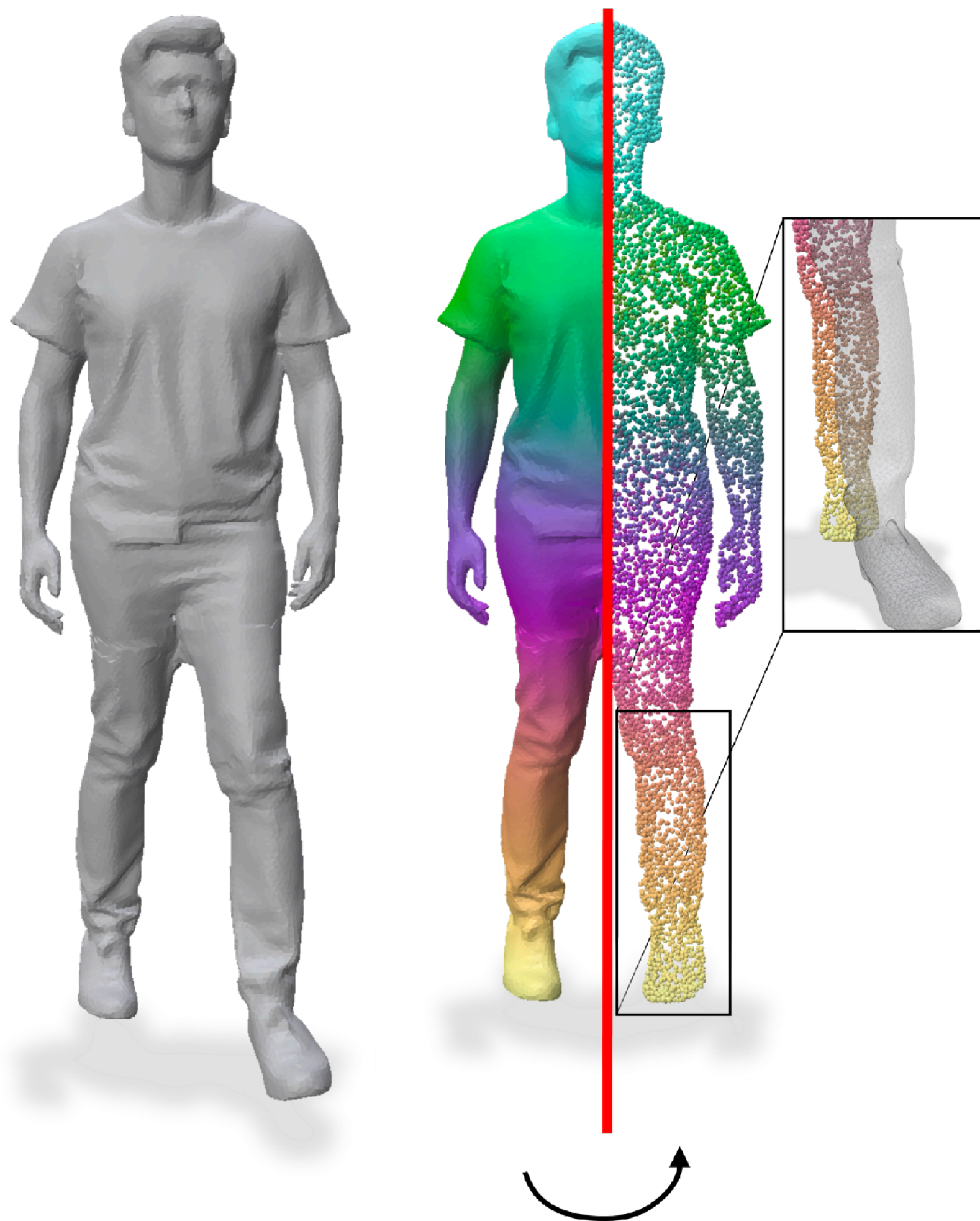


**“Deep Feature  
Deformation  
Weights”**  
Liu et al. 2026



# Feature Fields: Unrestricted Query Domain

Ambient space query: off-surface evaluation for symmetry detection

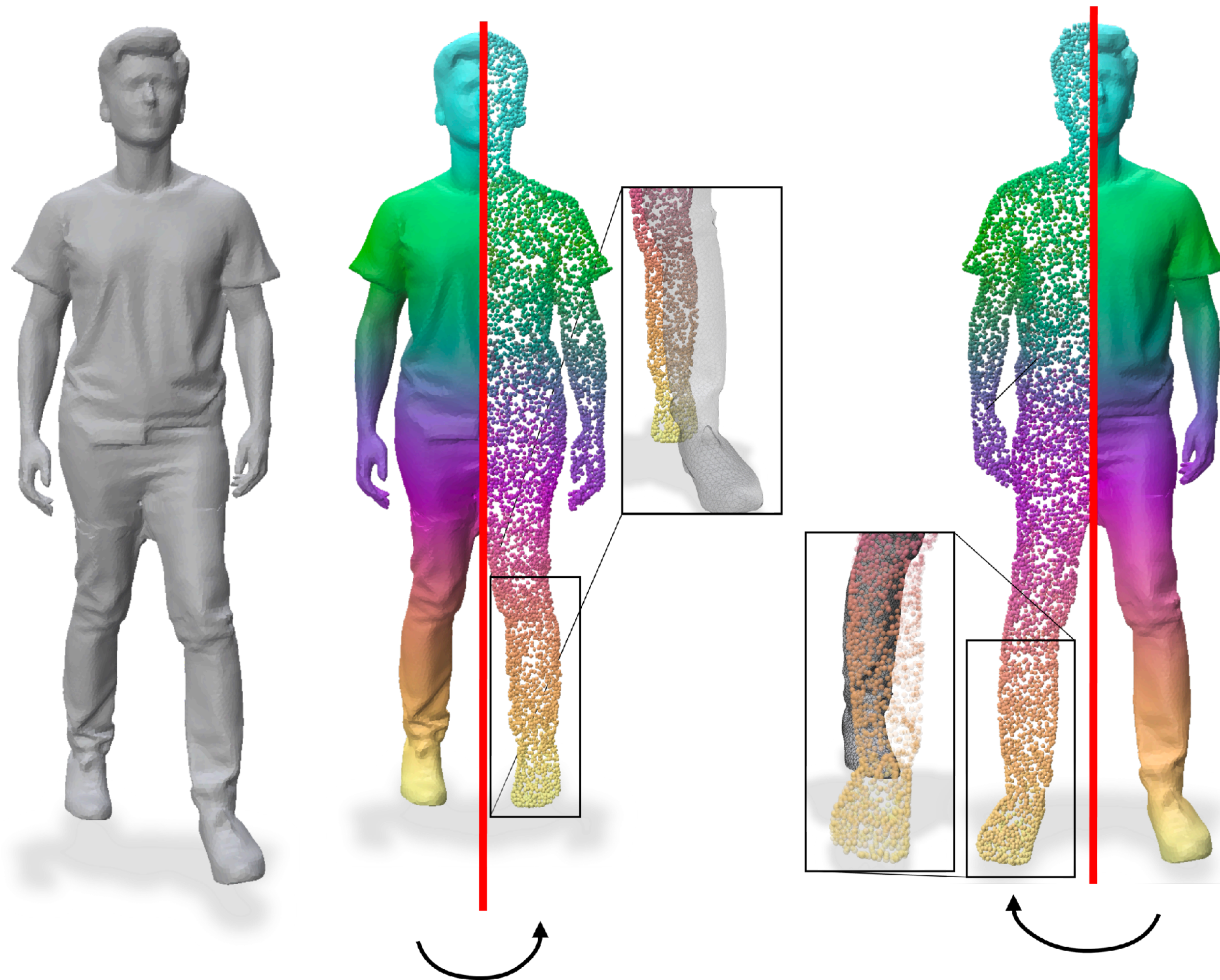


**“Deep Feature  
Deformation  
Weights”**  
Liu et al. 2026



# Feature Fields: Unrestricted Query Domain

Ambient space query: off-surface evaluation for symmetry detection

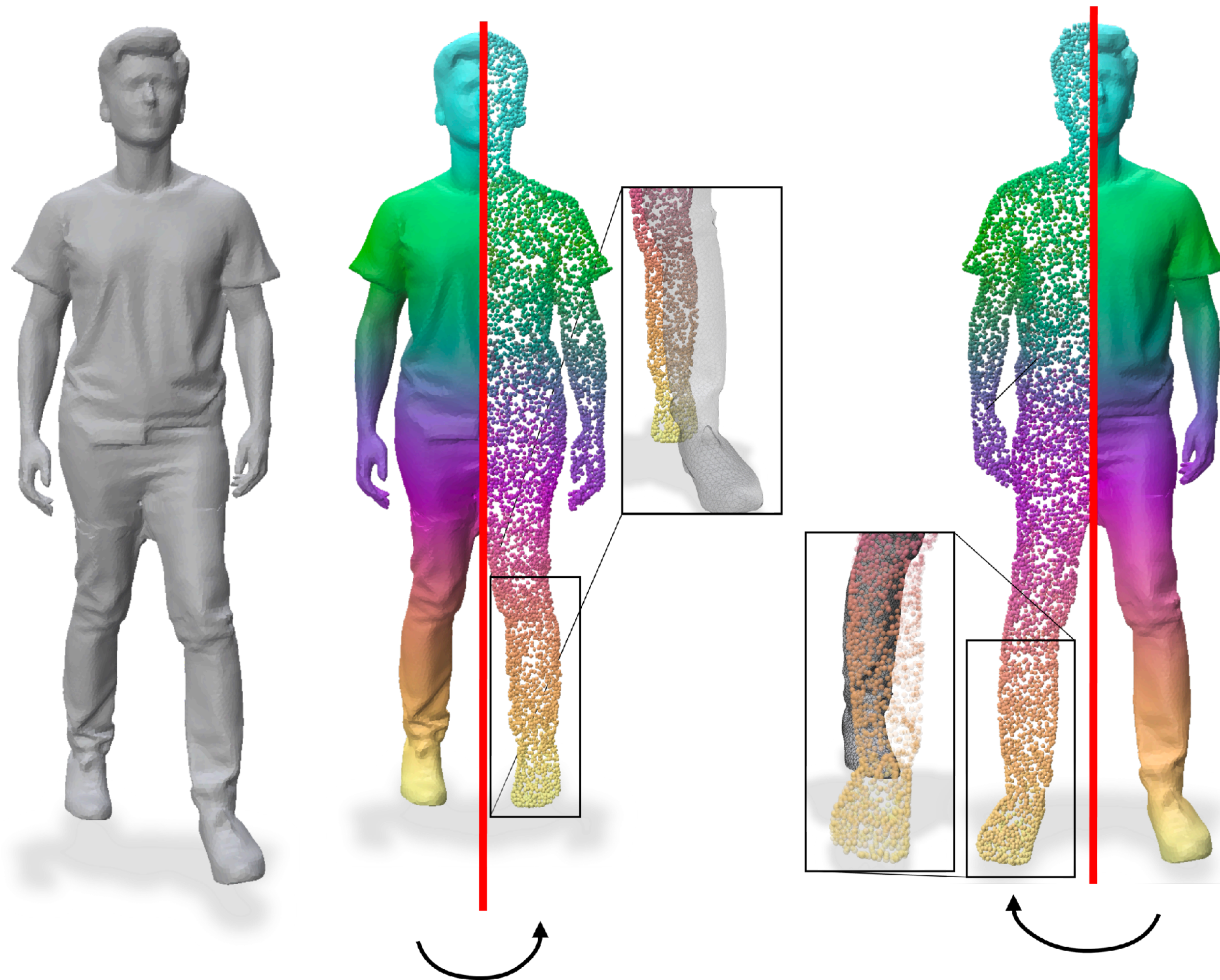


**“Deep Feature  
Deformation  
Weights”**  
Liu et al. 2026



# Feature Fields: Unrestricted Query Domain

Ambient space query: off-surface evaluation for symmetry detection



“Visual” symmetry: not captured by either extrinsic or intrinsic symmetry

**“Deep Feature  
Deformation  
Weights”**  
Liu et al. 2026



# Feature Fields: Unrestricted Query Domain

Texture space query: find all texels relevant to surface point

# Feature Fields: Unrestricted Query Domain

Texture space query: find all texels relevant to surface point



# Feature Fields: Unrestricted Query Domain

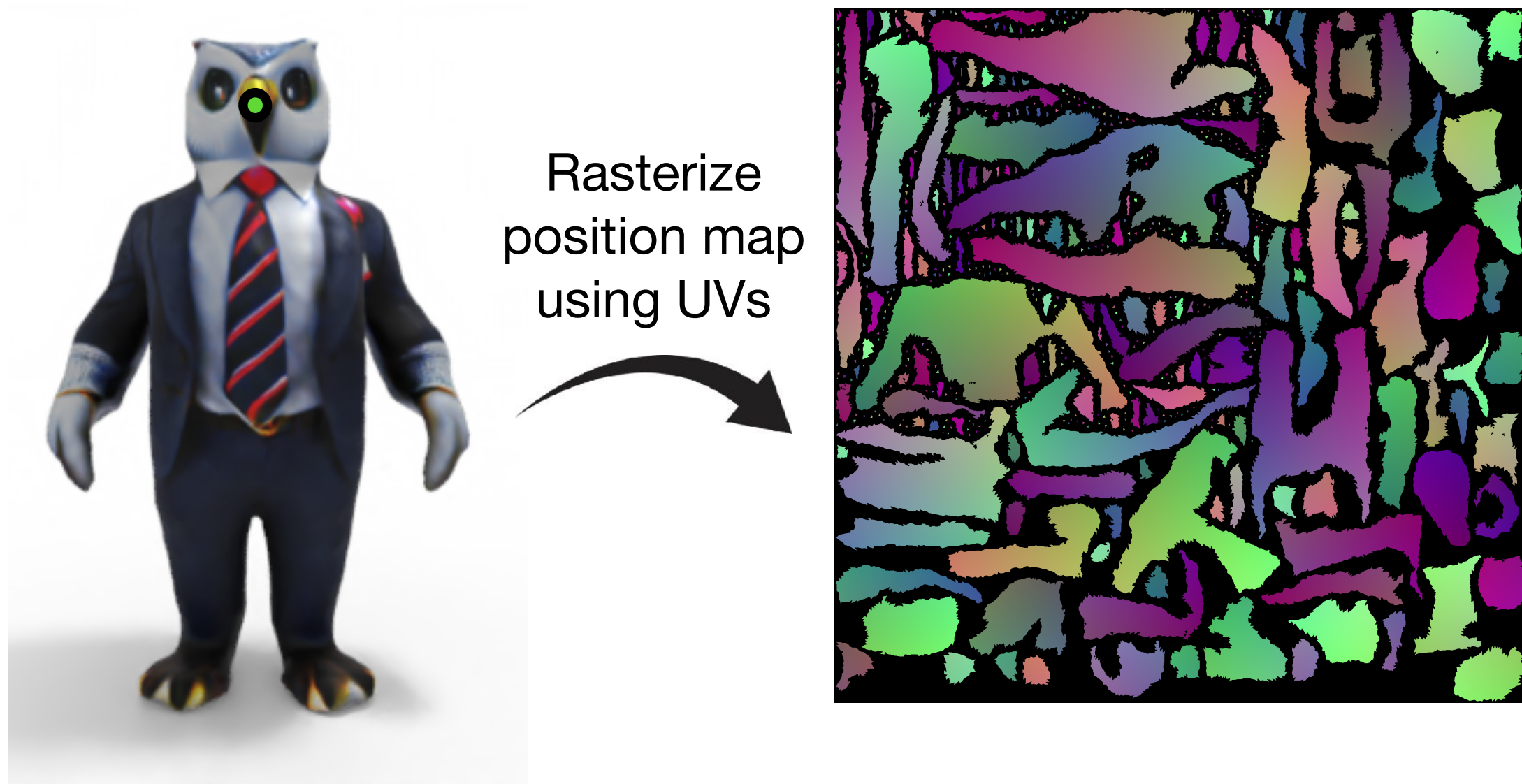
Texture space query: find all texels relevant to surface point





# Feature Fields: Unrestricted Query Domain

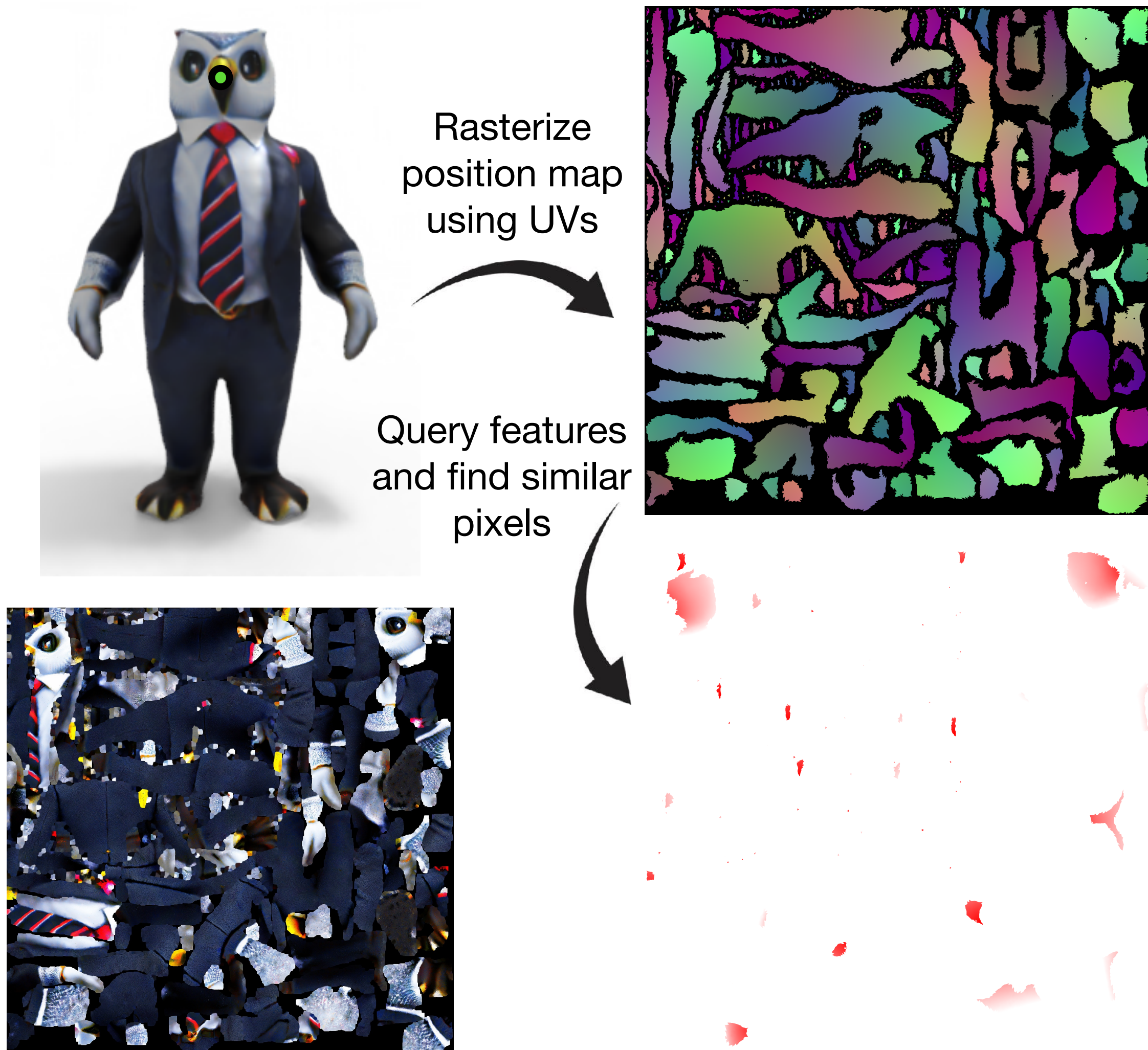
Texture space query: find all texels relevant to surface point





# Feature Fields: Unrestricted Query Domain

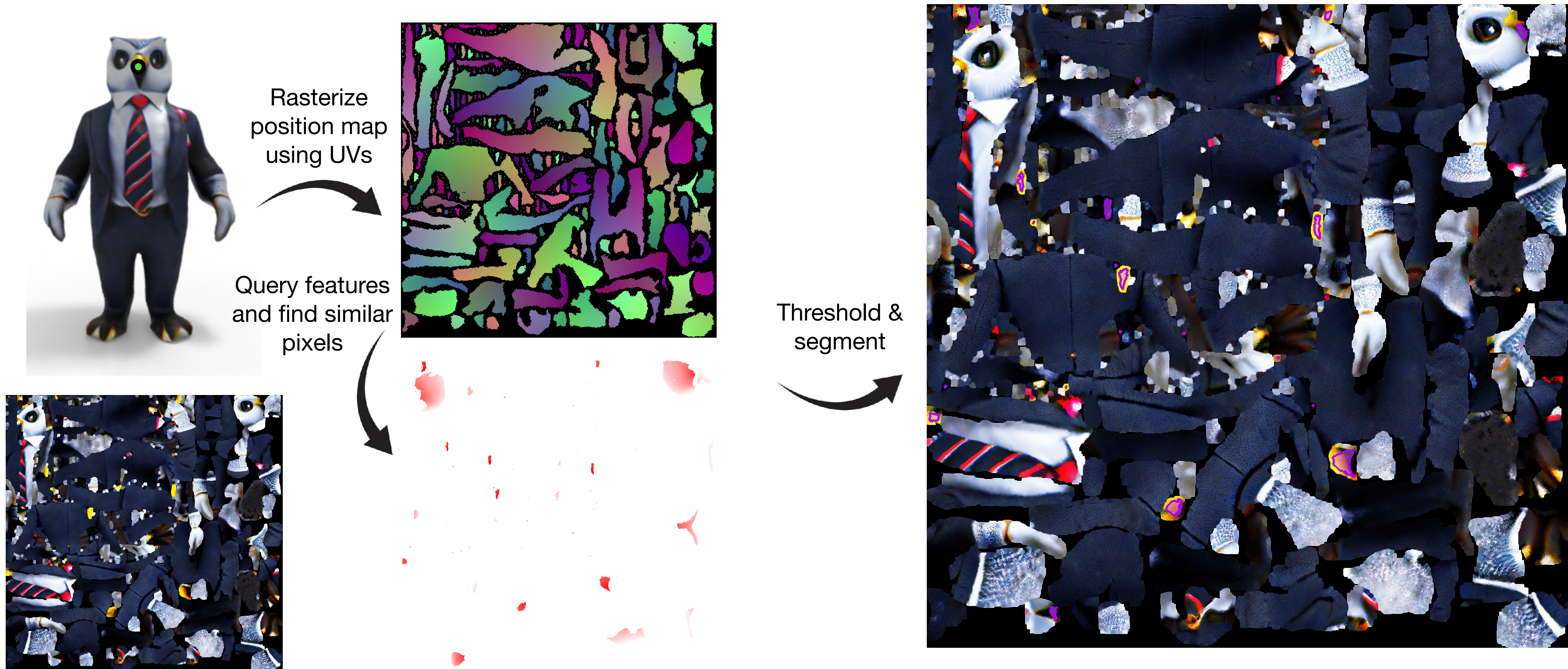
Texture space query: find all texels relevant to surface point





