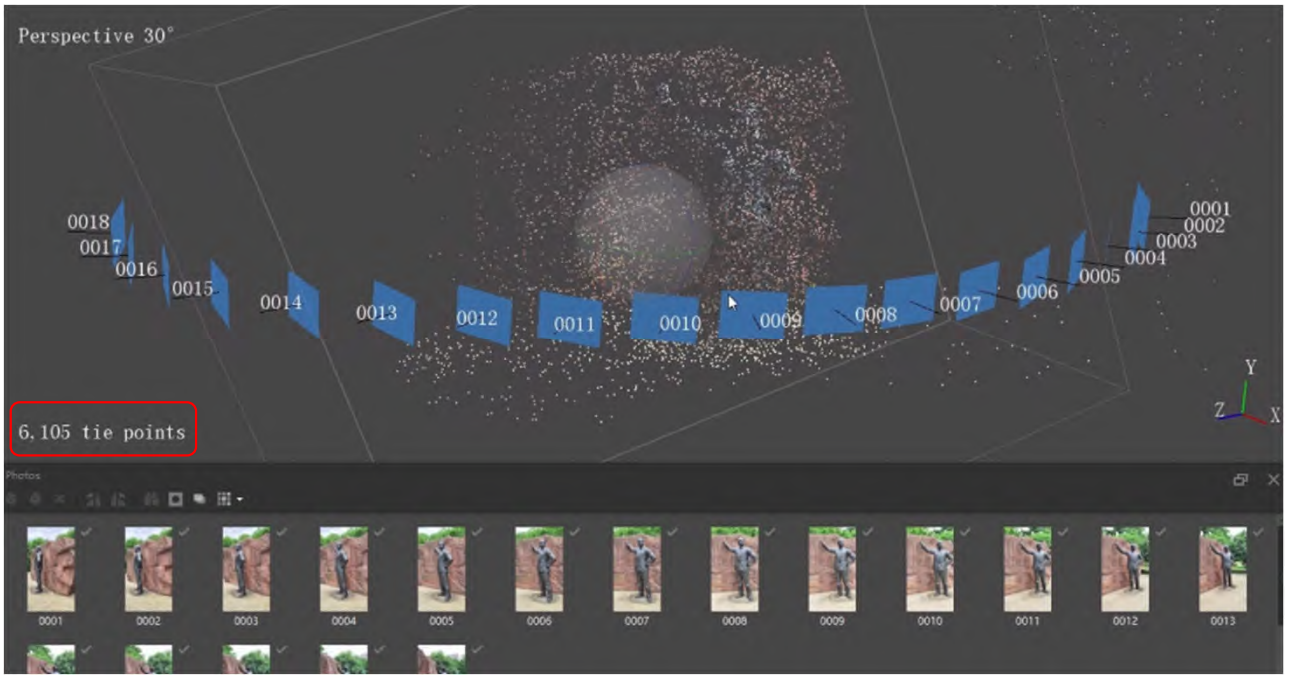
单视角三维重建
3D from Single View

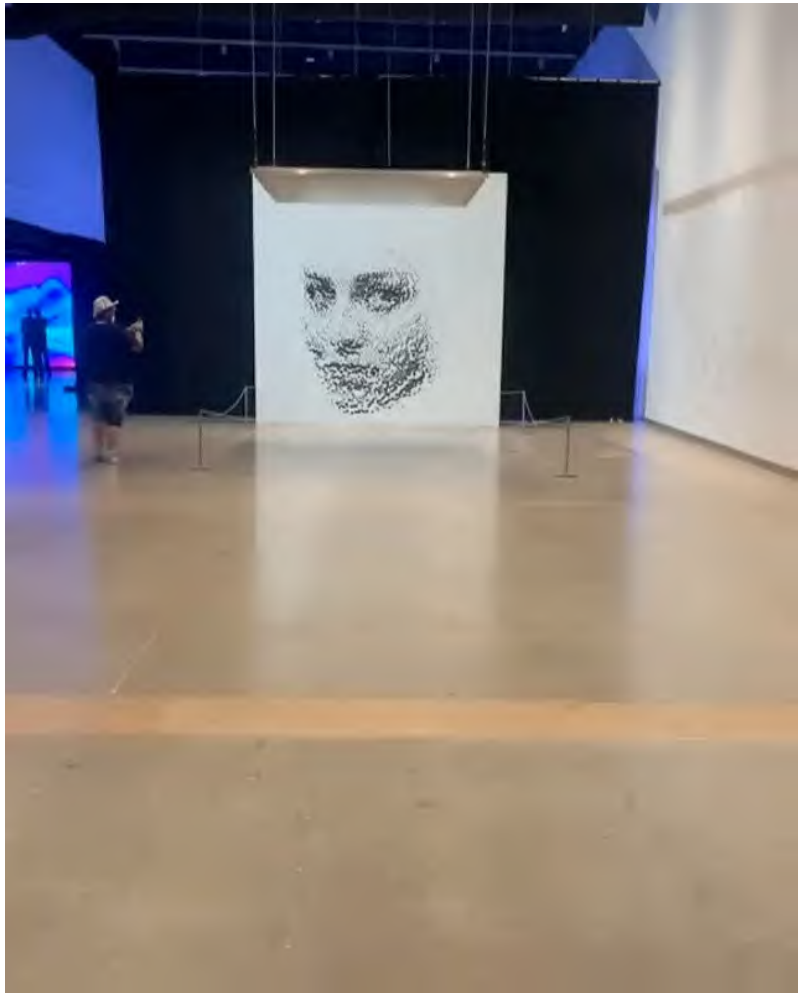
Review Multi-view

Review Multi-view



But Points for Rendering…

But Points for Rendering…



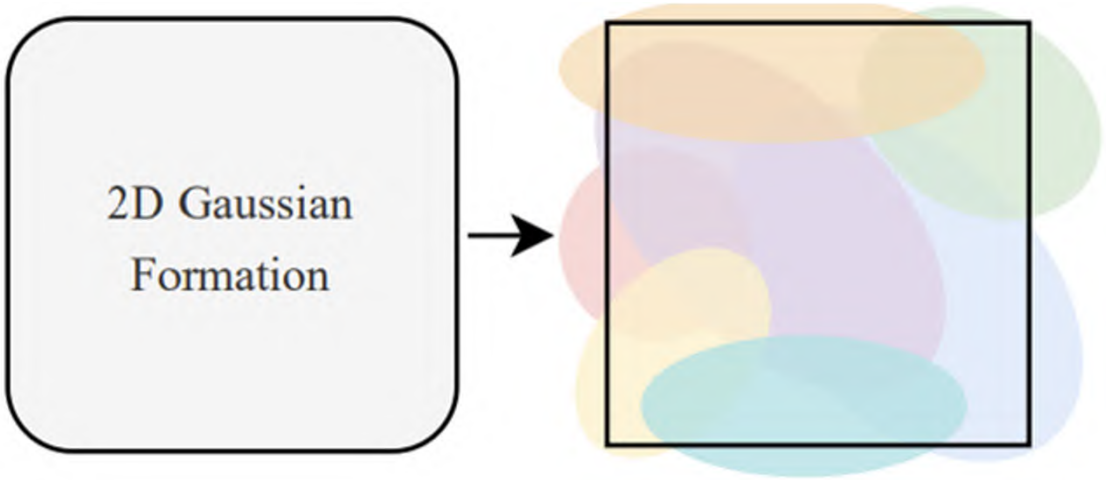
因每个点影响范围有限（Point Size / Radius）

稀疏点云渲染稀疏像素图像

稠密点云O(N^3)