# Feature Fields: Unrestricted Query Domain

Texture space query: find all texels relevant to surface point





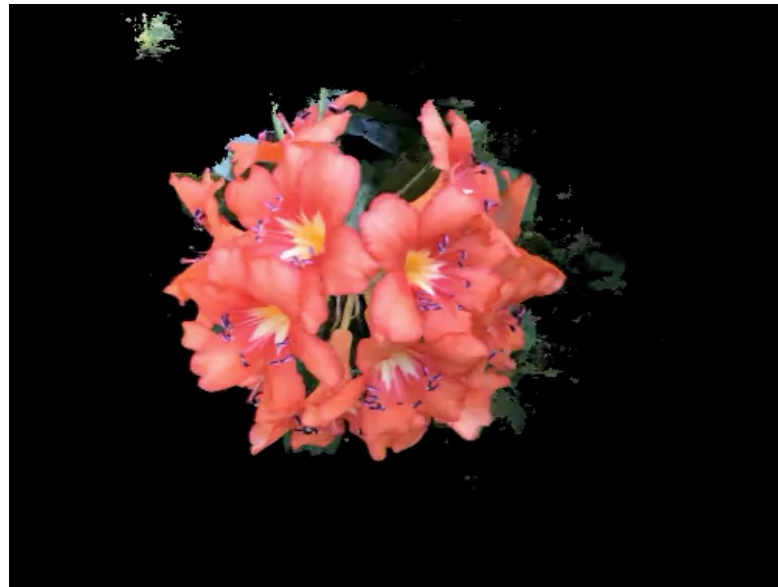
# 2D Feature Applications in 3D

## Editing

NeRF

Extraction

Deletion



*Decomposing NeRF for Editing via Feature Field Distillation*  
Kobayashi et al. 2022

## Generation

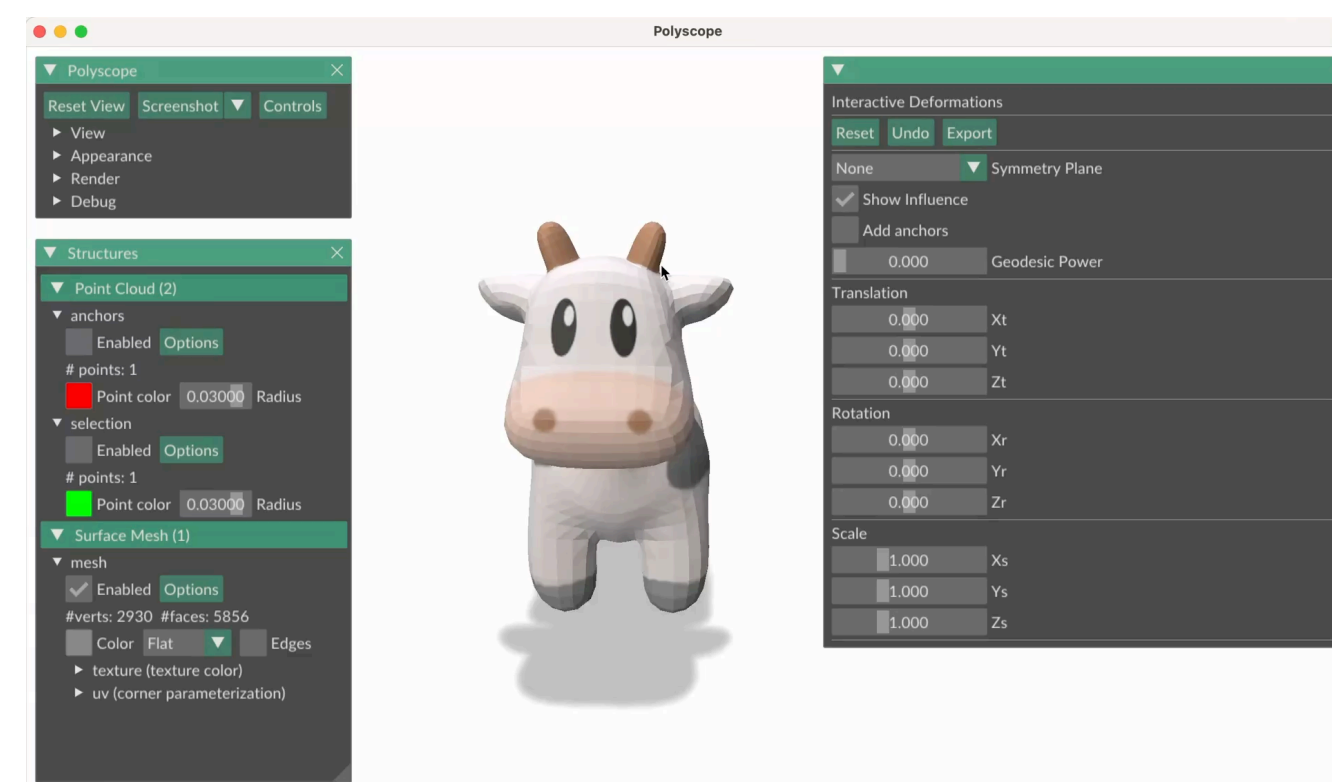
3D Assets Encoding & Decoding

Structured Latent Representation Learning



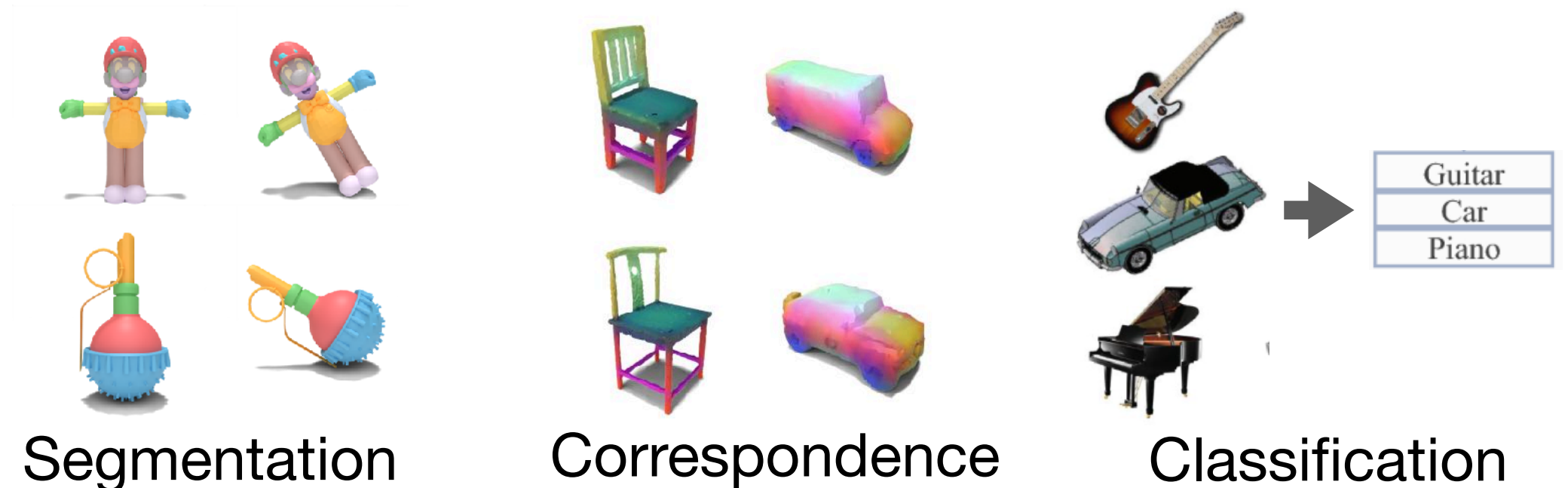
*Structured 3D Latents for Scalable and Versatile 3D Generation*  
Xiang et al. 2025

## Deformation



*Deep Feature Deformation Weights*  
Liu et al. 2026

## Zero-Shot Traditional Tasks



*2D Features Are All You Need for 3D Shape Understanding*  
Zhou et al. 2026



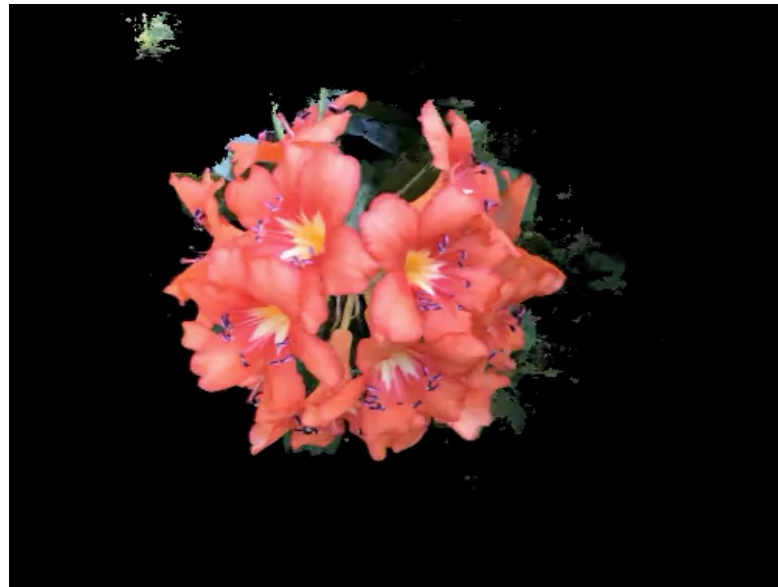
# 2D Feature Applications in 3D

## Editing

NeRF

Extraction

Deletion



*Decomposing NeRF for Editing via Feature Field Distillation*  
Kobayashi et al. 2022

## Generation

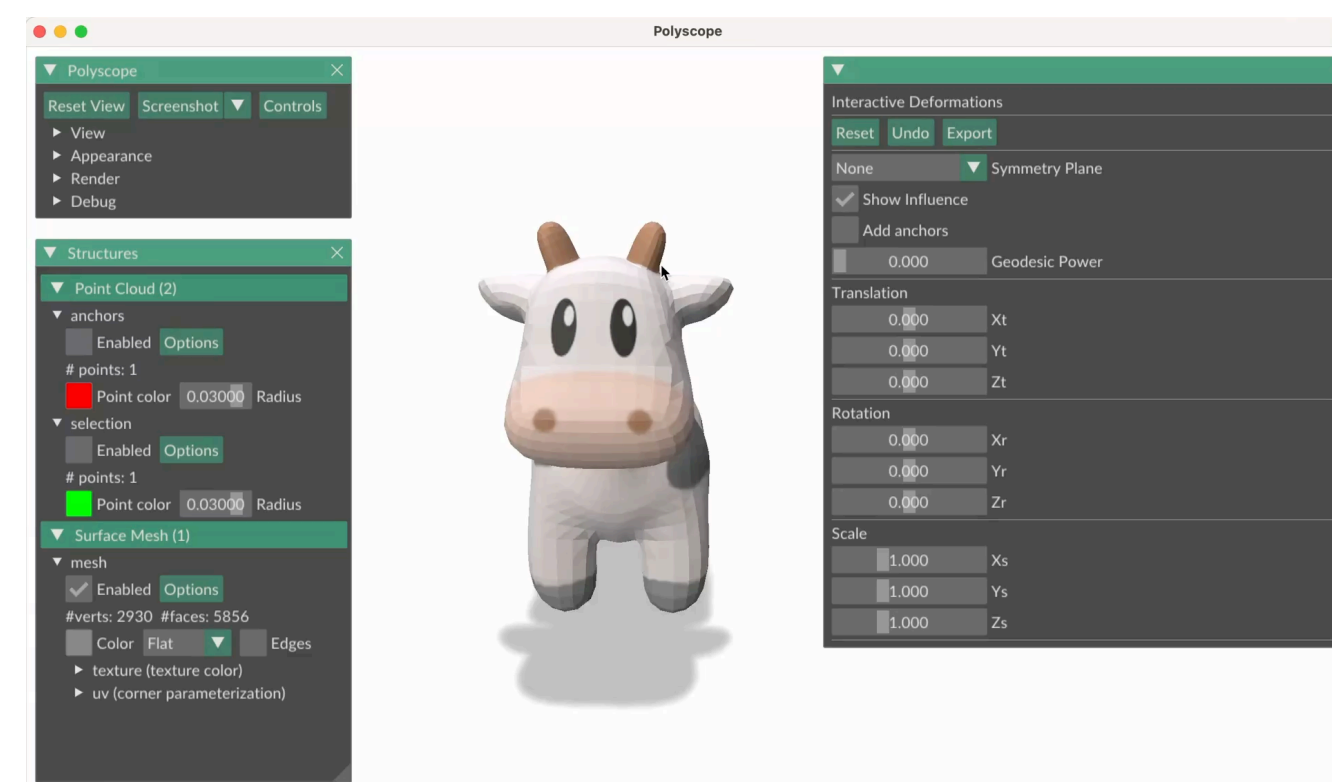
3D Assets Encoding & Decoding

Structured Latent Representation Learning



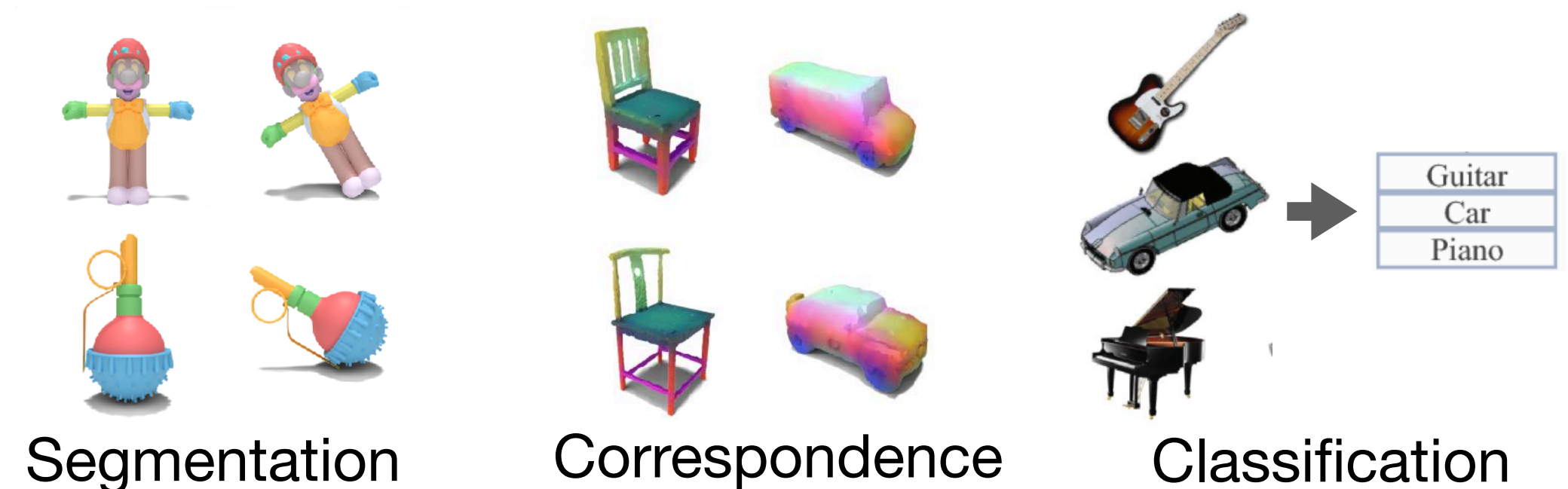
*Structured 3D Latents for Scalable and Versatile 3D Generation*  
Xiang et al. 2025

## Deformation



*Deep Feature Deformation Weights*  
Liu et al. 2026

## Zero-Shot Traditional Tasks



*2D Features Are All You Need for 3D Shape Understanding*  
Zhou et al. 2026



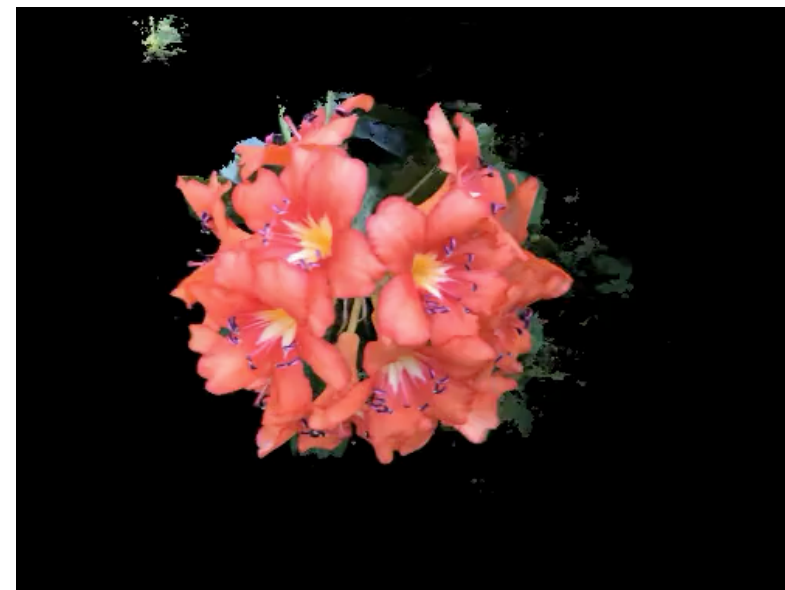
# 2D Feature Applications in 3D

## Editing

NeRF

Extraction

Deletion



*Decomposing NeRF for Editing via Feature Field Distillation*  
Kobayashi et al. 2022

## Generation

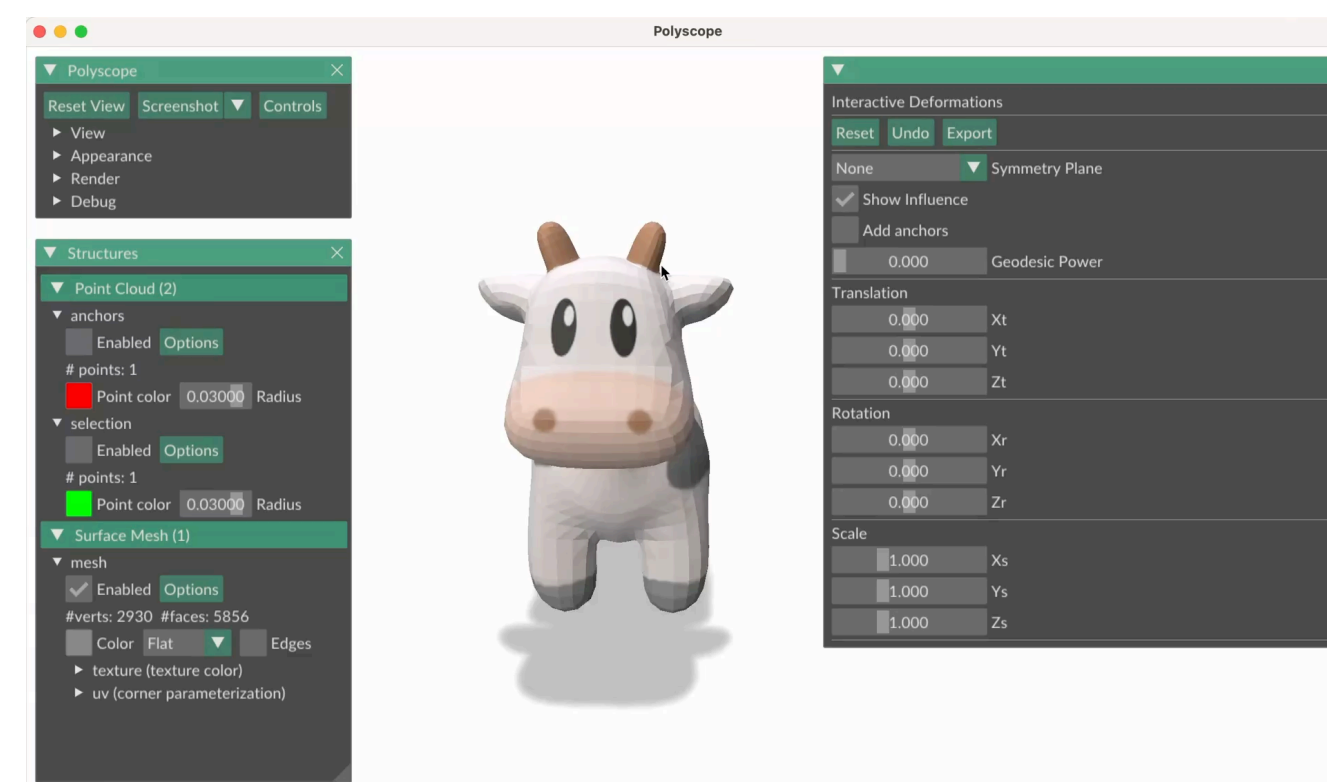
3D Assets Encoding & Decoding

Structured Latent Representation Learning



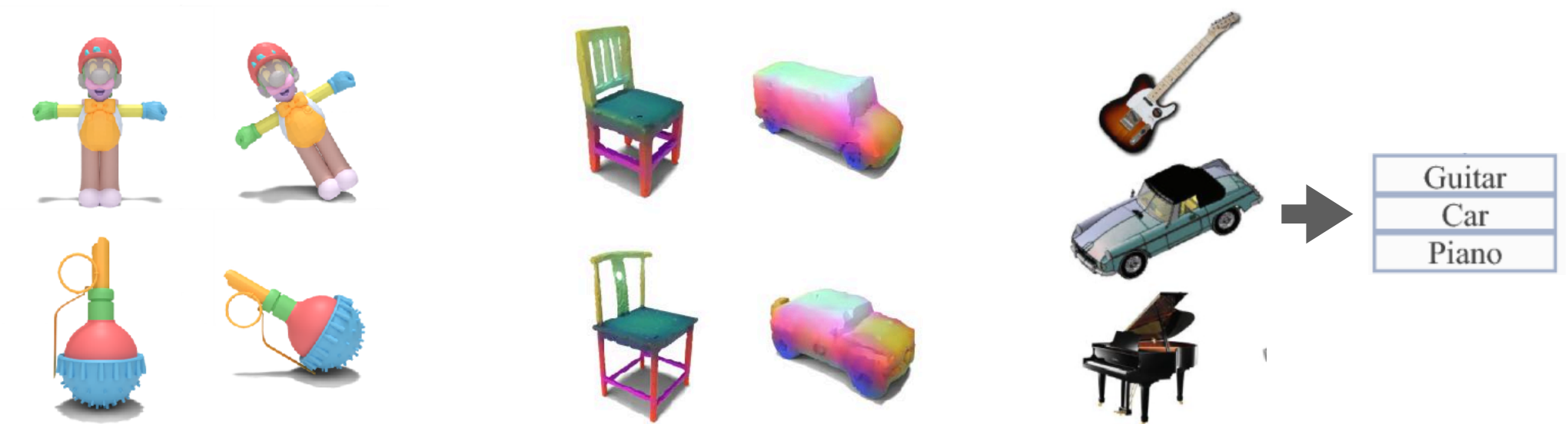
*Structured 3D Latents for Scalable and Versatile 3D Generation*  
Xiang et al. 2025

## Deformation



*Deep Feature Deformation Weights*  
Liu et al. 2026

## Zero-Shot Traditional Tasks



Segmentation

Correspondence

Classification

*2D Features Are All You Need for 3D Shape Understanding*  
Zhou et al. 2026

**More to come!**



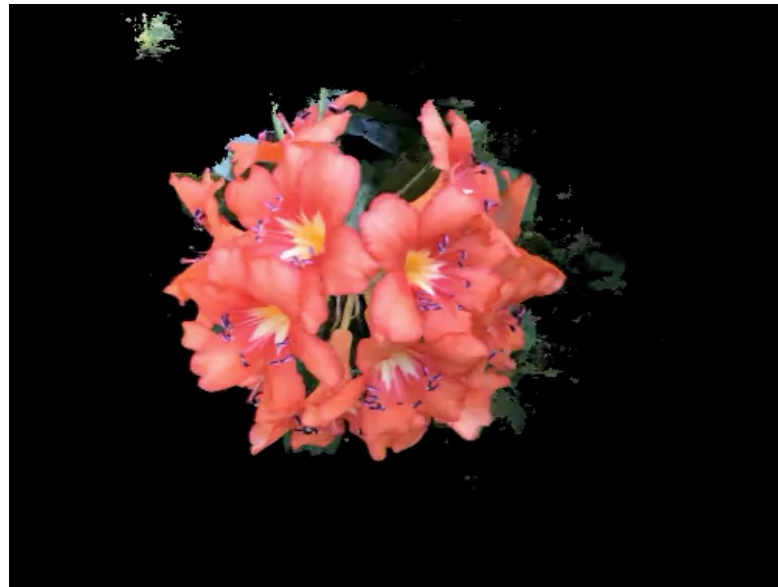
# 2D Feature Applications in 3D

## Editing

NeRF

Extraction

Deletion



*Decomposing NeRF for Editing via Feature Field Distillation*  
Kobayashi et al. 2022

## Generation

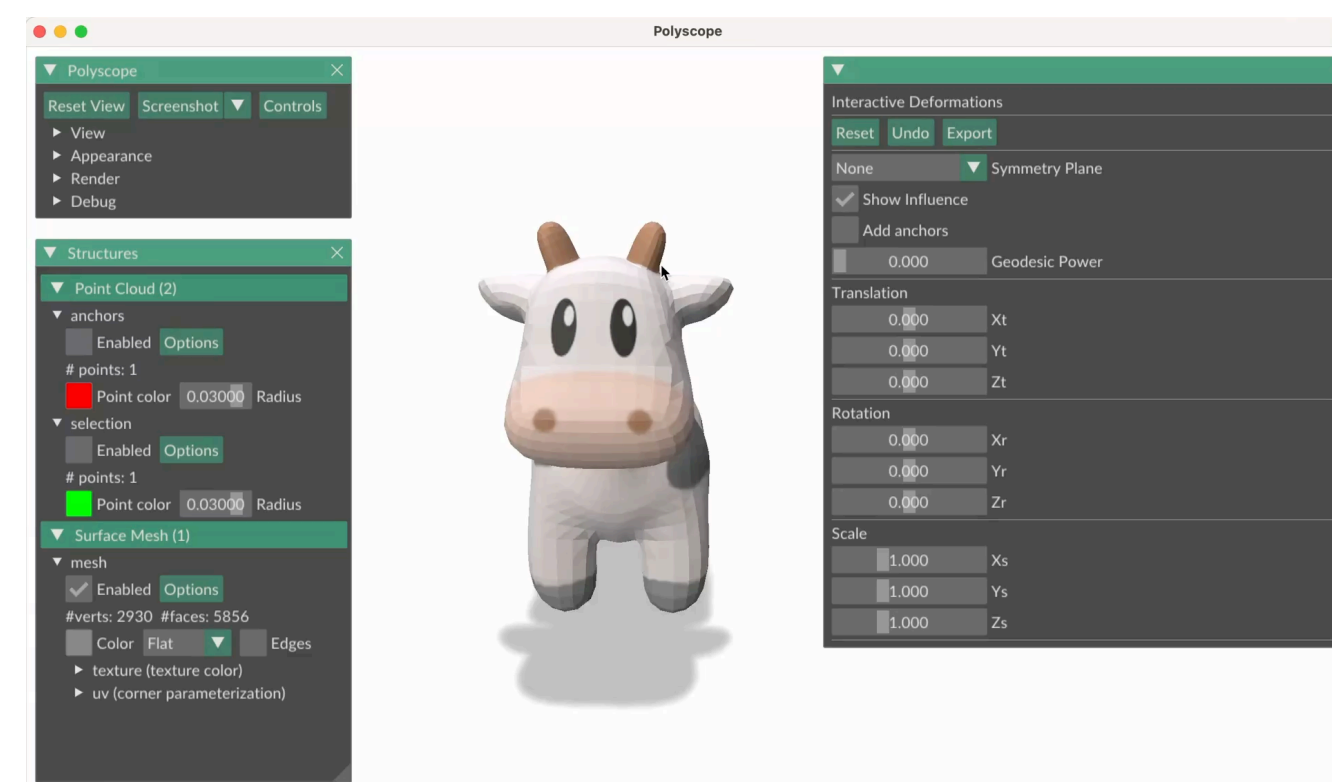
3D Assets Encoding & Decoding

Structured Latent Representation Learning



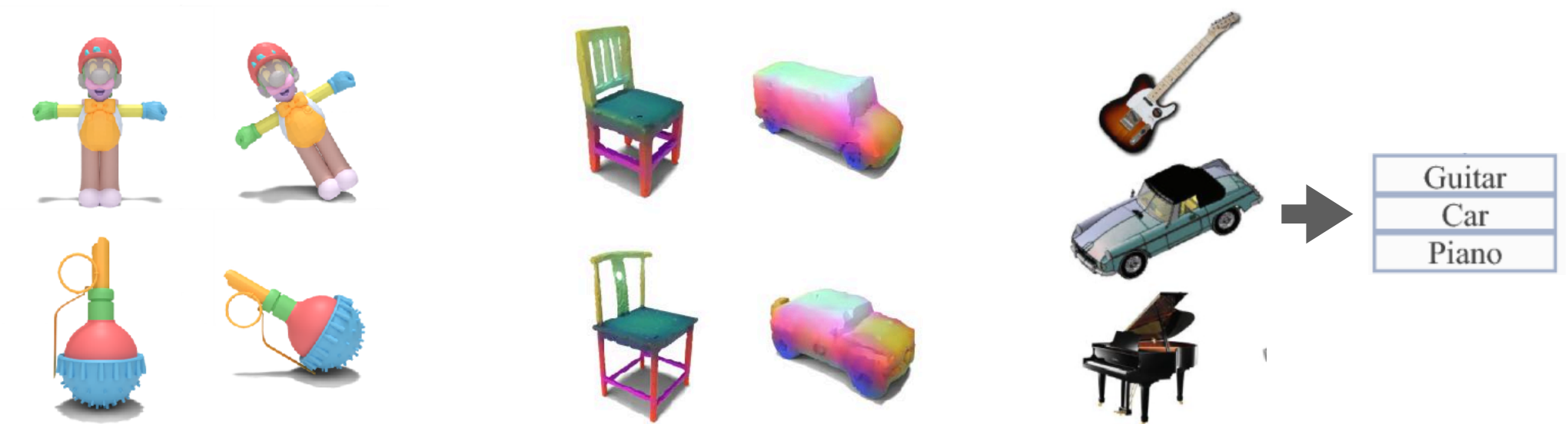
*Structured 3D Latents for Scalable and Versatile 3D Generation*  
Xiang et al. 2025

## Deformation



*Deep Feature Deformation Weights*  
Liu et al. 2026

## Zero-Shot Traditional Tasks



Segmentation

Correspondence

Classification

*2D Features Are All You Need for 3D Shape Understanding*  
Zhou et al. 2026

**More to come!**

# Deep Features for Linear Blend Skinning

$$V'_i = \sum_{k=1}^K W_{ik} D_k V_i$$

$V'_i / V_i$  Deformed vertex / starting vertex

$D_k$  Deformation for control structure k

$W_{ik}$  Blending weight  
(defines deformation subspace)

# Deep Features for Linear Blend Skinning

$$V'_i = \sum_{k=1}^K W_{ik} D_k V_i$$

$V'_i / V_i$  Deformed vertex / starting vertex  
 $D_k$  Deformation for control structure k  
 $W_{ik}$  Blending weight  
(defines deformation subspace)

**Control structures:** point/region handles, skeleton rig, cage nodes, etc ...



# Deep Features for Linear Blend Skinning

$$V'_i = \sum_{k=1}^K W_{ik} D_k V_i$$

$V'_i / V_i$  Deformed vertex / starting vertex  
 $D_k$  Deformation for control structure k  
 $W_{ik}$  Blending weight  
(defines deformation subspace)

**Control structures:** **point**/region handles, skeleton rig, cage nodes, etc ...

Weights sometimes blend control structure *positions* rather than *deformations*,  
but we collapse that difference for simplicity.

# Deep Features for Linear Blend Skinning

$$V'_i = \sum_{k=1}^K W_{ik} D_k V_i$$

$V'_i / V_i$  Deformed vertex / starting vertex  
 $D_k$  Deformation for control structure k  
 $W_{ik}$  Blending weight  
 (defines deformation subspace)

**Control structures:** **point**/region handles, skeleton rig, cage nodes, etc ...

Weights sometimes blend control structure *positions* rather than *deformations*, but we collapse that difference for simplicity.

Obtaining  $W$  is expensive and is dependent on the specific control structure set.

# Deep Features for Linear Blend Skinning

$$V'_i = \sum_{k=1}^K W_{ik} D_k V_i$$

$V'_i / V_i$  Deformed vertex / starting vertex  
 $D_k$  Deformation for control structure k  
 $W_{ik}$  Blending weight  
(defines deformation subspace)

## Deep Feature Deformation Weights

Every surface point / vertex can be a control handle



# Deep Features for Linear Blend Skinning

$$V'_i = \sum_{k=1}^K W_{ik} D_k V_i$$

$V'_i / V_i$  Deformed vertex / starting vertex  
 $D_k$  Deformation for control structure k  
 $W_{ik}$  Blending weight  
 (defines deformation subspace)

## Deep Feature Deformation Weights

Every surface point / vertex can be a control handle

$$W_{ik} = 1 - ||\Phi(V_i) - \Phi(H_k)||_2$$

$\Phi$  Feature field       $H_k$  Point handle k

# Deep Features for Linear Blend Skinning

$$V'_i = \sum_{k=1}^K W_{ik} D_k V_i$$

$V'_i / V_i$  Deformed vertex / starting vertex

$D_k$  Deformation for control structure k

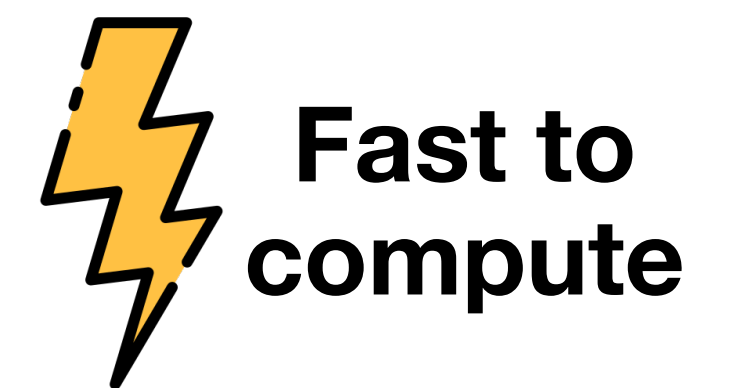
$W_{ik}$  Blending weight  
(defines deformation subspace)

## Deep Feature Deformation Weights

Every surface point / vertex can be a control handle

$$W_{ik} = 1 - ||\Phi(V_i) - \Phi(H_k)||_2$$

$\Phi$  Feature field       $H_k$  Point handle k



## Deep Feature Deformation Weights

Every surface point / vertex can be a control handle

$$W_{ik} = 1 - ||\Phi(V_i) - \Phi(H_k)||_2$$

$\Phi$  Feature field       $H_k$  Point handle k



## Deep Feature Deformation Weights

Every surface point / vertex can be a control handle

$$W_{ik} = 1 - ||\Phi(V_i) - \Phi(H_k)||_2$$

$\Phi$  Feature field       $H_k$  Point handle k

For unit-norm features,

$$-1 \leq W_{ik} \leq 1$$

Negative weights unintuitive  
and thus clamp to 0.

## Deep Feature Deformation Weights

Every surface point / vertex can be a control handle

$$W_{ik} = 1 - ||\Phi(V_i) - \Phi(H_k)||_2$$

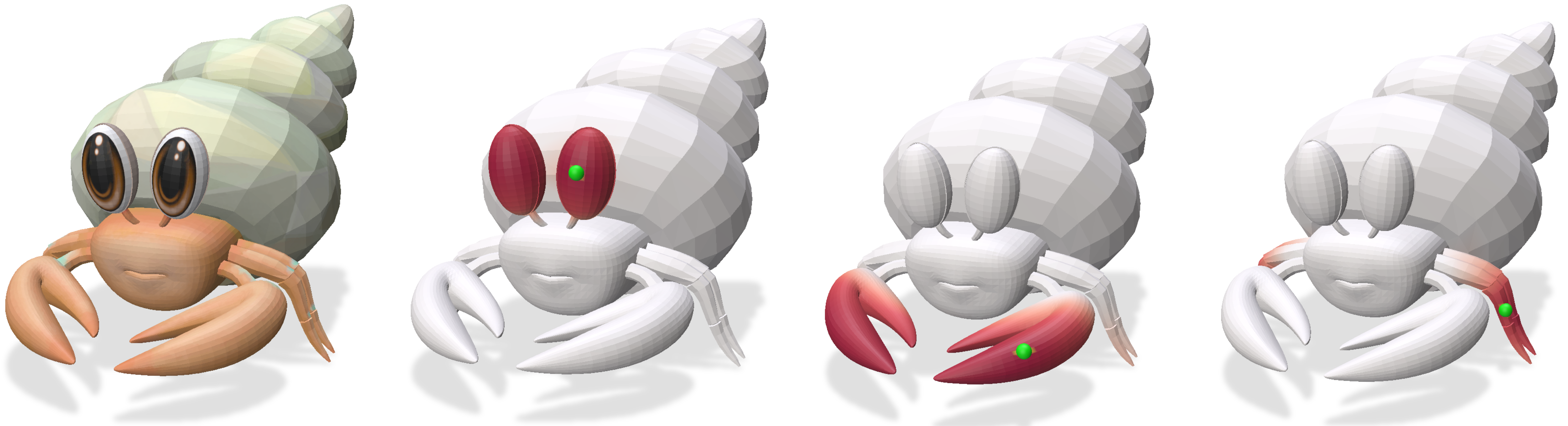
$\Phi$  Feature field       $H_k$  Point handle k

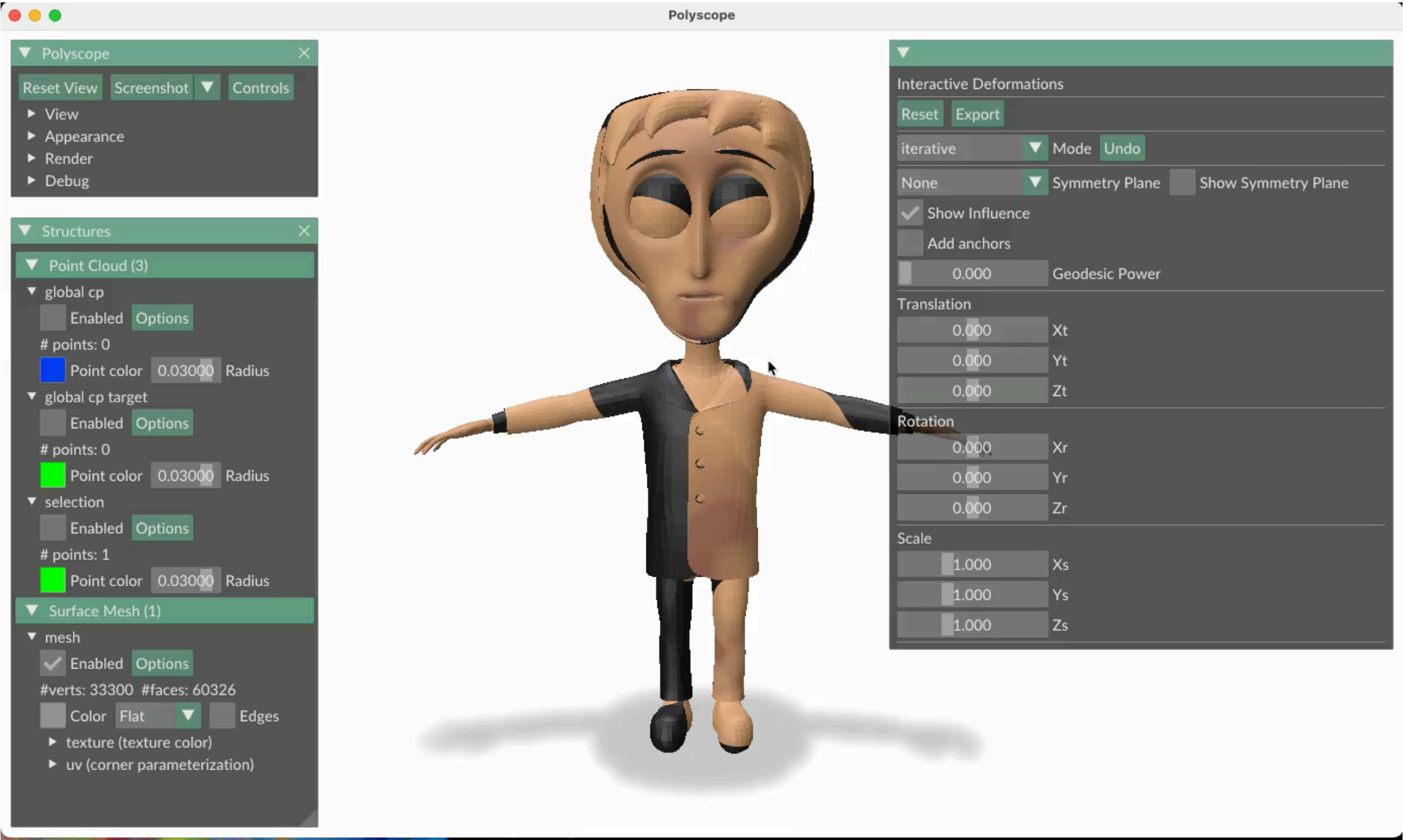
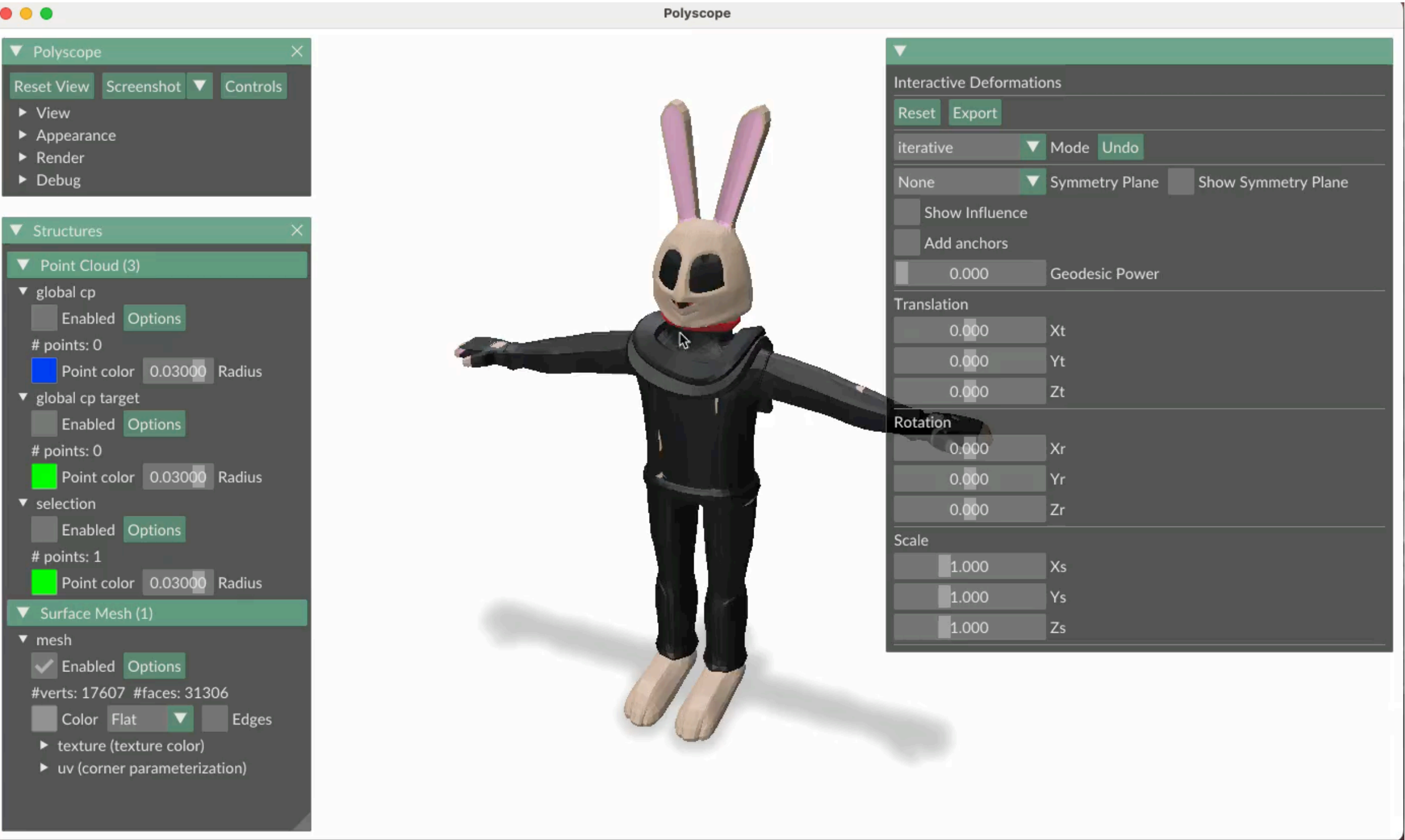
For unit-norm features,