How to render efficiently & perfectly with Points?

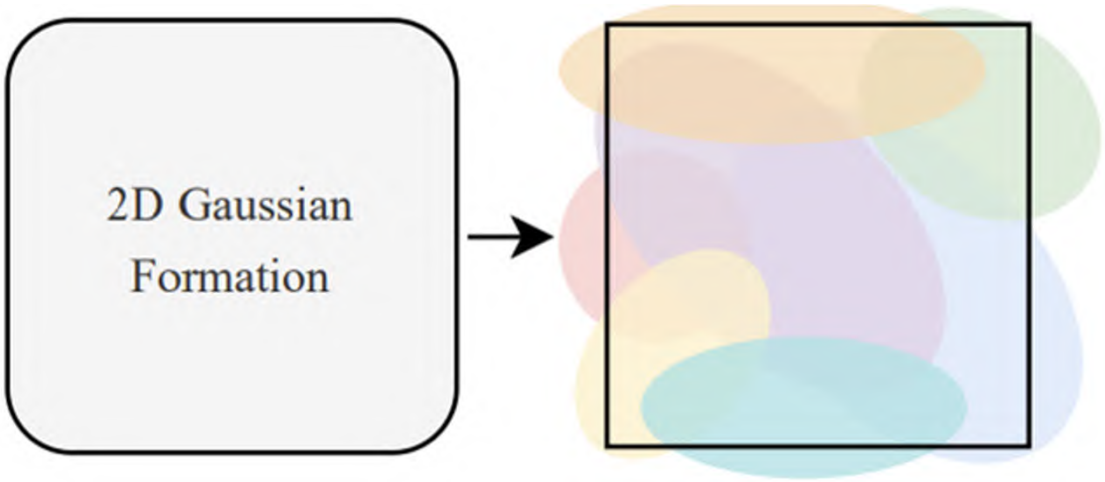
Gaussian Splatting — 从图像上理解



能否通过若干二维高斯（高斯分布体现在密度 / 权重 上，每个高斯有一个常数的颜色）

通过加权求和来逼近左边的图像？

Detailed Formulation



position $\boldsymbol{\mu} \in \mathbb{R}^2$ 2D covariance matrix $\boldsymbol{\Sigma}$