$$-1 \leq W_{ik} \leq 1$$

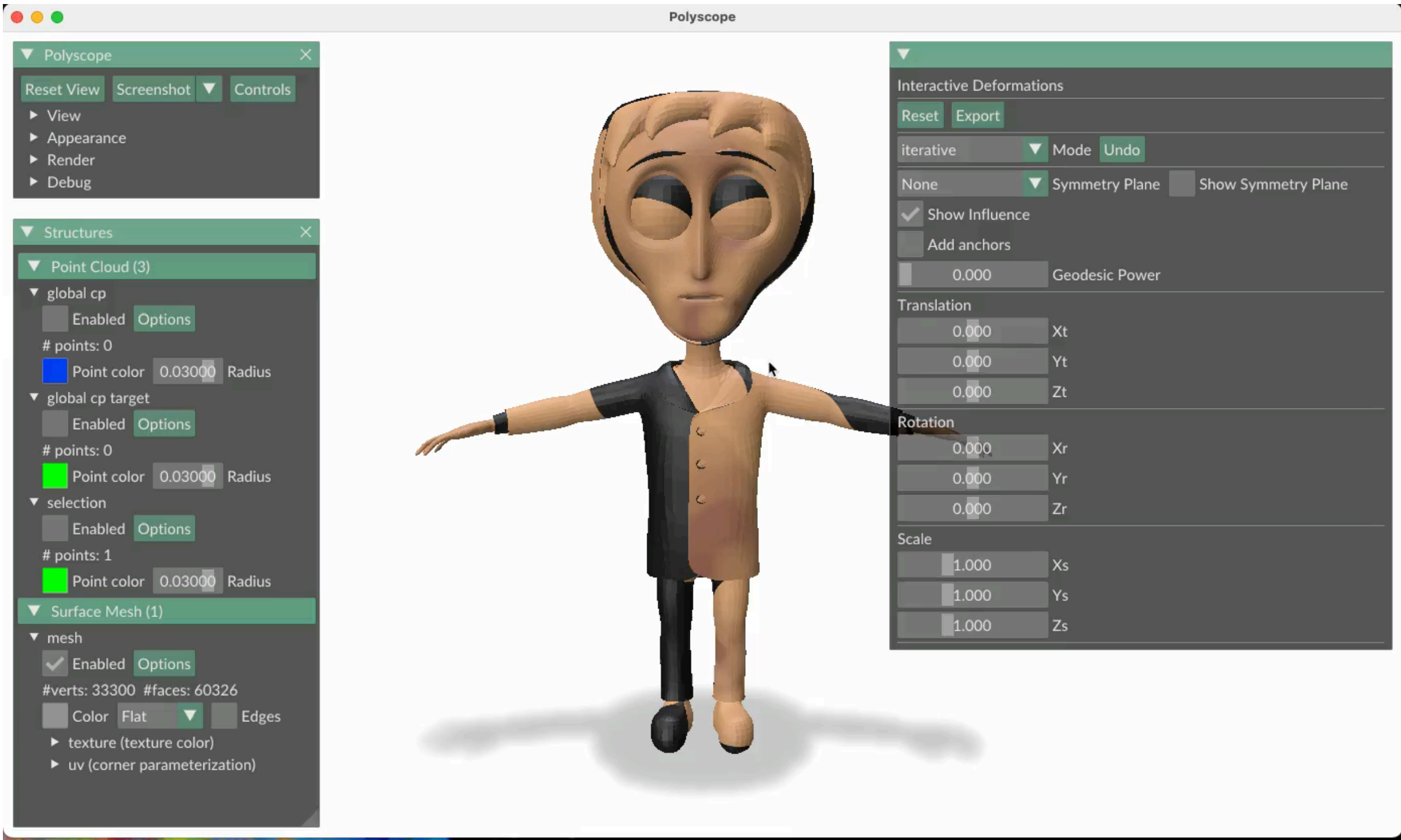
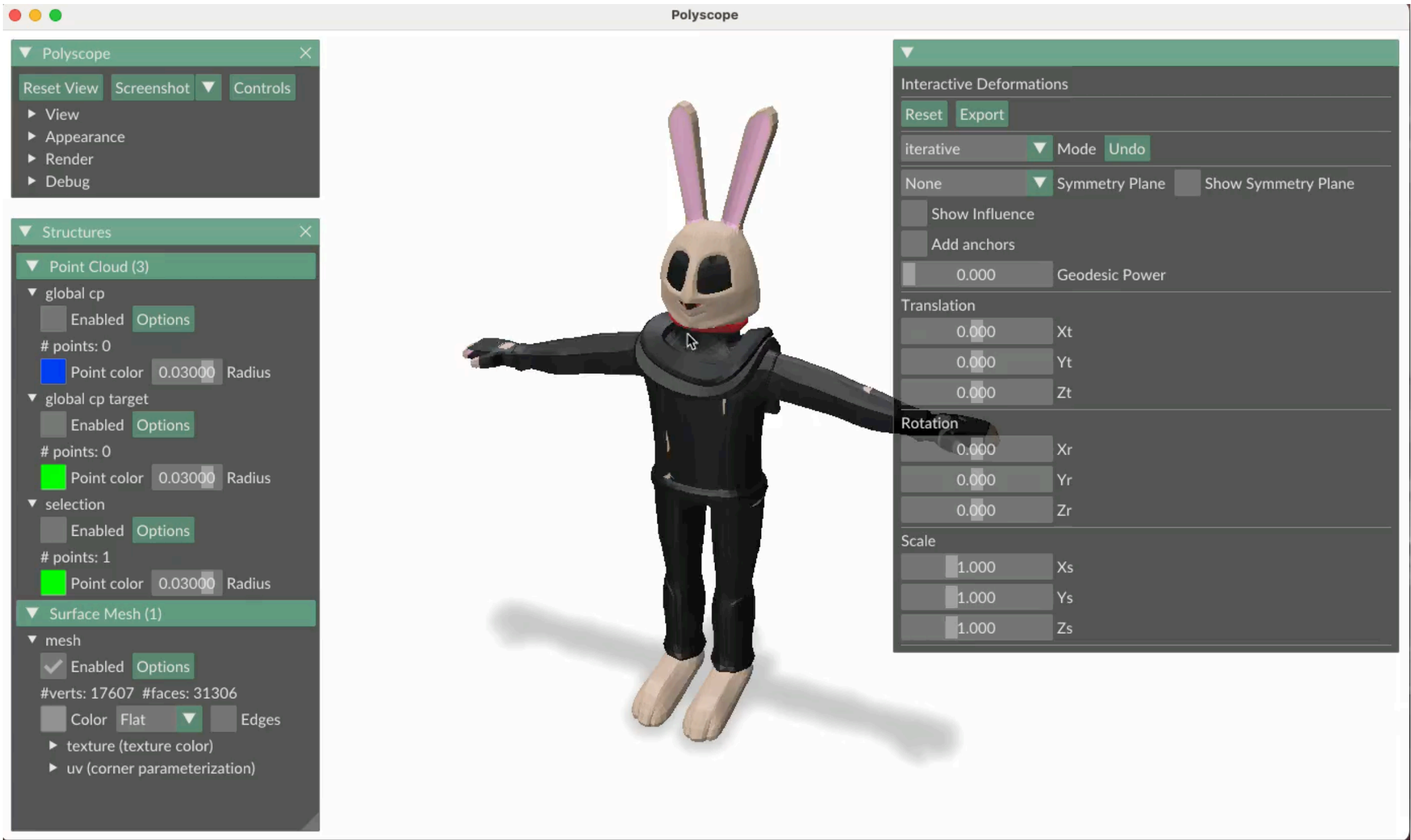
Negative weights unintuitive  
and thus clamp to 0.

“Points with similar visual features should co-deform”









# Tradeoffs

# Tradeoffs

No partition of unity/linear precision



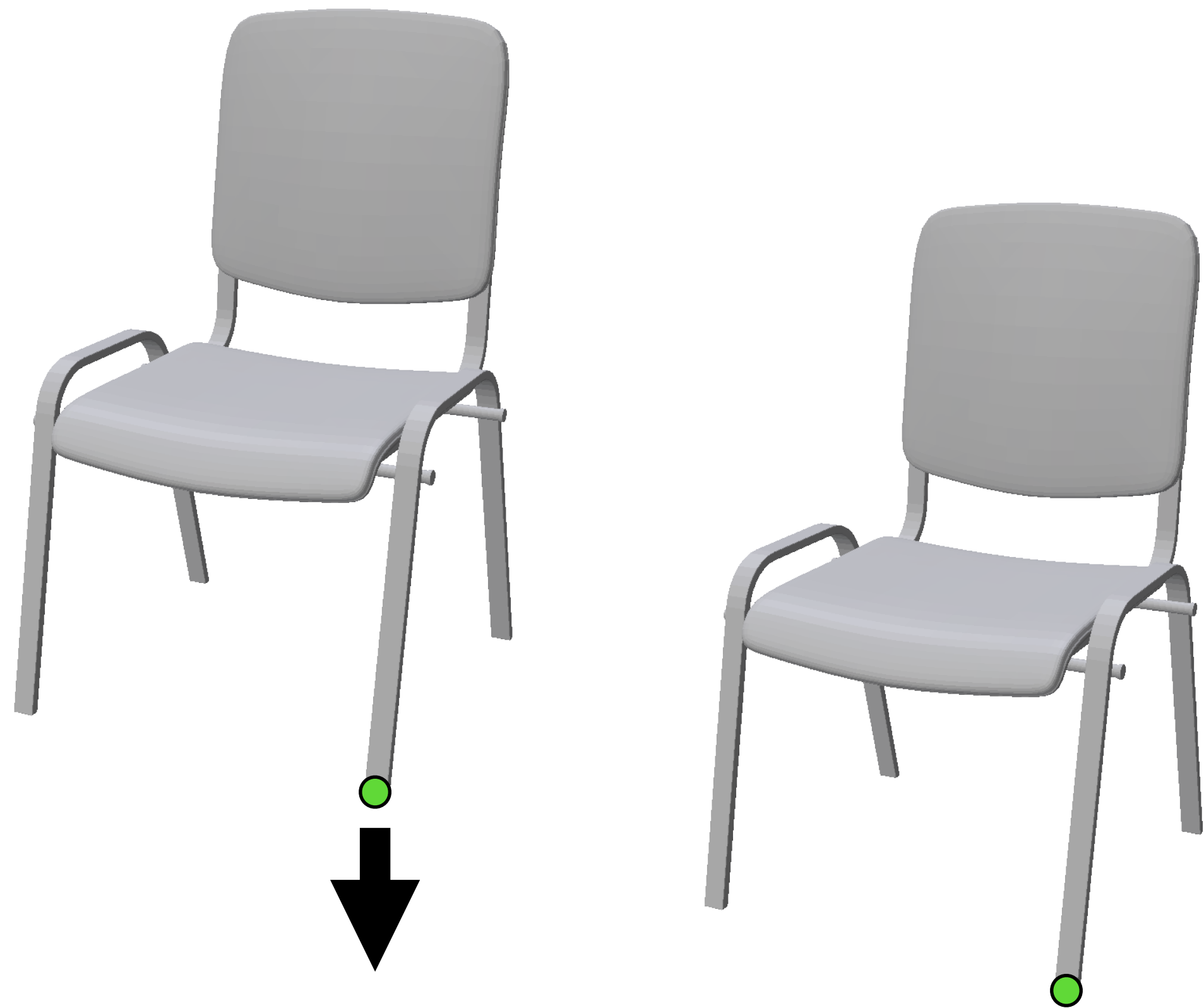
# Tradeoffs

No partition of unity/linear precision



# Tradeoffs

No partition of unity/linear precision



# Tradeoffs

No partition of unity/linear precision | Meaningful single-handle deformations

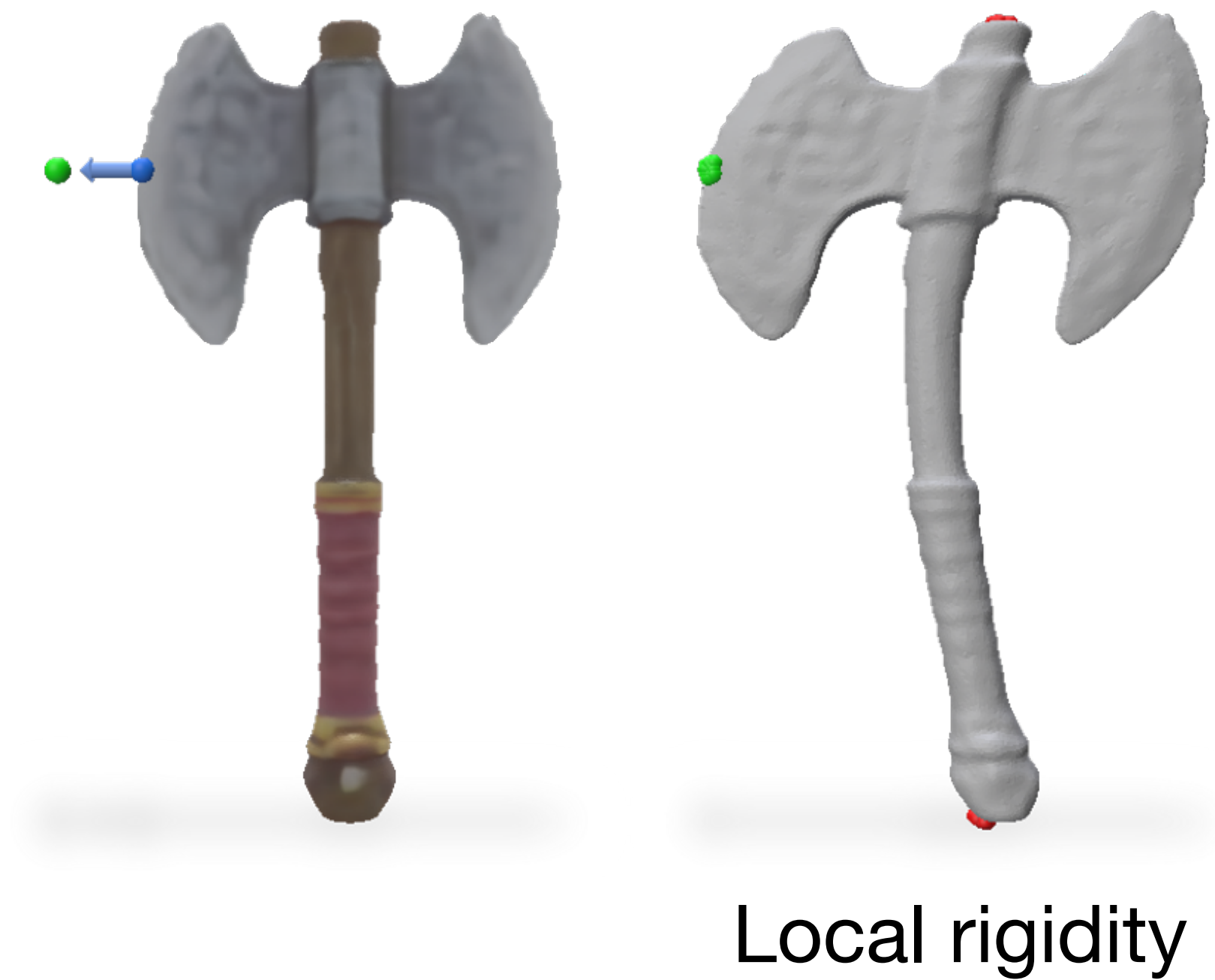




# Tradeoffs

No partition of unity/linear precision | Meaningful single-handle deformations

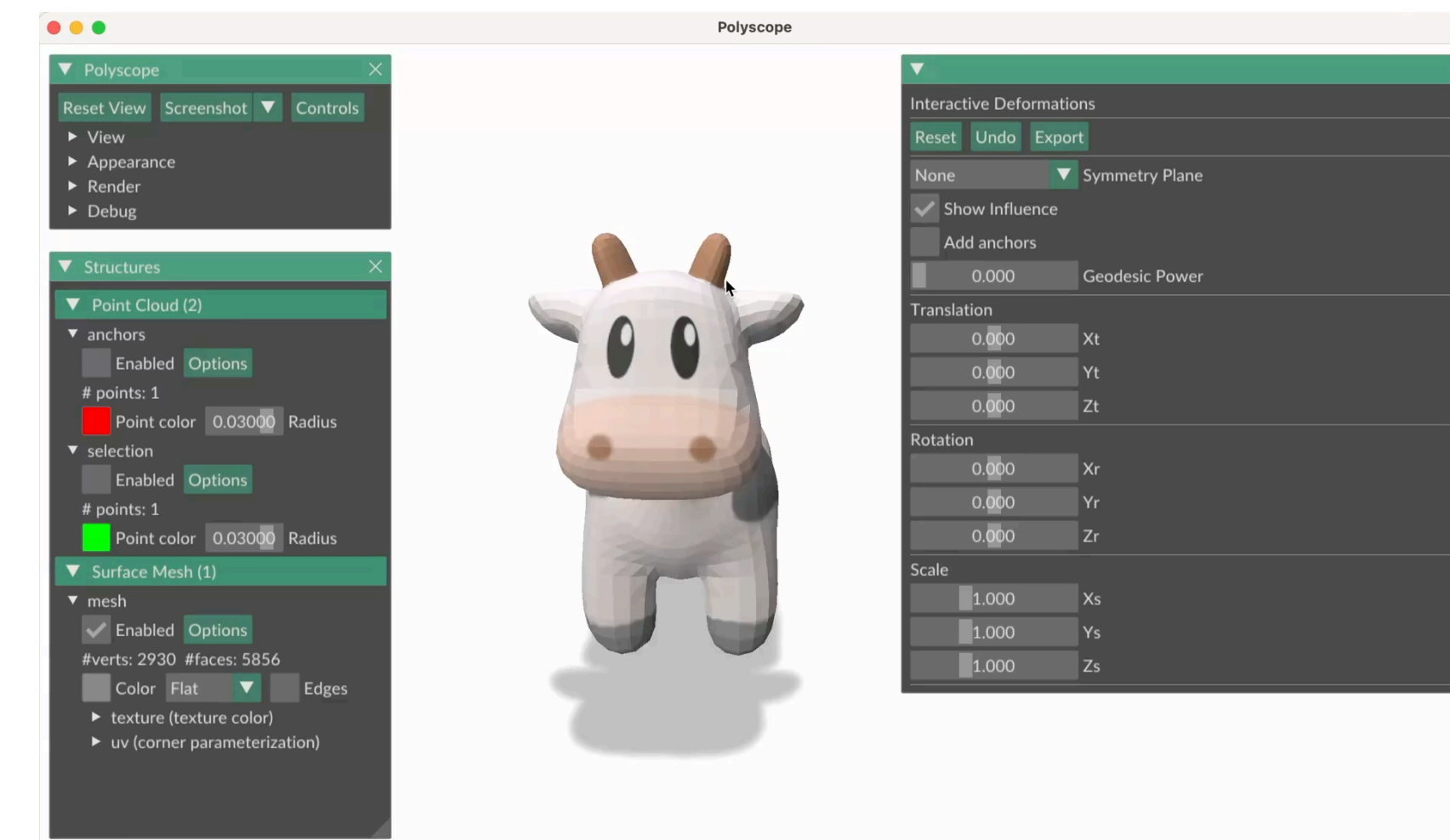
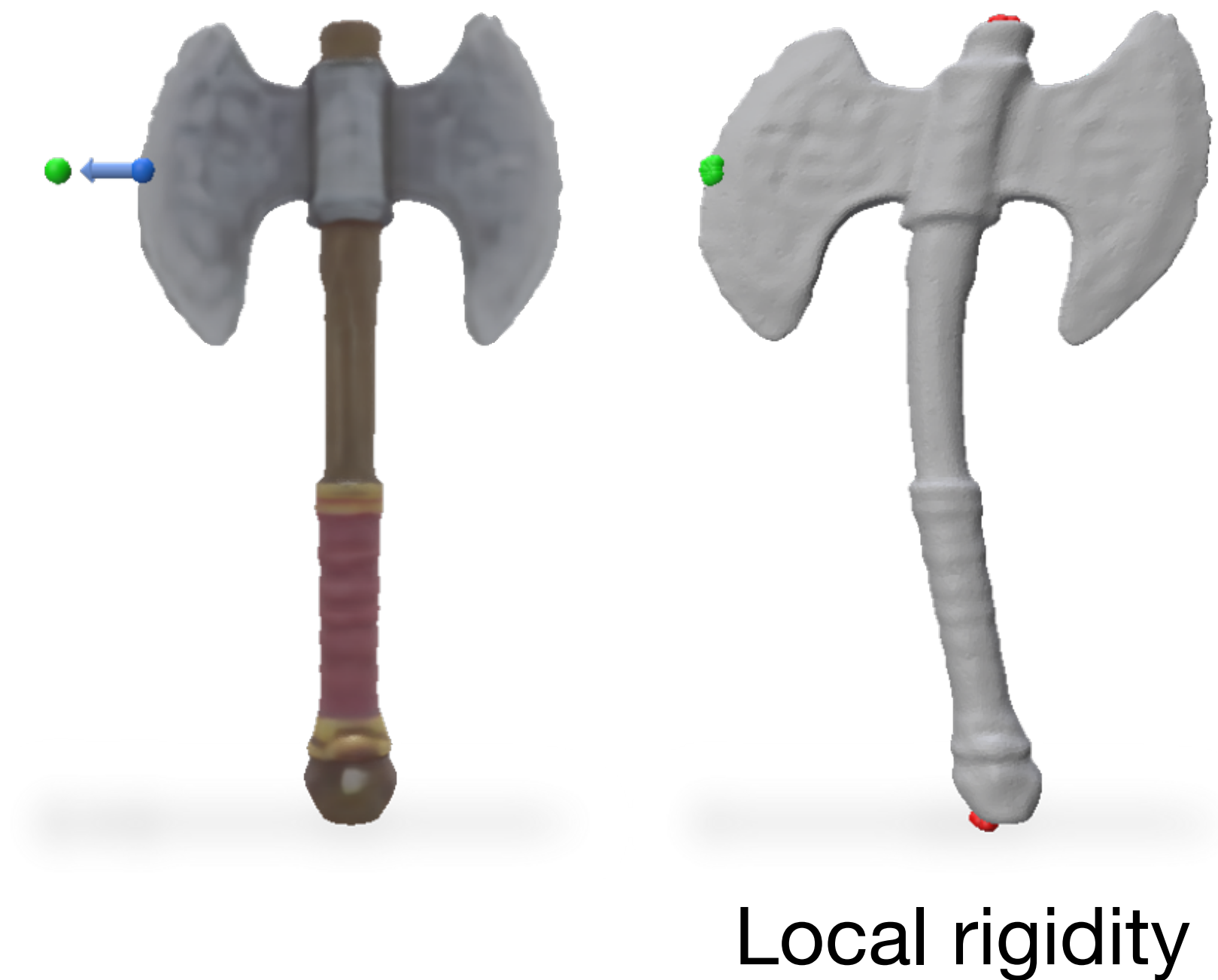
No geometry-based regularization



# Tradeoffs

No partition of unity/linear precision | Meaningful single-handle deformations

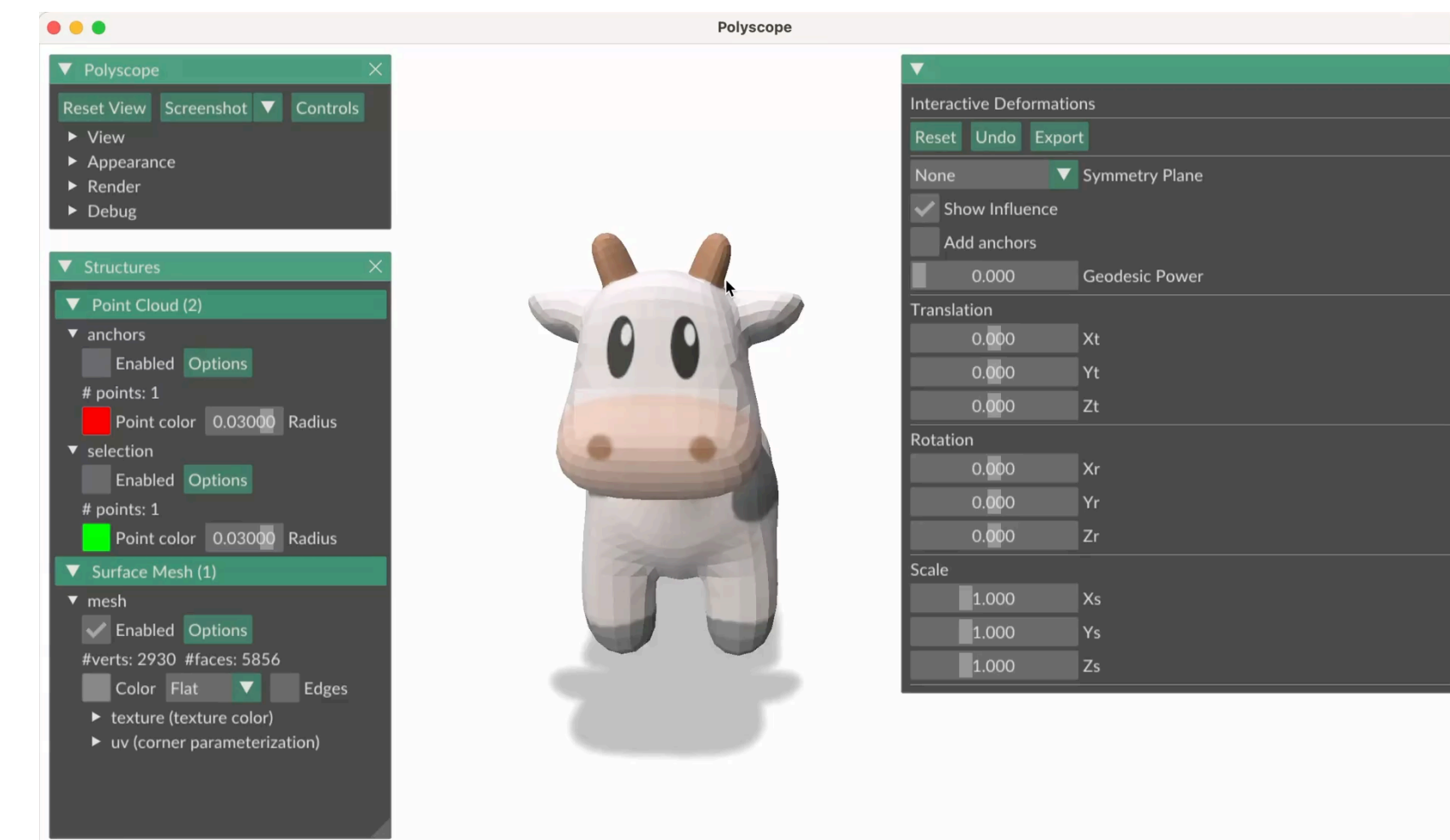
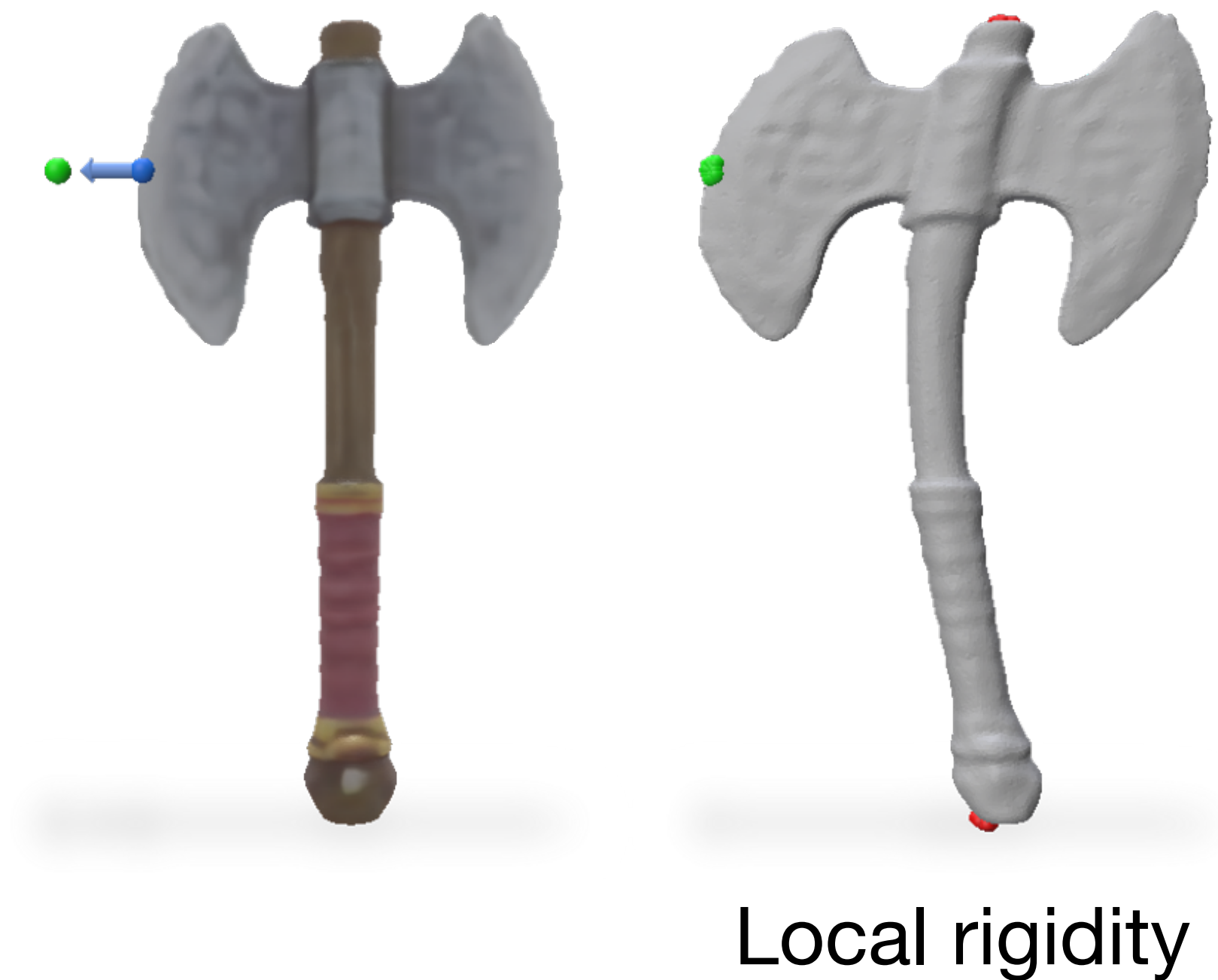
No geometry-based regularization | Real-time control point updates / sculpting