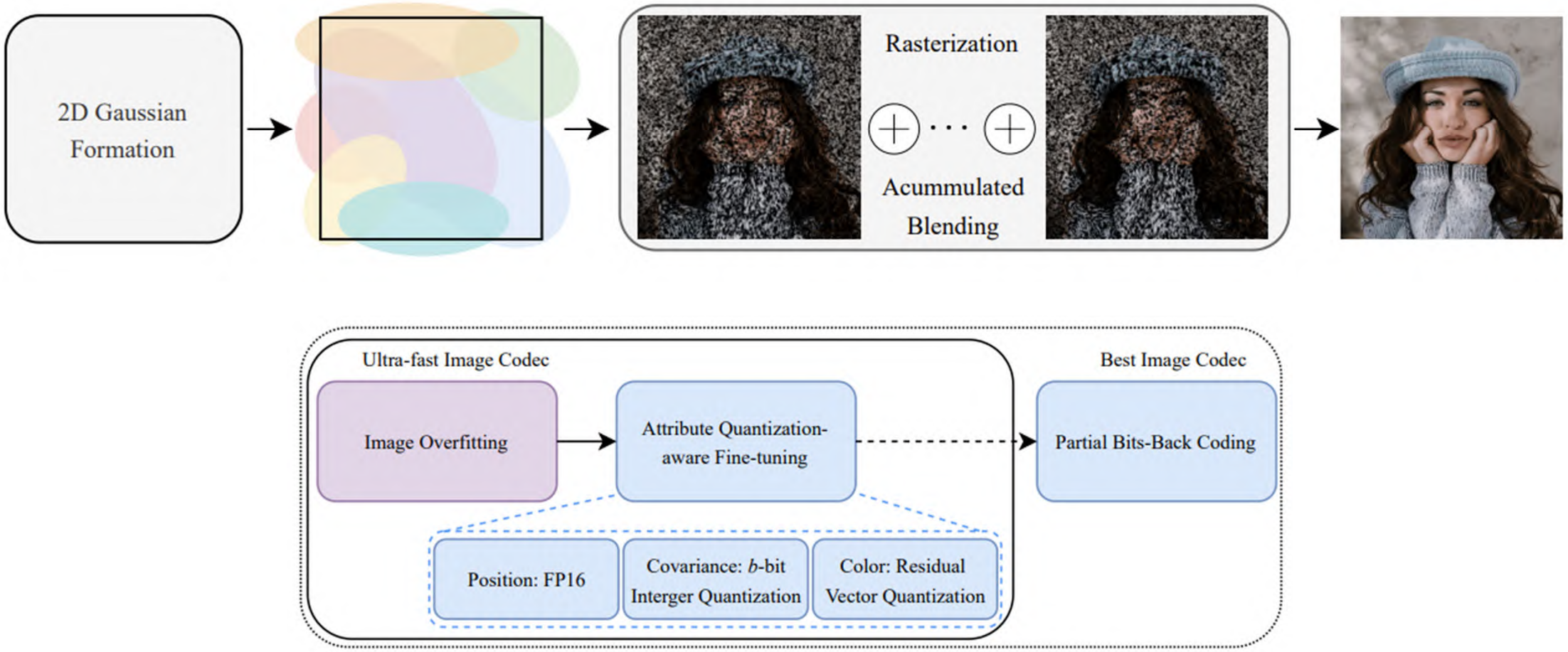
color coefficients $\boldsymbol{c} \in \mathbb{R}^3$ opacity $o \in \mathbb{R}$

$$\sigma_n = \frac{1}{2}\boldsymbol{d}_n^T \boldsymbol{\Sigma}^{-1} \boldsymbol{d}_n$$

$$\boldsymbol{C}_i = \sum_{n \in \mathcal{N}} \boldsymbol{c}_n \cdot \alpha_n = \sum_{n \in \mathcal{N}} \boldsymbol{c}_n \cdot o_n \cdot \exp(-\sigma_n).$$

$$\boldsymbol{C}_i = \sum_{n \in \mathcal{N}} \boldsymbol{c}'_n \cdot \exp(-\sigma_n),$$

For Image Compression



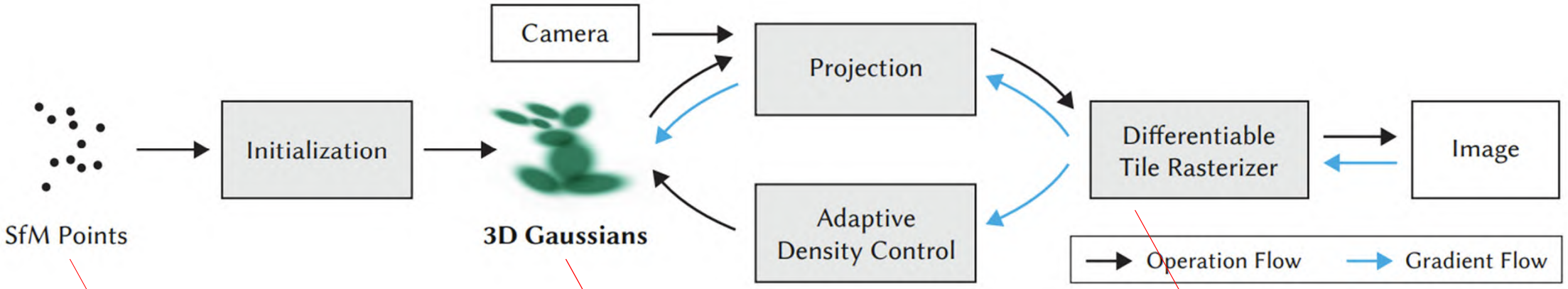
For Image Compression

Fig. 5: Subjective comparison of various codecs on Kodak at low Bpp.

Table 2: Computational complexity of traditional and learning-based image codecs on DIV2K Dataset at low and high Bpp.

Methods	Bpp↓	PSNR↑	MS-SSIM↑	Encoding FPS↑	Decoding FPS↑
JPEG [61]	0.3197/0.5638	25.2920/28.4299	0.9020/0.9559	608.61/557.35	614.68/545.59
JPEG2000 [55]	0.2394/0.5993	27.2792/30.9294	0.9305/0.9663	3.46/3.40	4.32/3.93
Ballé17 [5]	0.2271/0.4987	27.7168/30.7759	0.9508/0.9775	21.23/16.53	18.83/17.87
Ballé18 [6]	0.2533/0.5415	28.7548/32.2351	0.9584/0.9816	16.53/13.56	15.87/15.20
COIN [23]	0.3419/0.6780	25.8012/27.6126	0.8905/0.9306	$5.30e^{-4}/3.51e^{-4}$	166.31/93.74
Ours	0.3221/0.6417	25.6631/27.5656	0.9154/0.9483	$4.11e^{-3}/4.73e^{-3}$	1970.76/1980.54

Point Cloud -> 3DGS

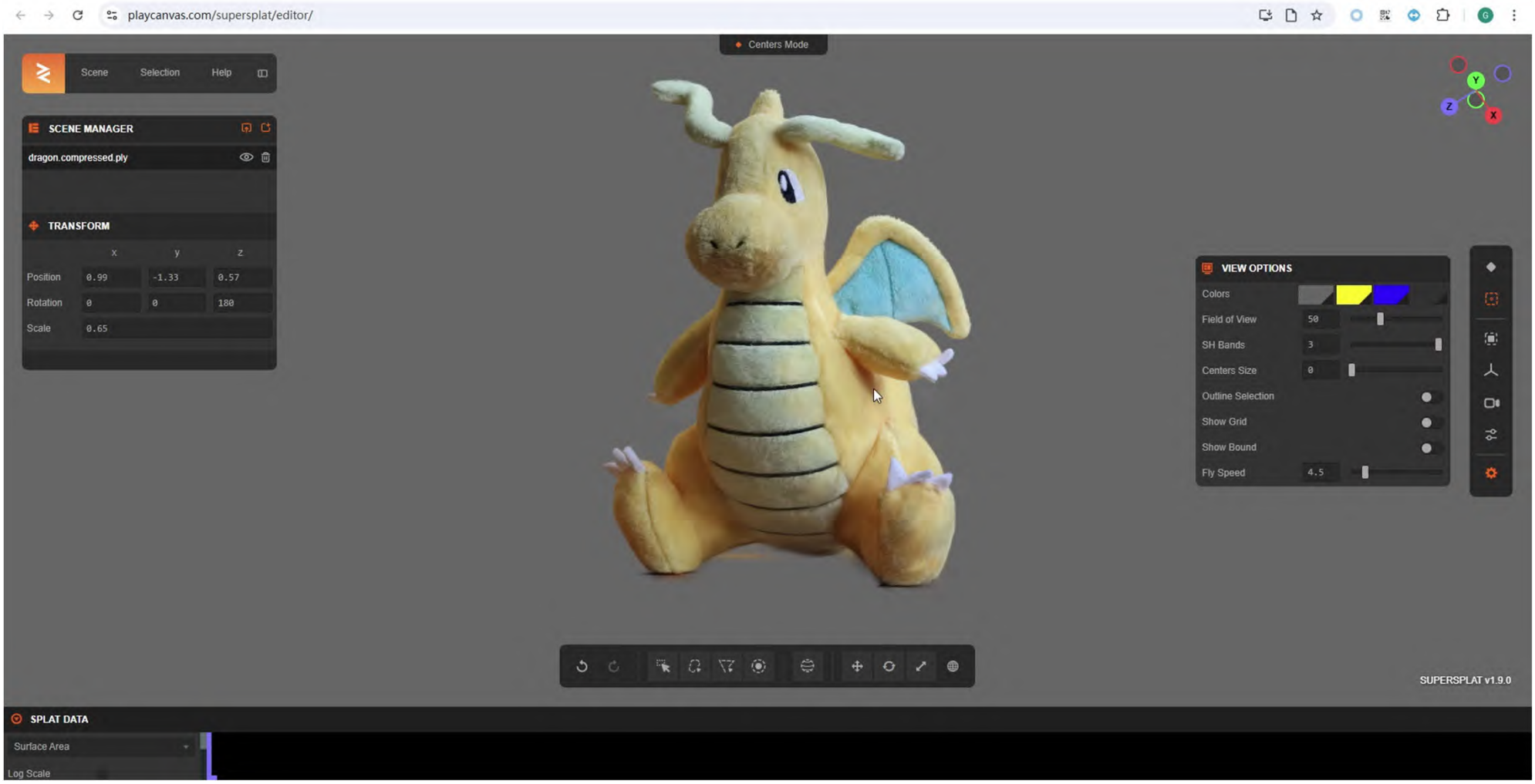


多视角立体匹配得到的点云

给这些点云赋予三维协方差、密度、颜色

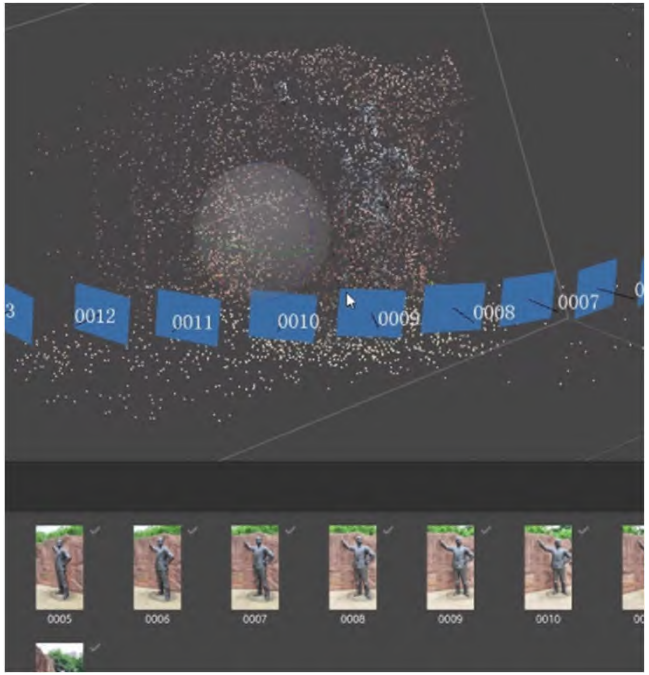
基于投影 + 混合（Blending）的三维高斯渲染

Point Cloud -> 3DGS

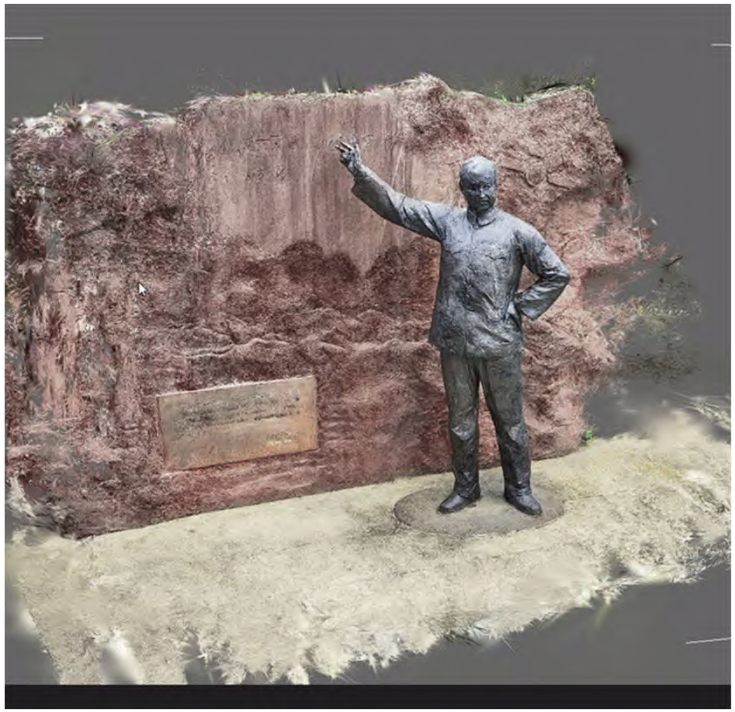


Pipeline of Realistic Rendering from Multi-view

Multi-View Images



Point Clouds &
Cameras



3DGS / NeRF

Today's Topic

Today's Topic

Task

Single 2D Image

How?

What?

3D Representation
of the underlying world

Ambiguity

"Ames room"