# Tradeoffs

No partition of unity/linear precision | Meaningful single-handle deformations

No geometry-based regularization | Real-time control point updates / sculpting





# Tradeoffs

No partition of unity/linear precision | Meaningful single-handle deformations

No geometry-based regularization | Real-time control point updates

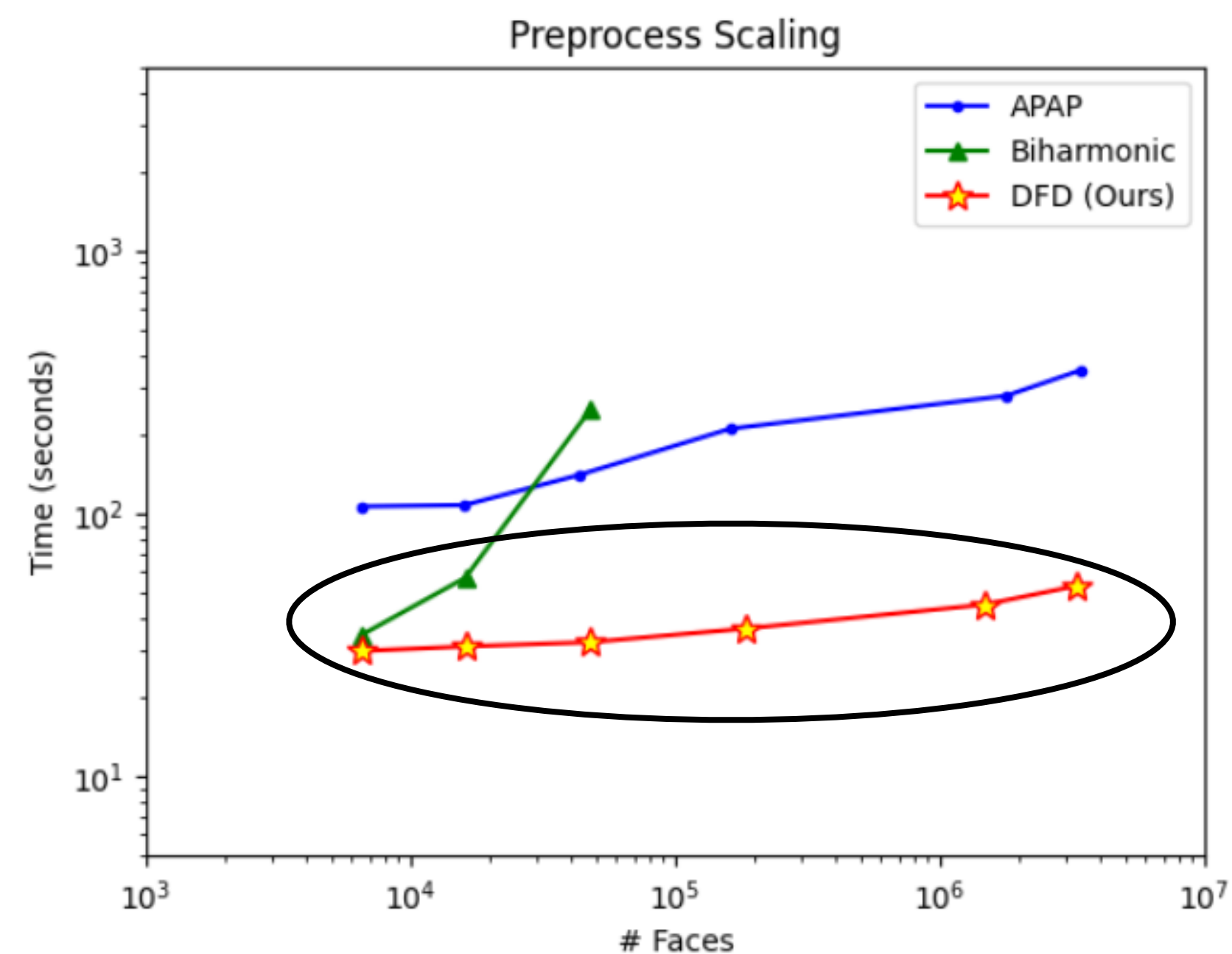
Need to optimize feature field

# Tradeoffs

No partition of unity/linear precision | Meaningful single-handle deformations

No geometry-based regularization | Real-time control point updates

Need to optimize feature field

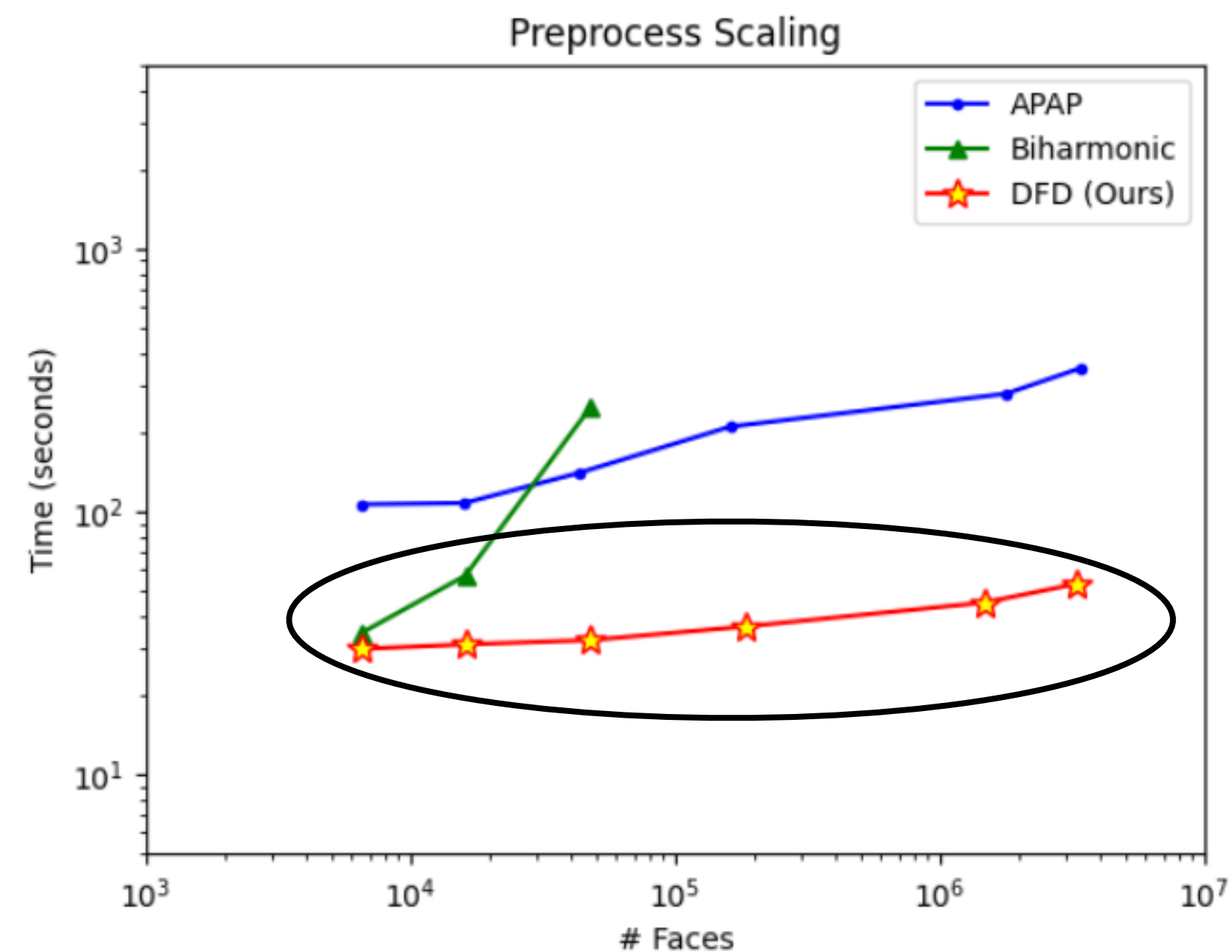


# Tradeoffs

No partition of unity/linear precision | Meaningful single-handle deformations

No geometry-based regularization | Real-time control point updates

Need to optimize feature field | Mesh resolution & representation agnostic



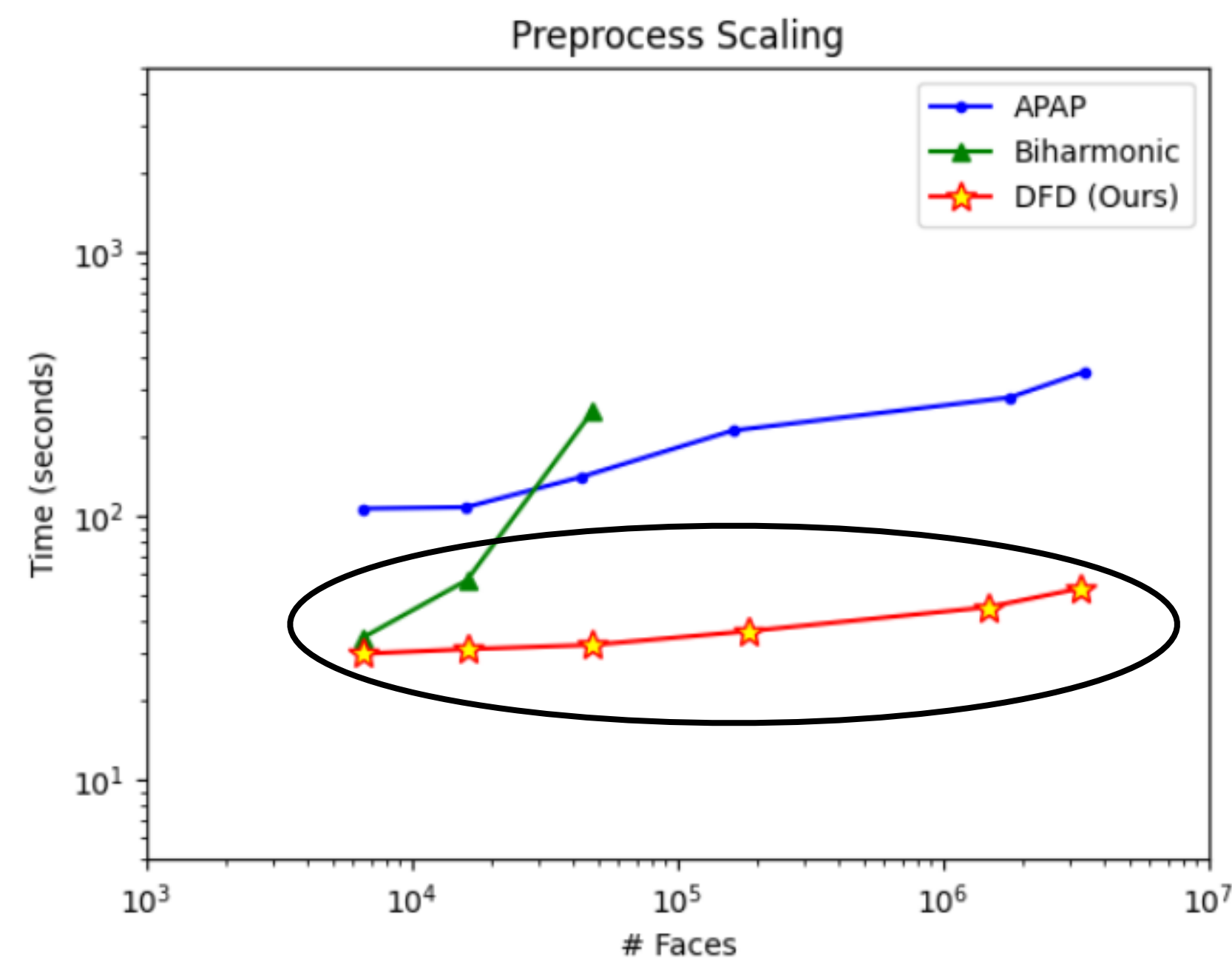


# Tradeoffs

No partition of unity/linear precision | Meaningful single-handle deformations

No geometry-based regularization | Real-time control point updates

Need to optimize feature field | Mesh resolution & representation agnostic



Mesh



GS



NeRF



PC

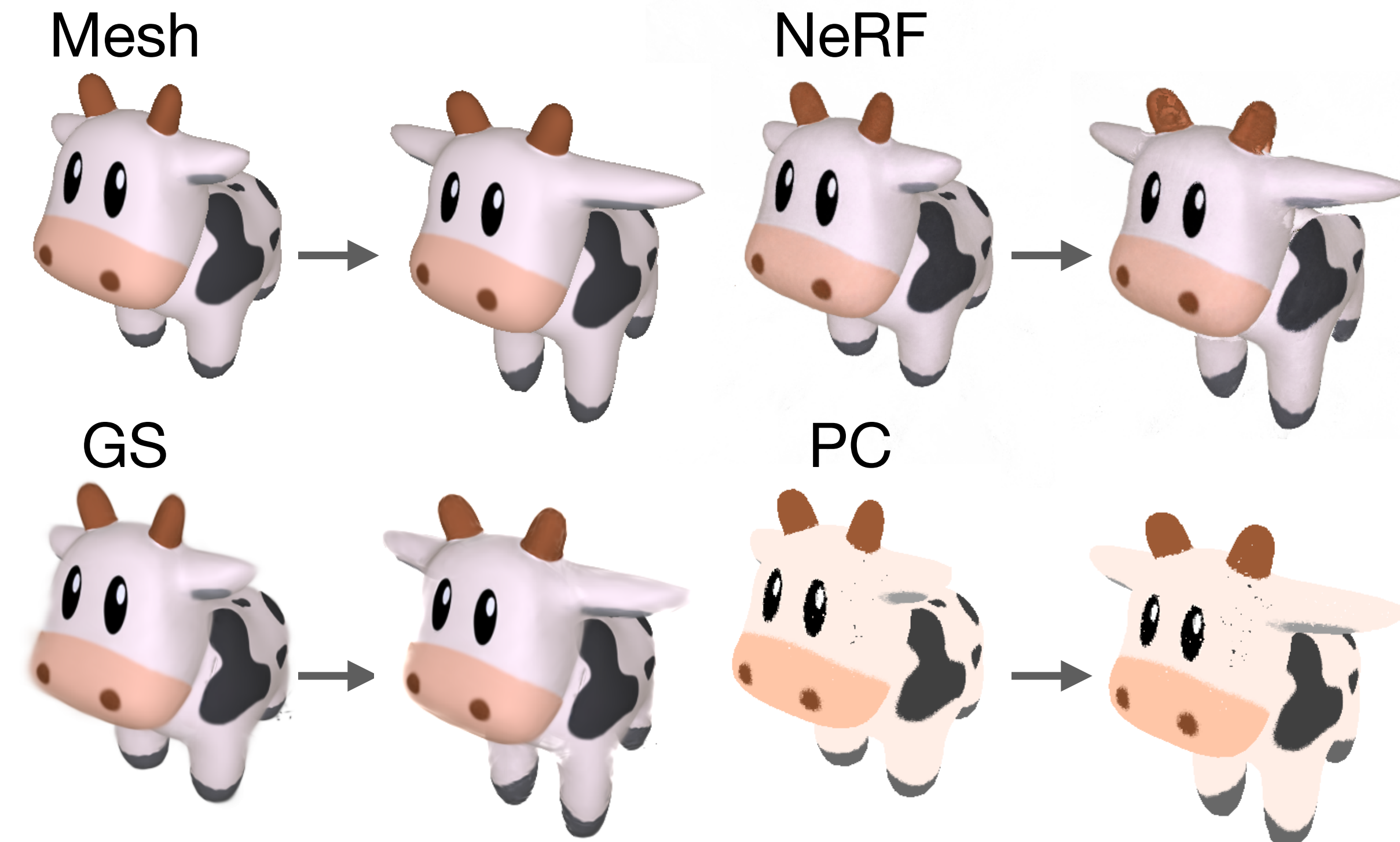
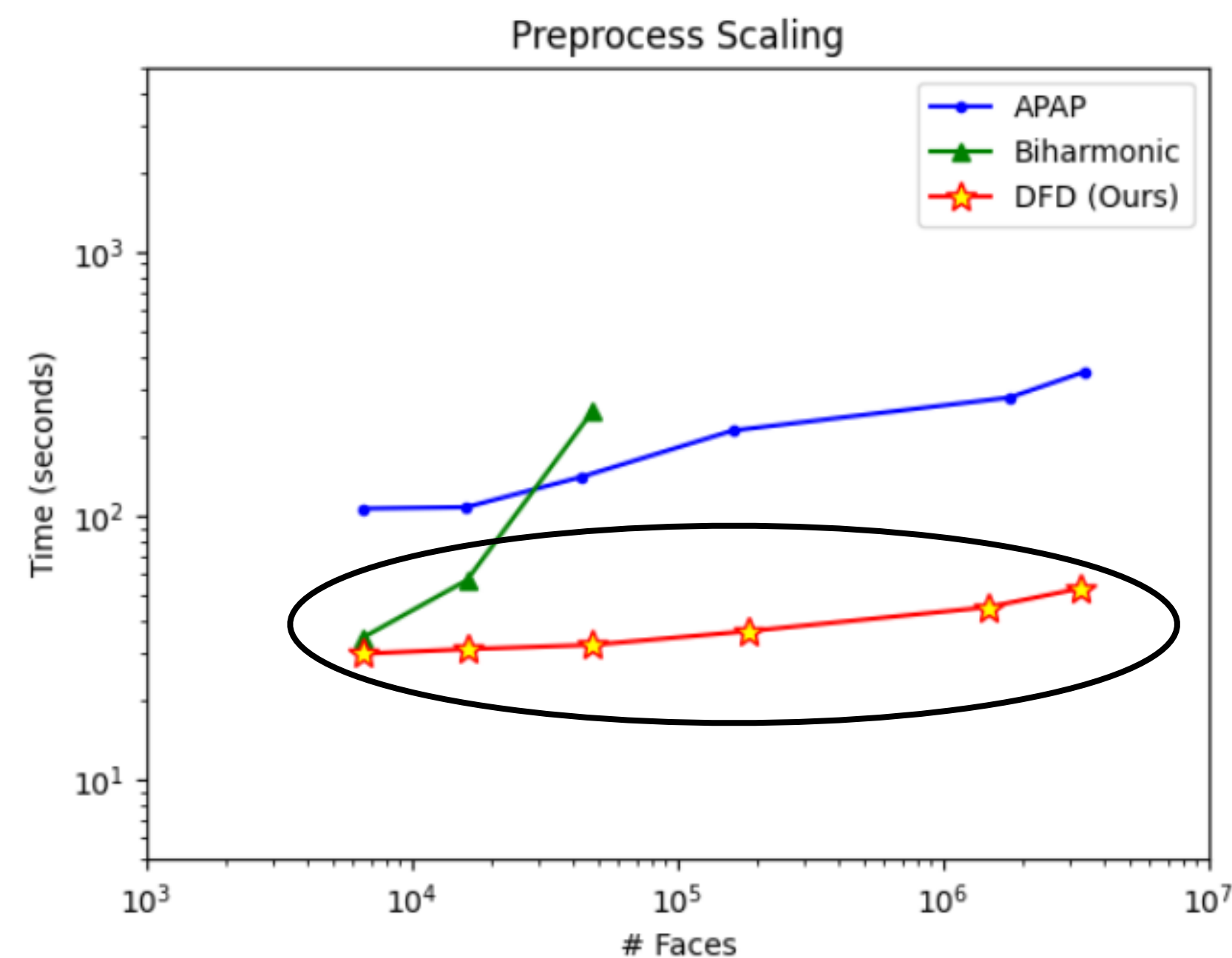


# Tradeoffs

No partition of unity/linear precision | Meaningful single-handle deformations

No geometry-based regularization | Real-time control point updates

Need to optimize feature field | Mesh resolution & representation agnostic



# Tradeoffs

No partition of unity/linear precision | Meaningful single-handle deformations

No geometry-based regularization | Real-time control point updates

Need to optimize neural field | Mesh resolution & representation agnostic

Weights determined by black box image features



# Tradeoffs

No partition of unity/linear precision | Meaningful single-handle deformations

No geometry-based regularization | Real-time control point updates

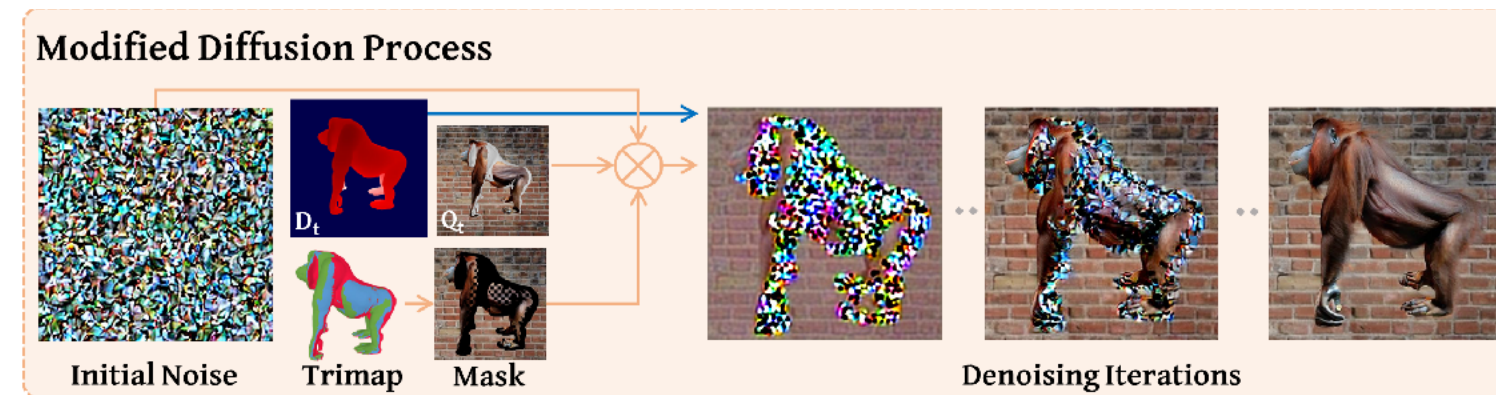
Need to optimize neural field | Mesh resolution & representation agnostic

Weights determined by black box image features

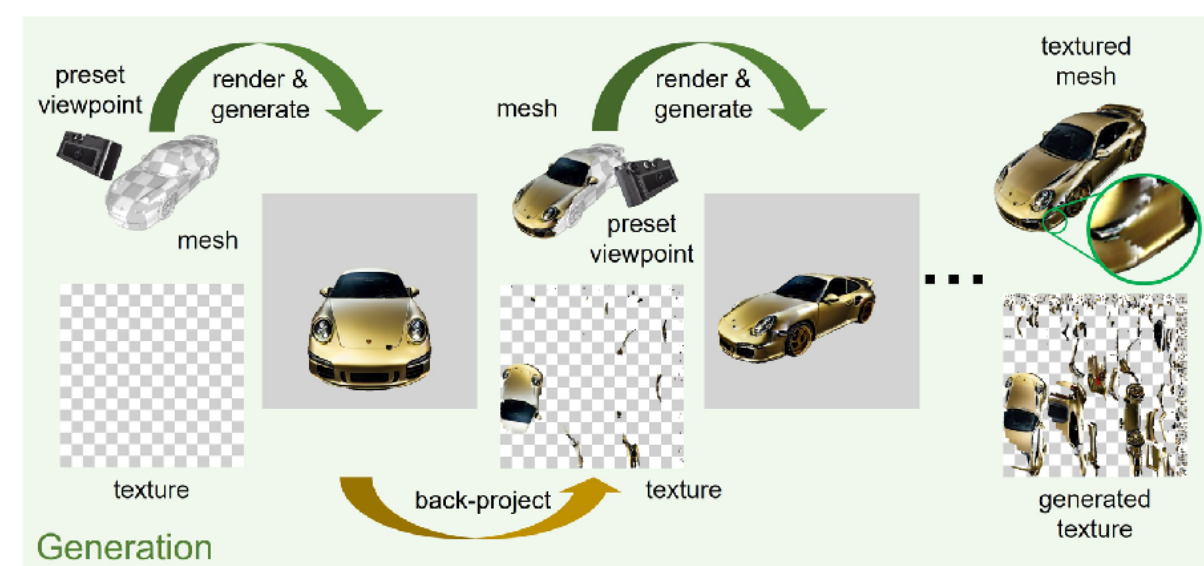


# Backprojection of...

## Pixels

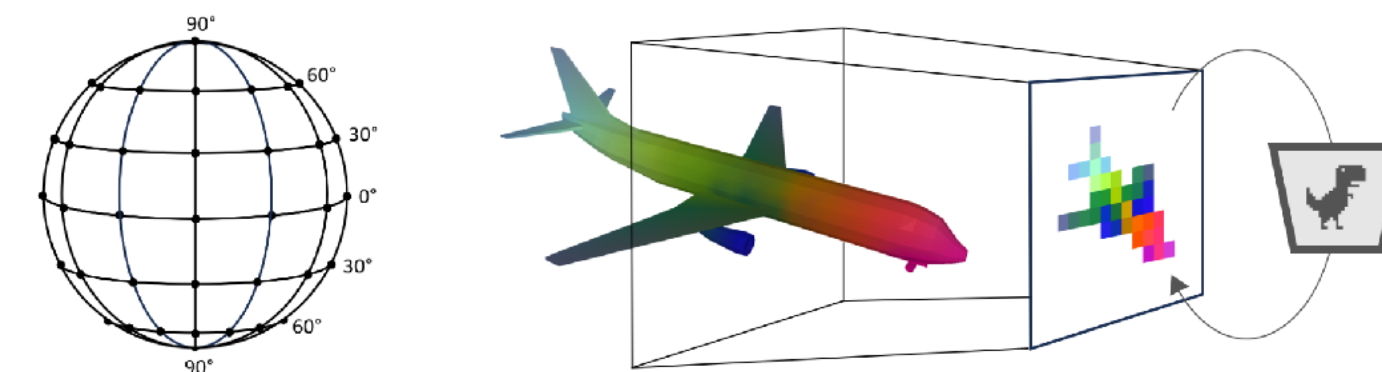


**TEXTure, Richardson et al. 2023**

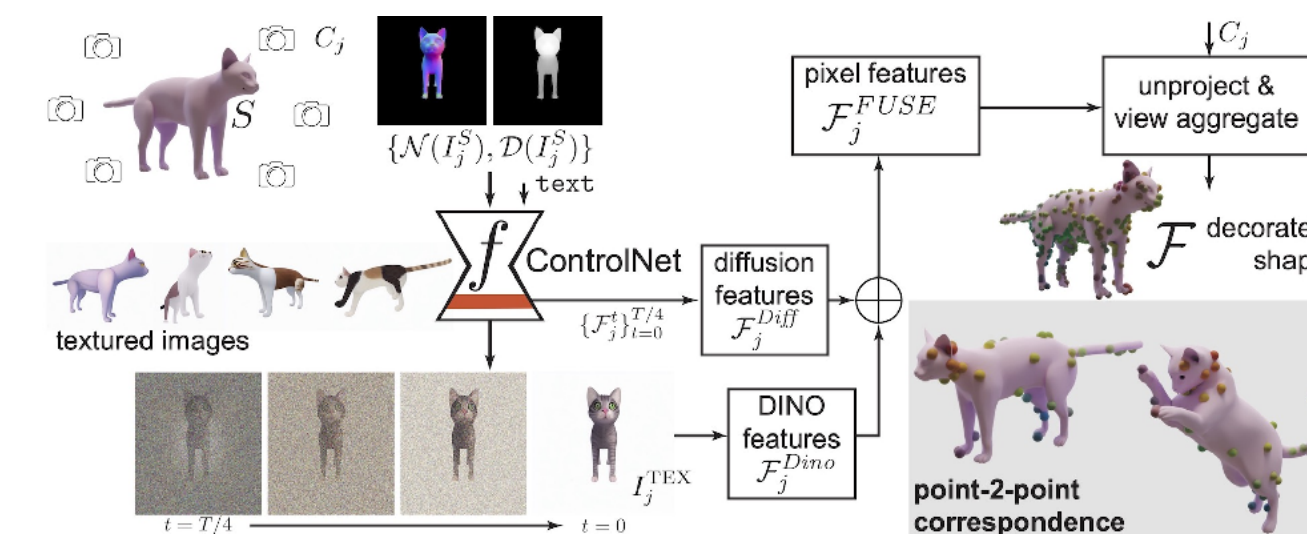


**Text2Tex, Chen et al. 2023**

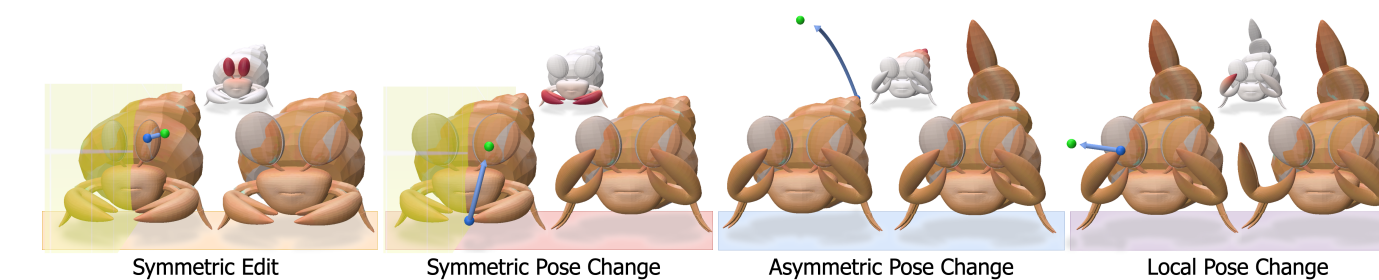
## Deep Features



**Back to 3D, Wimmer et al. 2023**



**Diff3f, Dutt et al. 2024**



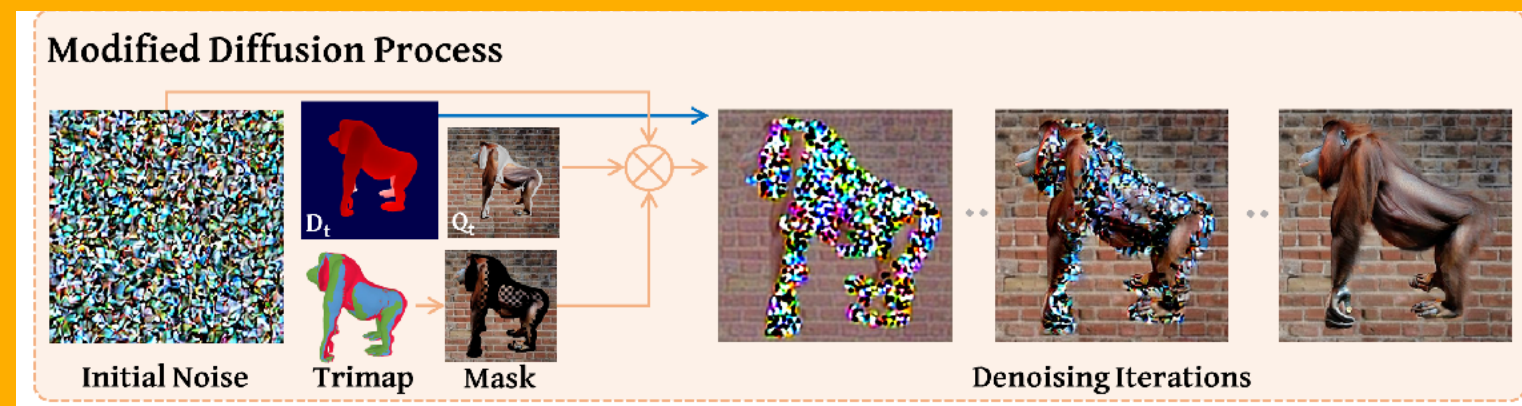
**DFD, Liu et al. 2026**



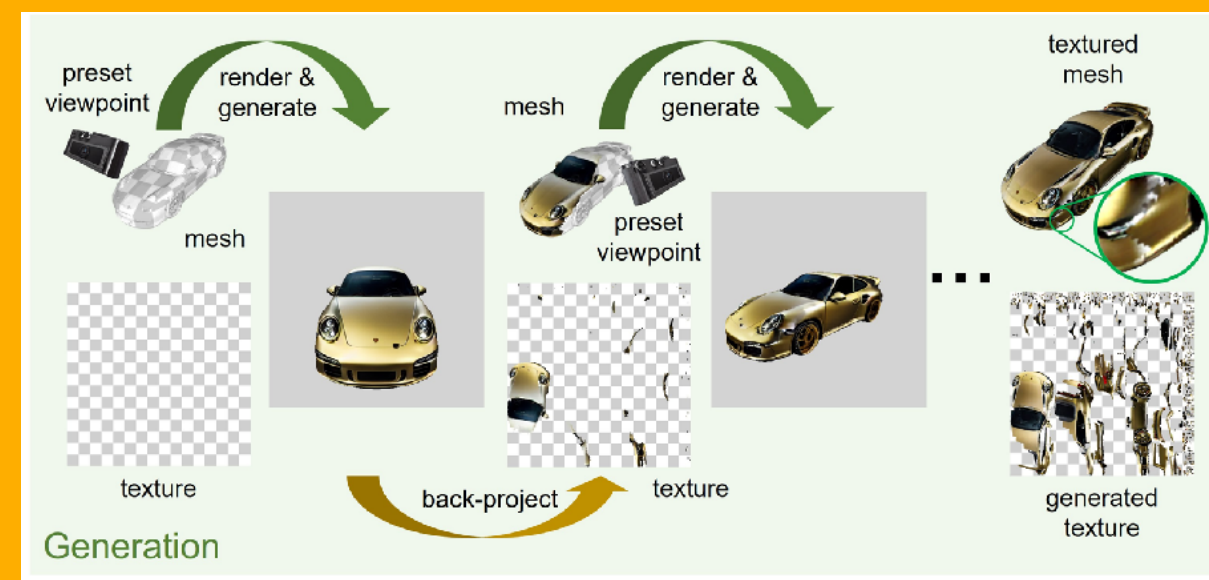
# Backprojection of...