The good news: A world with regularities

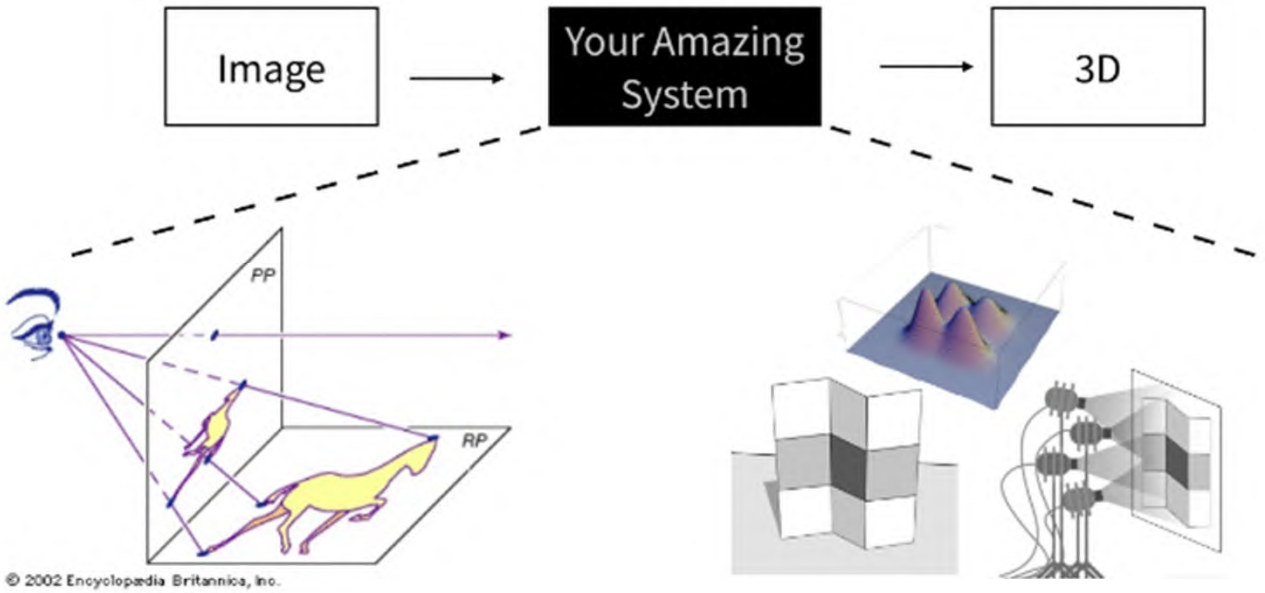
Abstract World

Our World

Infinite possible explanations - but some more likely than others!