## Projection Invariant

### Pixels

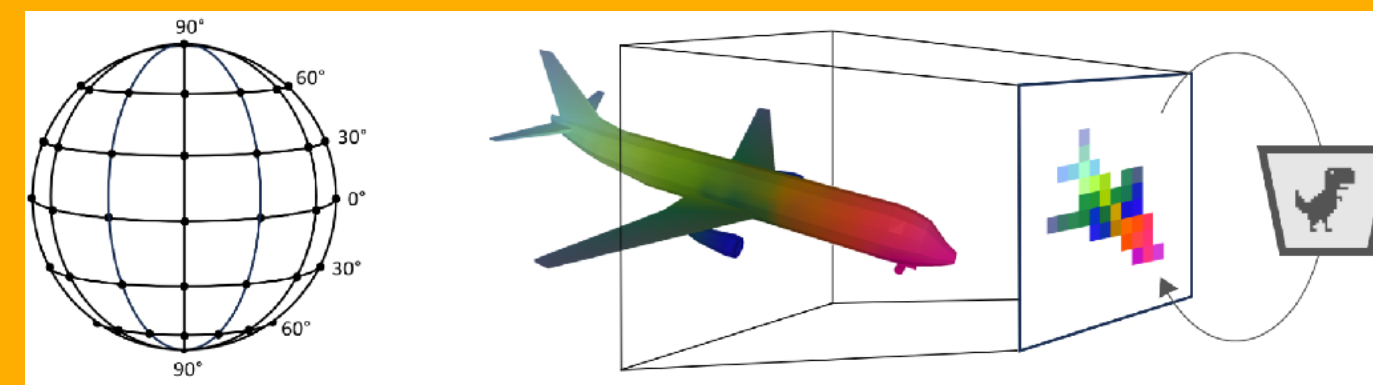


**TEXTure, Richardson et al. 2023**

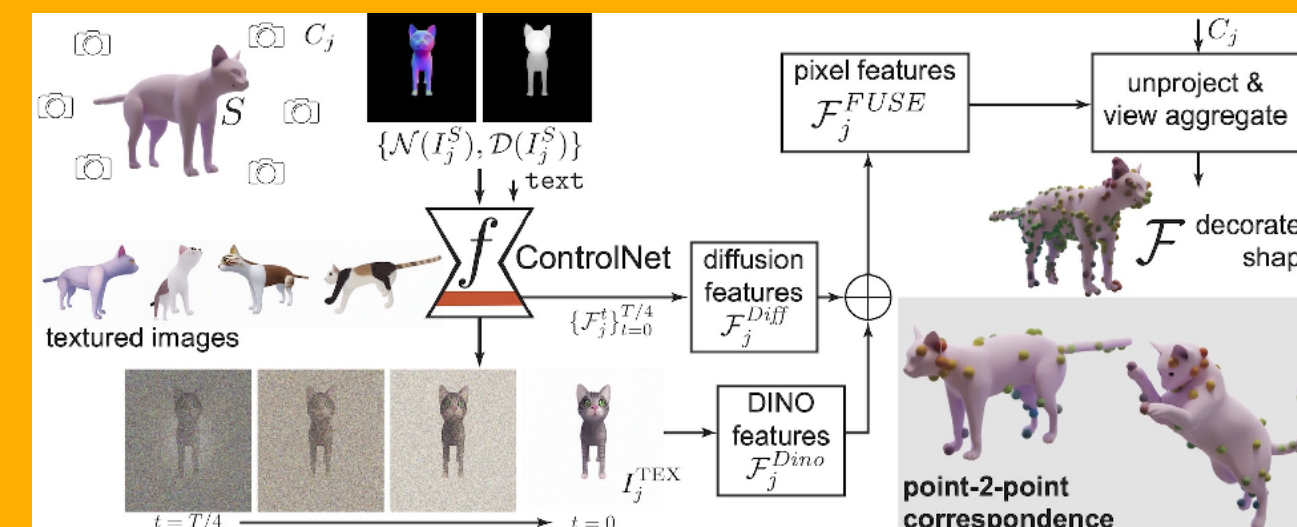


**Text2Tex, Chen et al. 2023**

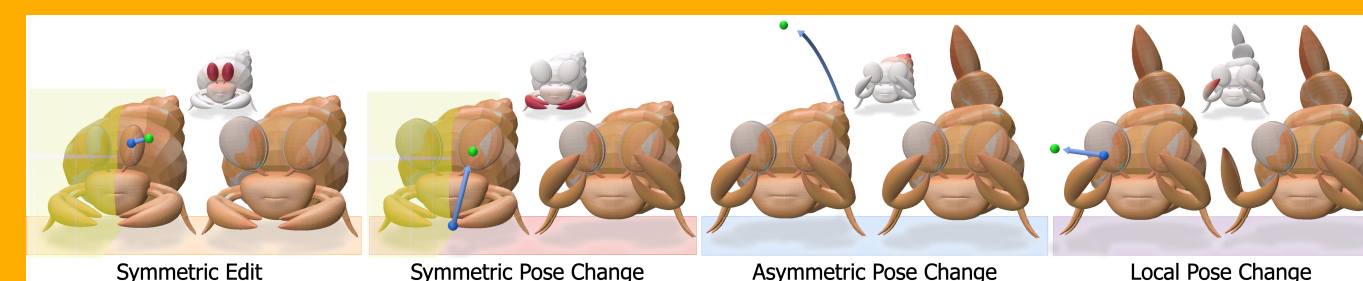
### Deep Features



**Back to 3D, Wimmer et al. 2023**



**Diff3f, Dutt et al. 2024**



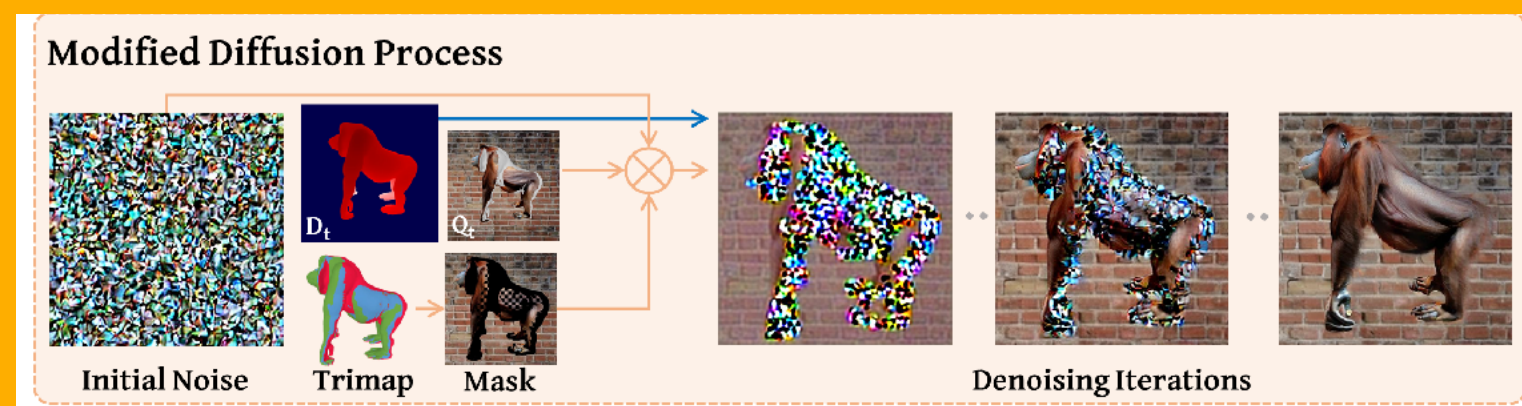
**DFD, Liu et al. 2026**



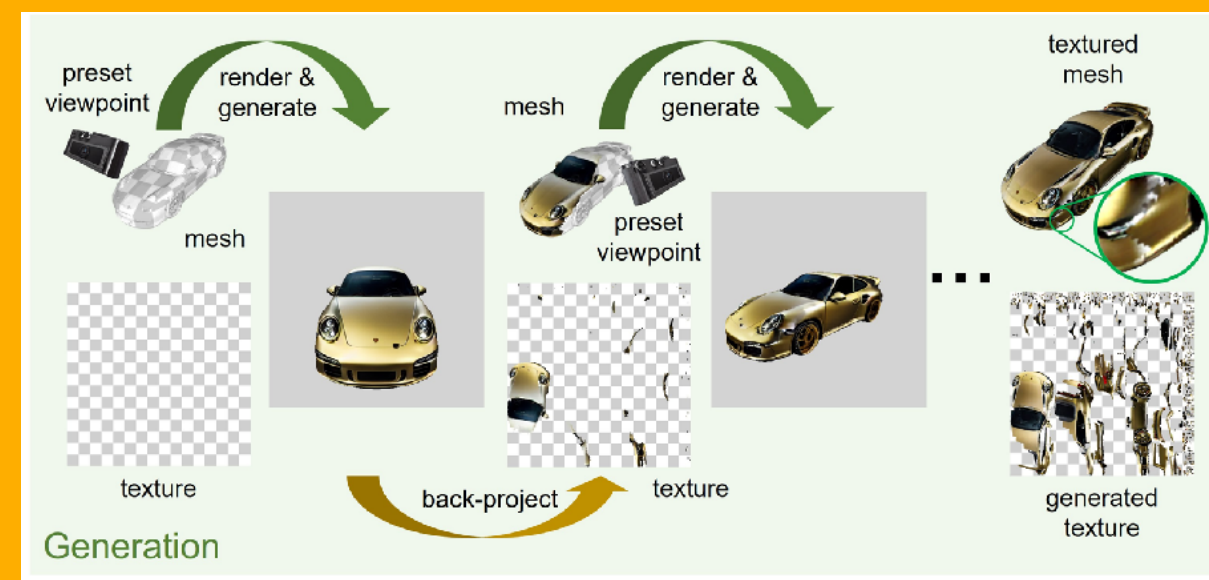
# Backprojection of...

## Projection Invariant

### Pixels

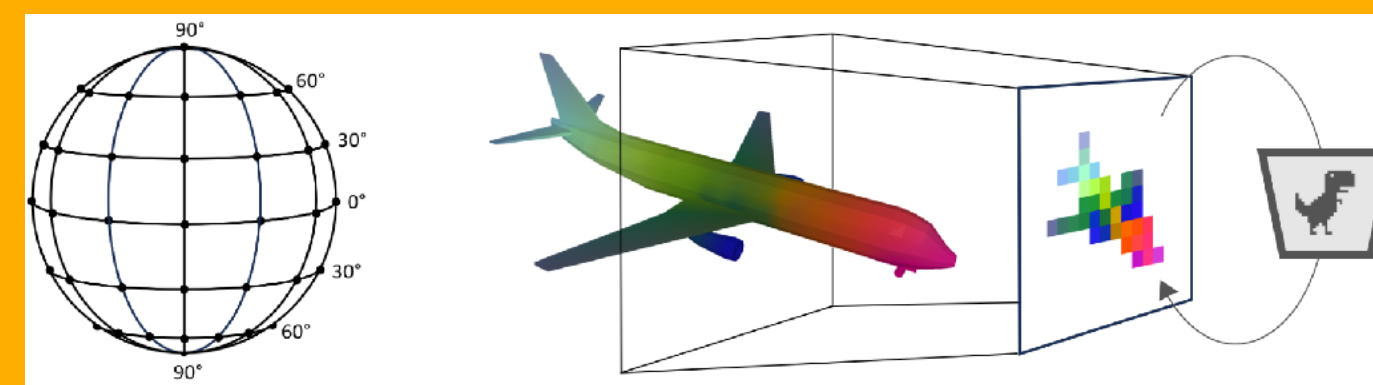


**TEXTure, Richardson et al. 2023**

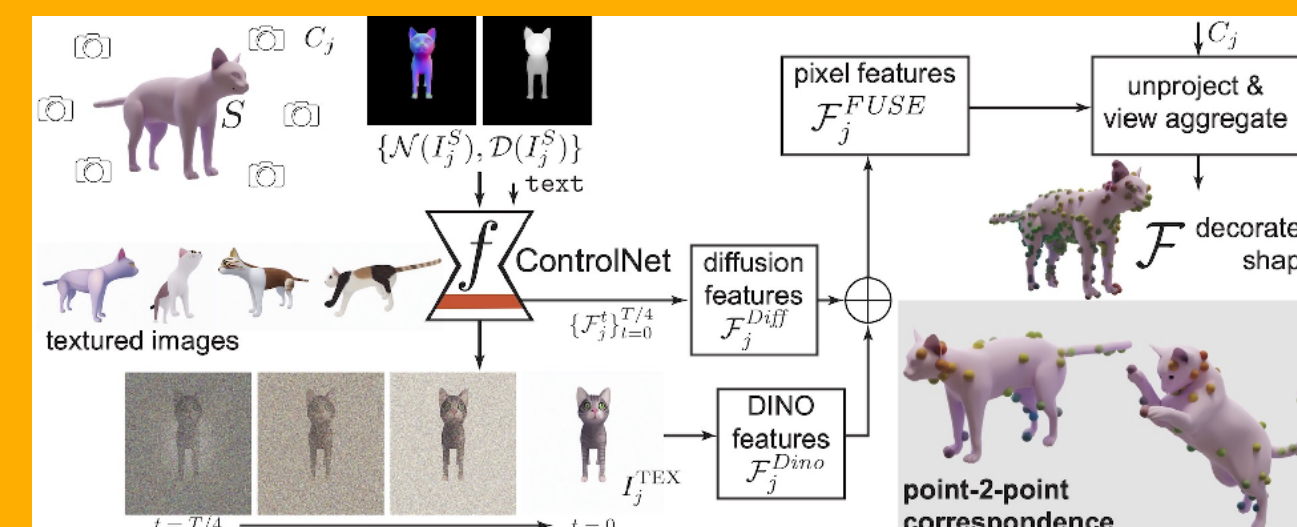


**Text2Tex, Chen et al. 2023**

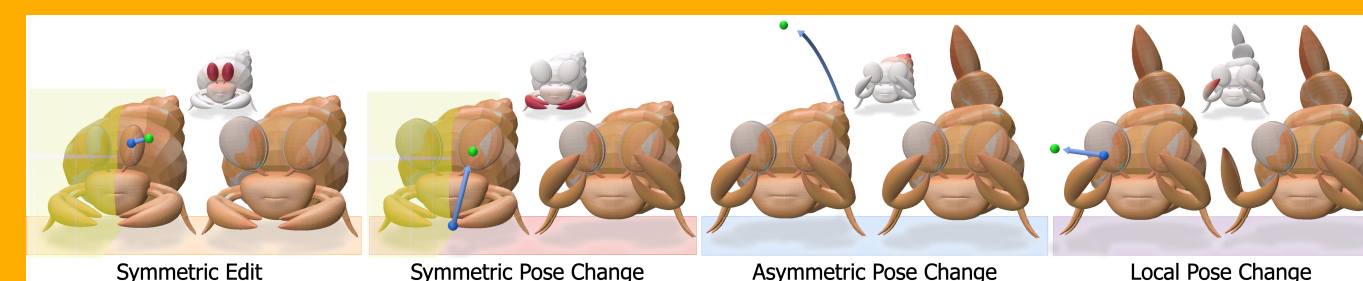
### Deep Features



**Back to 3D, Wimmer et al. 2023**



**Diff3f, Dutt et al. 2024**



**DFD, Liu et al. 2026**

## Projection Equivariant

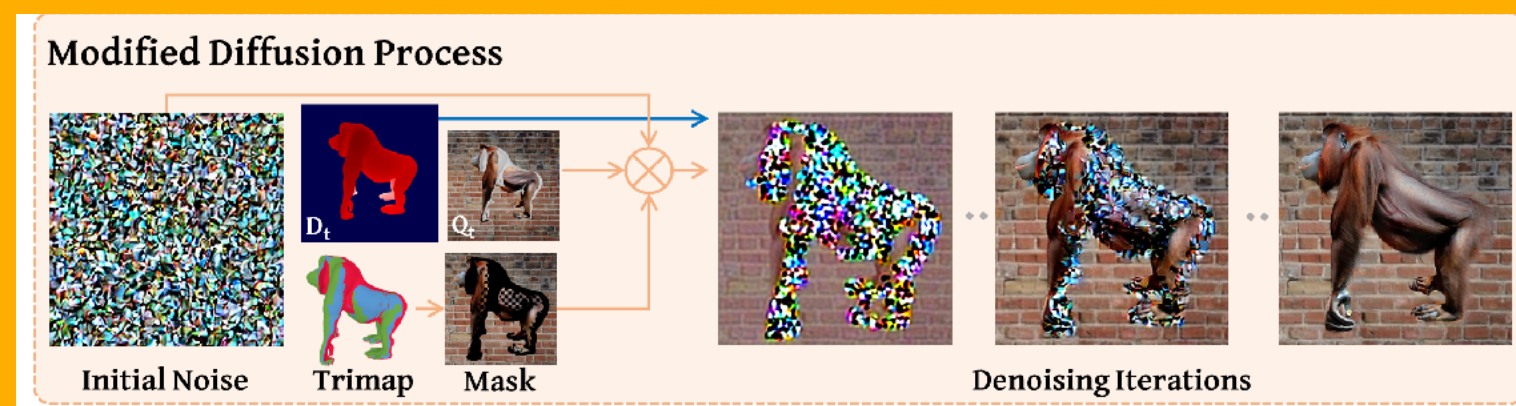
### Geometric Quantities



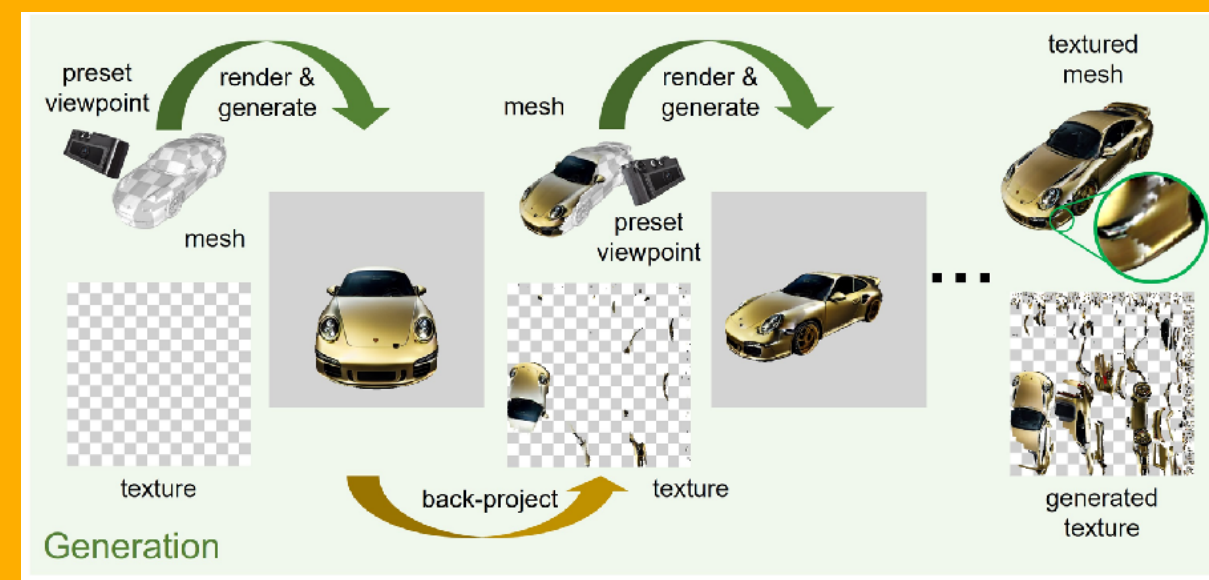
# Backprojection of...