Image credits: *Derek Hoiem*

The good news: A structured world

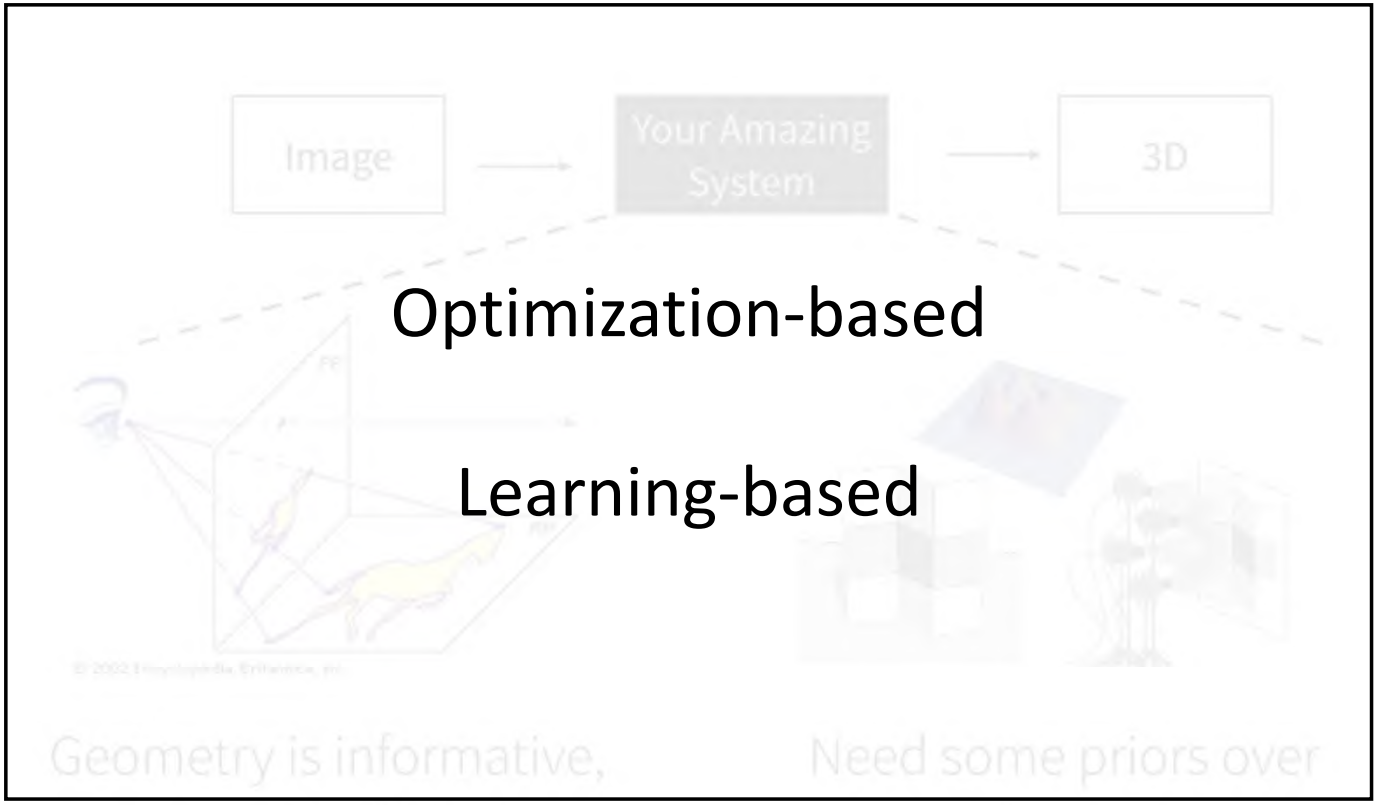


Geometry is informative, but **not** sufficient

Need some priors over possible structures

Reconstruct with Prios

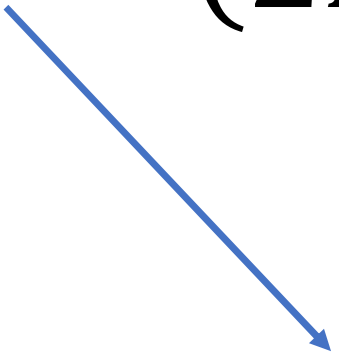
The good news: A structured world



Reconstruct with Prios

Mathematic Formulation

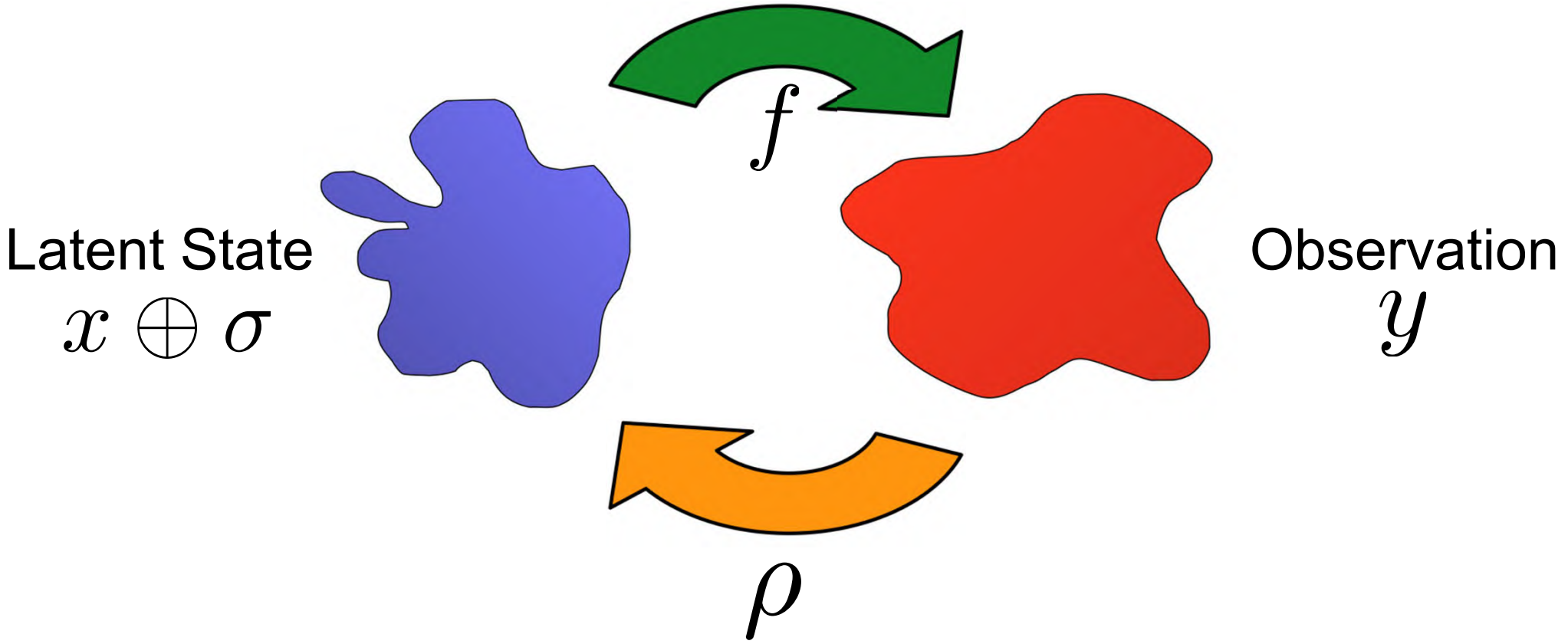