## Projection Invariant

### Pixels

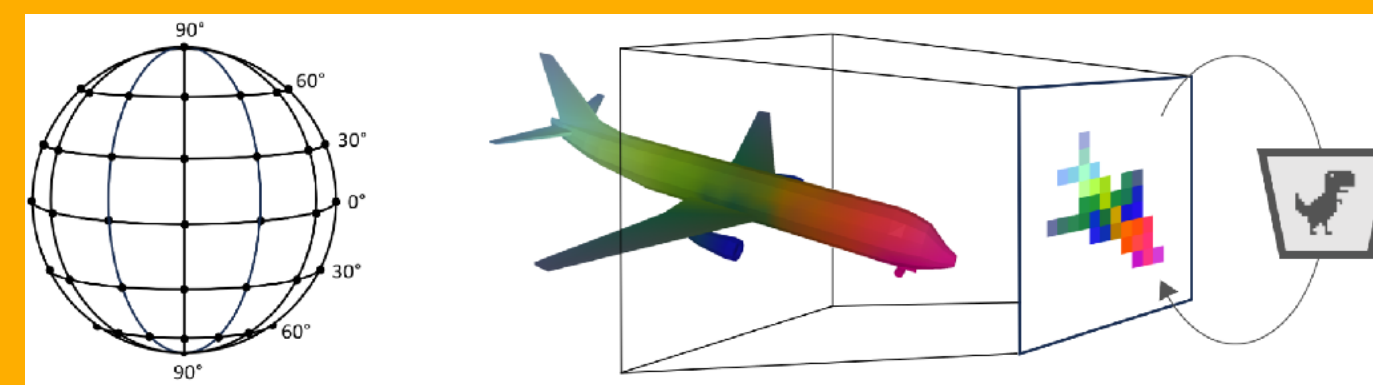


**TEXTure, Richardson et al. 2023**

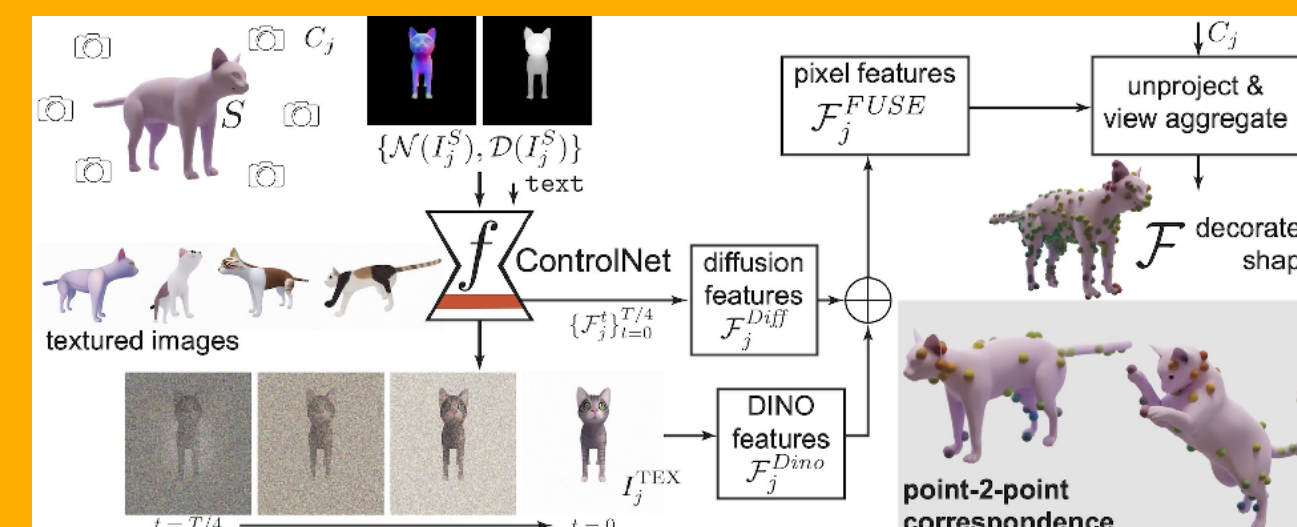


**Text2Tex, Chen et al. 2023**

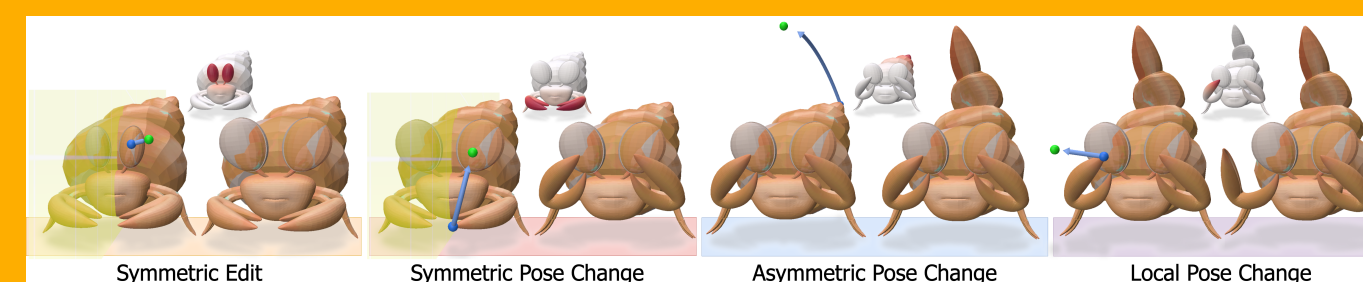
### Deep Features



**Back to 3D, Wimmer et al. 2023**



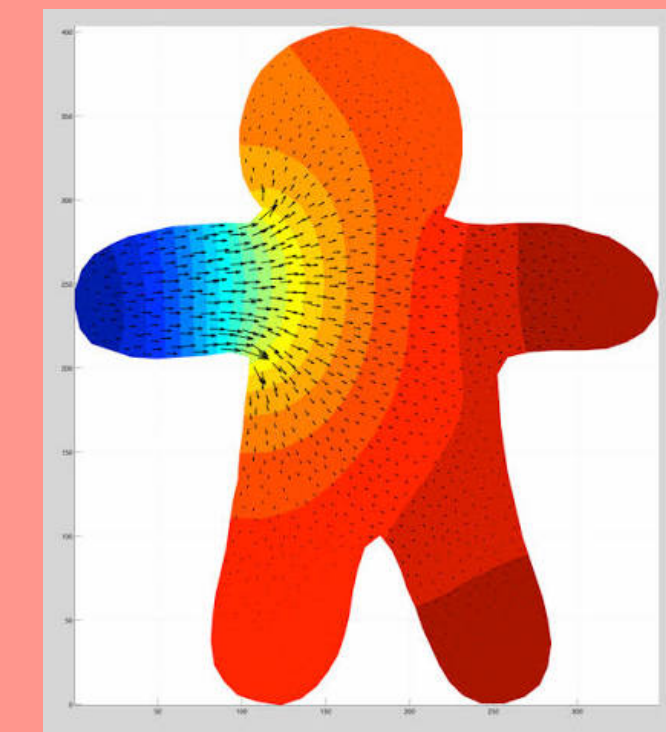
**Diff3f, Dutt et al. 2024**



**DFD, Liu et al. 2026**

## Projection Equivariant

### Geometric Quantities



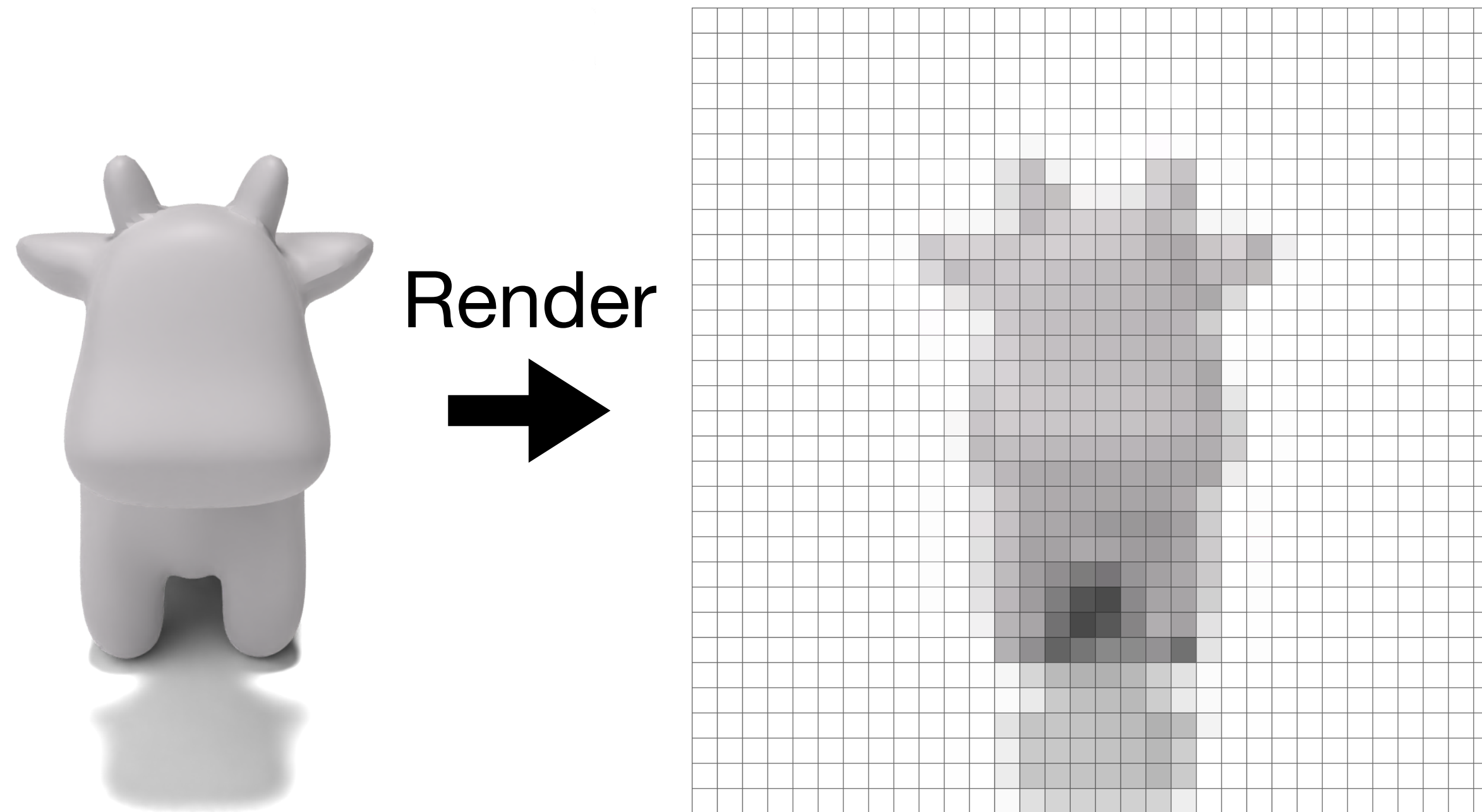
**Image space gradients<sup>1</sup>**

# Future Work: Compute & back project image gradients





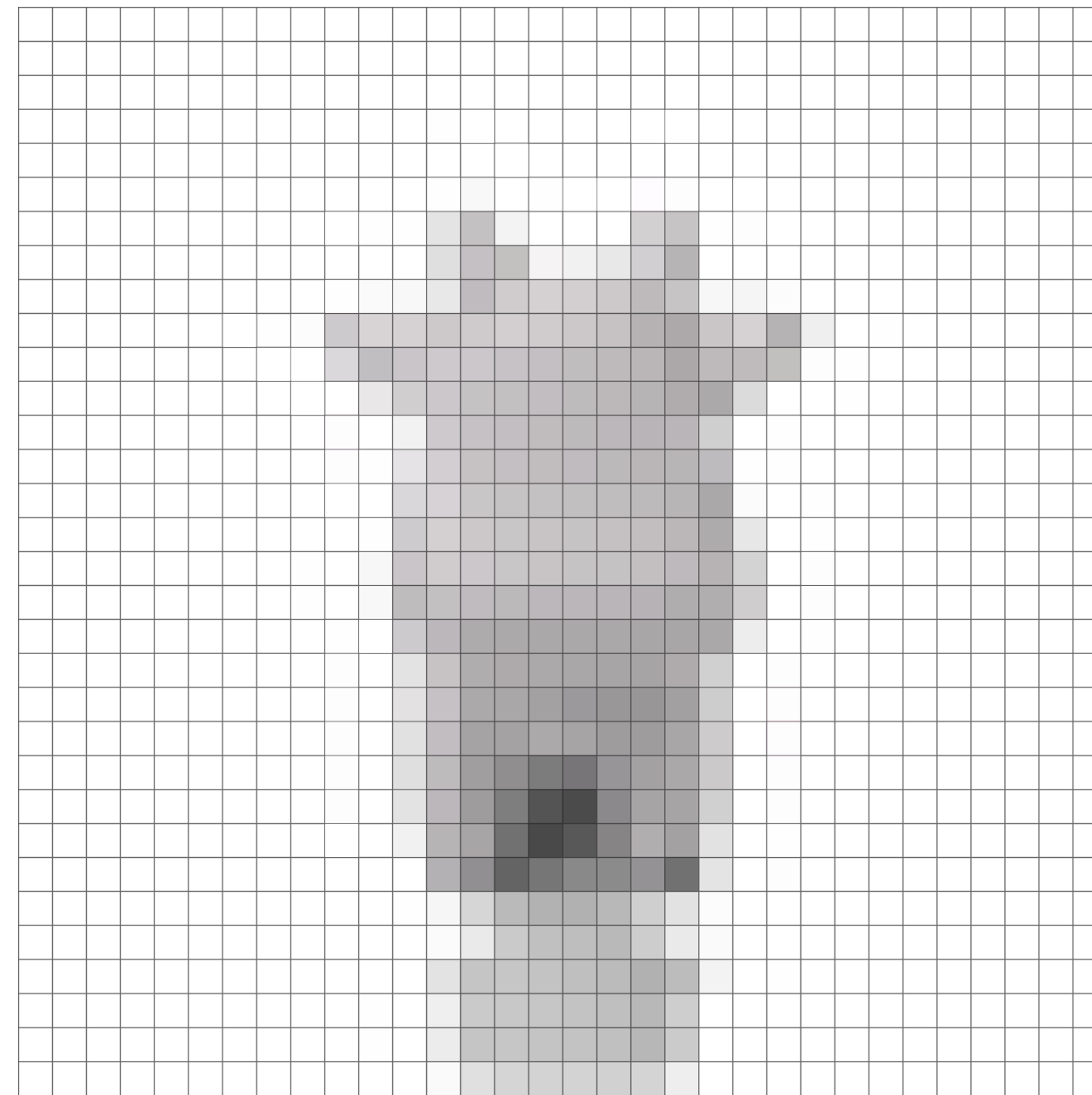
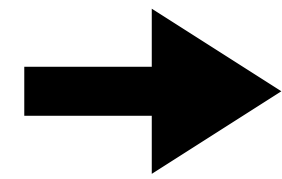
# Future Work: Compute & back project image gradients



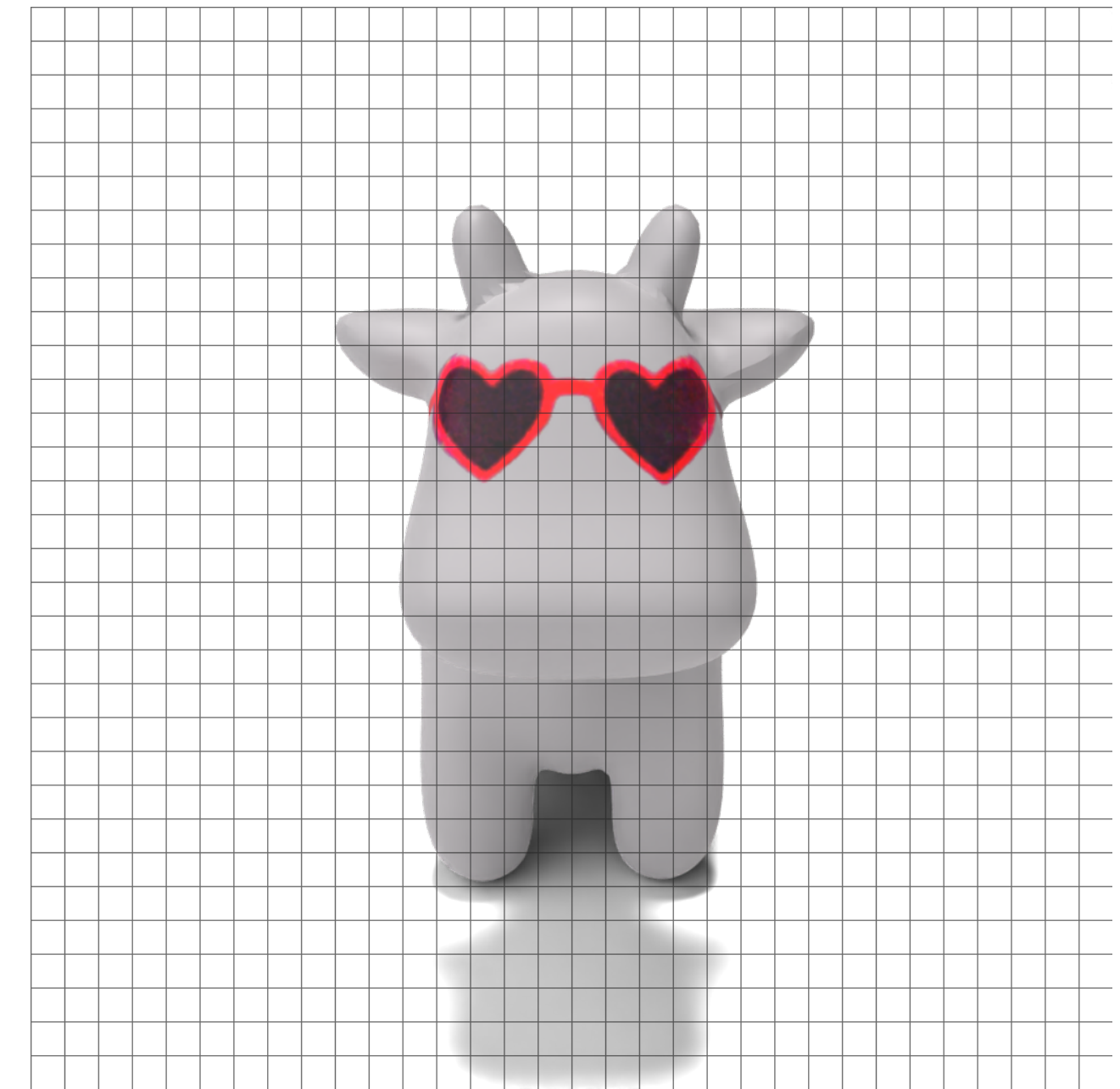
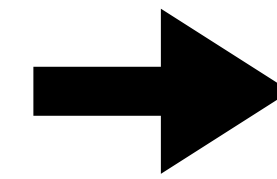
# Future Work: Compute & back project image gradients



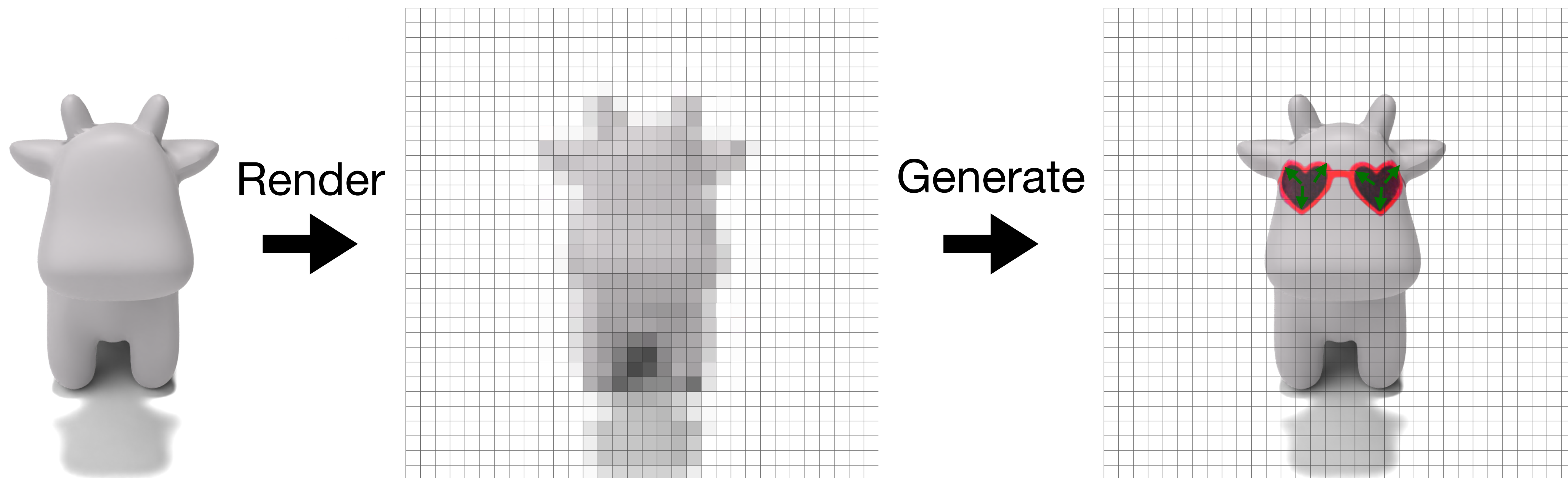
Render



Generate

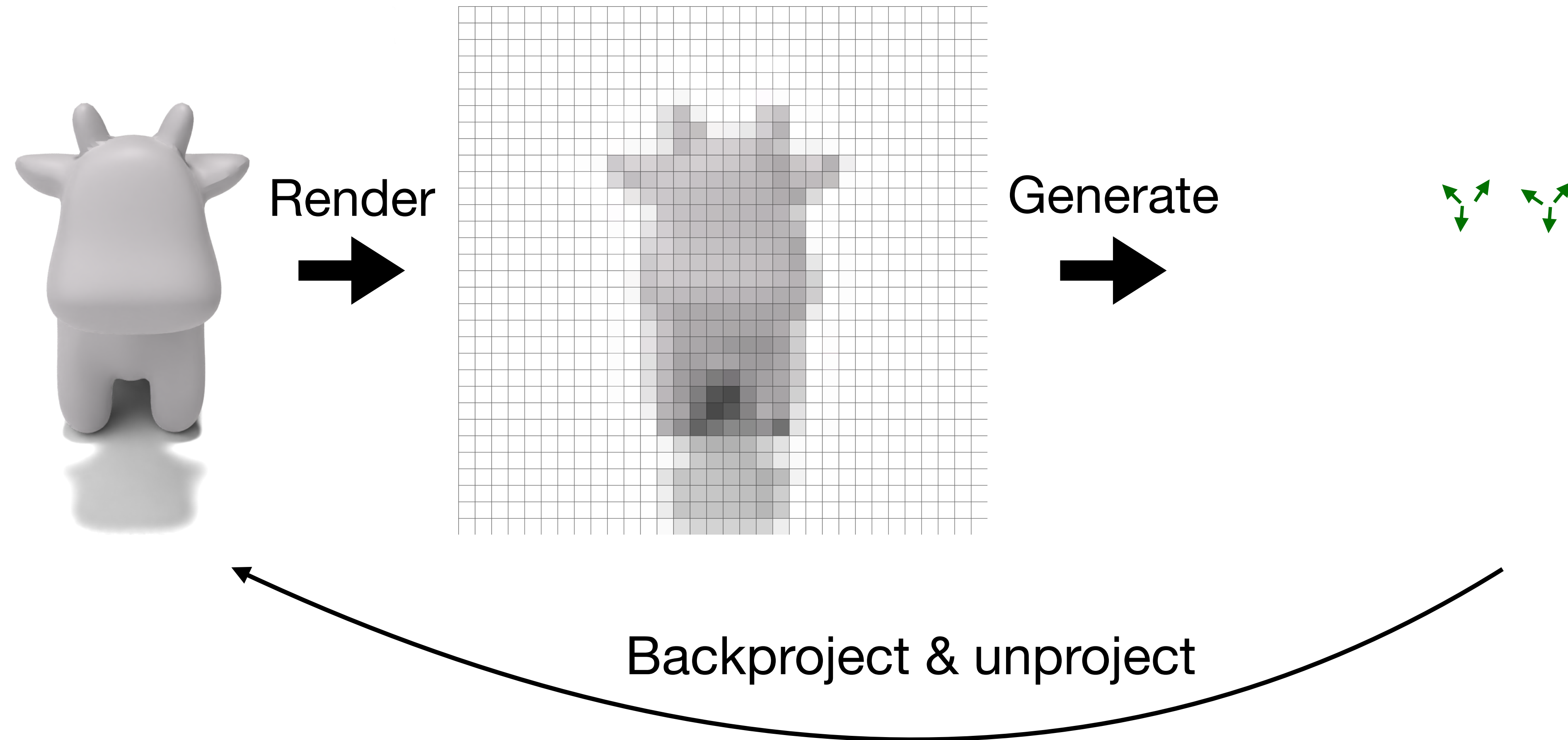


# Future Work: Compute & back project image gradients

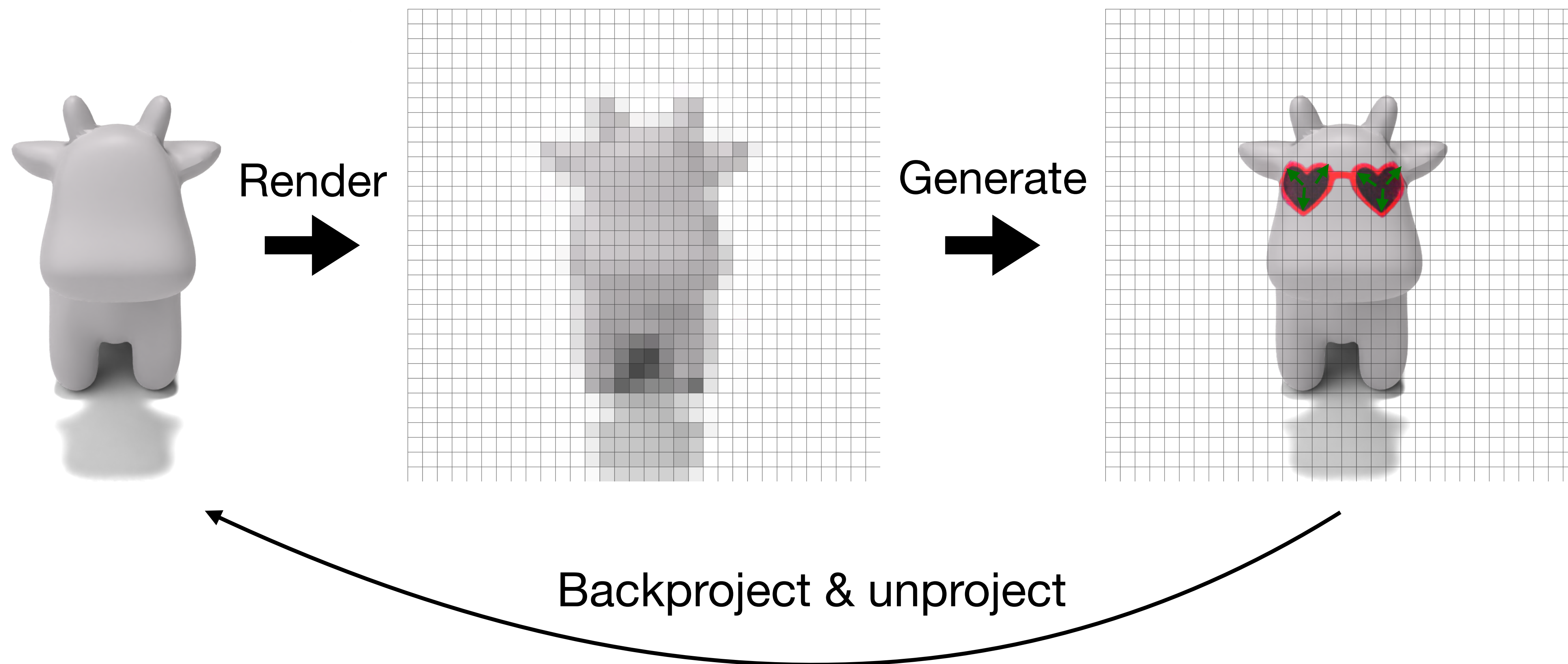




# Future Work: Compute & back project image gradients



# Future Work: Compute & back project image gradients



# Conclusion



# Conclusion

2D priors are useful for 3D tasks when:

# Conclusion

2D priors are useful for 3D tasks when:

- 3D data is scarce

# Conclusion

2D priors are useful for 3D tasks when:

- 3D data is scarce
- The objective benefits from visual reasoning



# Conclusion

2D priors are useful for 3D tasks when:

- 3D data is scarce
- The objective benefits from visual reasoning
- The quantity can be visually represented

# Conclusion

2D priors are useful for 3D tasks when:

- 3D data is scarce
- The objective benefits from visual reasoning
- The quantity can be visually represented

Not relevant for all tasks, but can be very powerful when leveraged properly!

**Thank you!**