$$F^{-1}(2D) = 3D$$



Reconstruction: the inverse function

Solve for 3D from 2D observation

Inverse Problem

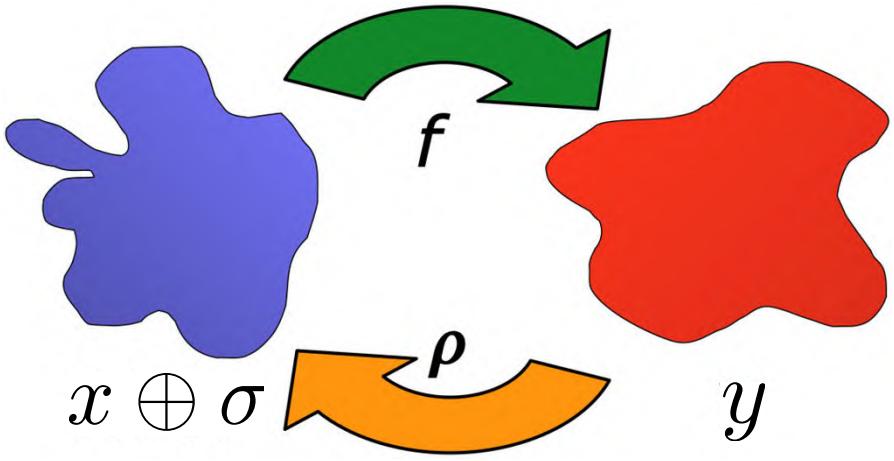


Slide credit: Ben Recht

Inverse Problem Solutions

Optimization! Estimate x from y using the forward model

$$\rho(y) = \arg\min_{x \in X} \ell_y(f(x), y)$$

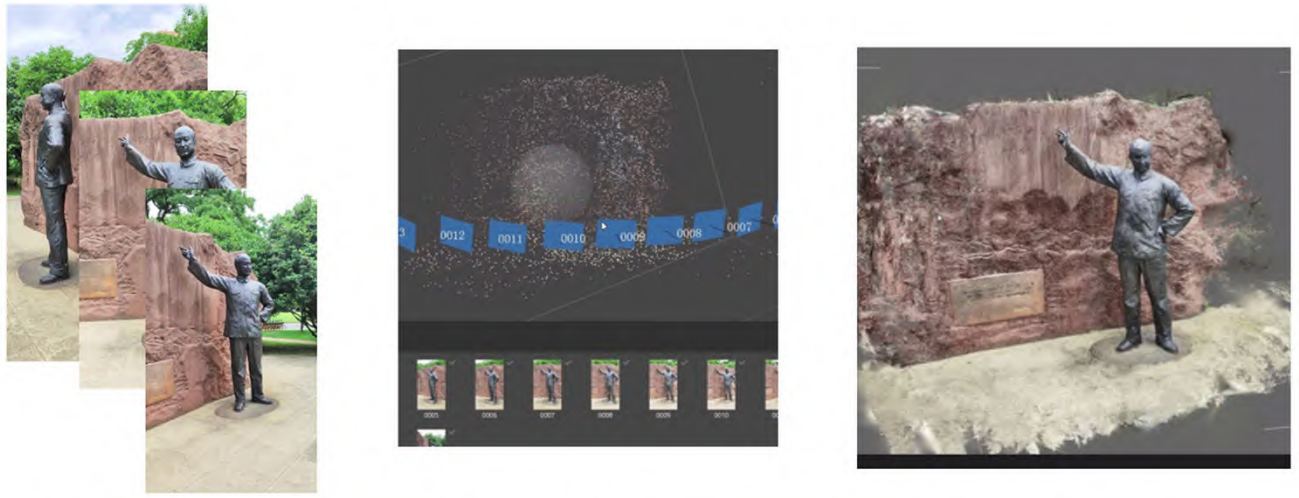


Machine Learning! Estimate $\boldsymbol{\rho}$ from examples

$$\rho(y) = \arg\min_{\varphi \in \mathcal{F}} \sum_{I=1}^{n} l_x(\varphi(y_i), x_i)$$

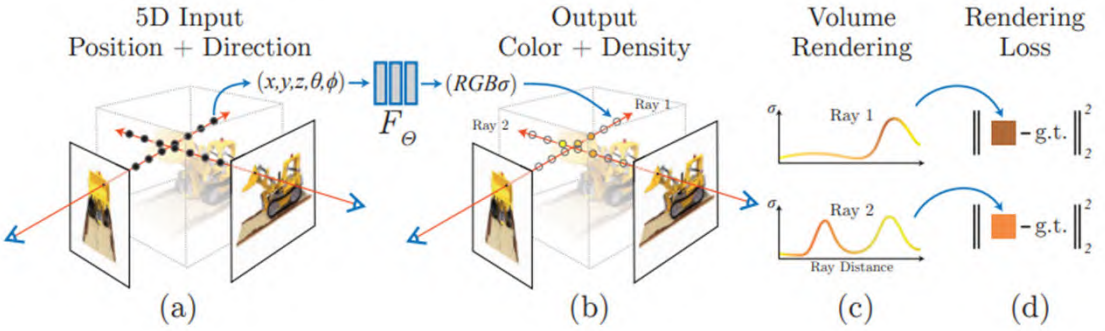
Slide credit: Ben Recht

Optimization (from Multi-view)

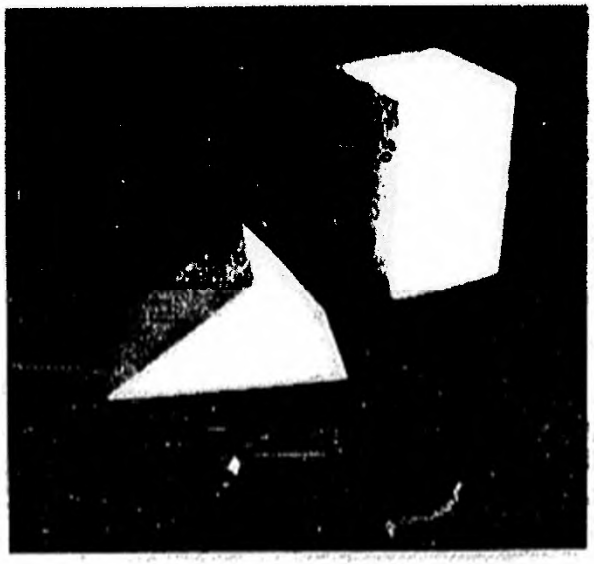


$$\min_{\{\mathbf{R}_k,\mathbf{t}_k\},\mathbf{K}} \sum_i \sum_k \|\mathbf{x}_{ik} - \pi_{\mathbf{K}}([\mathbf{R}_k|\mathbf{t}_k]\mathbf{X}_i)\|_2^2$$

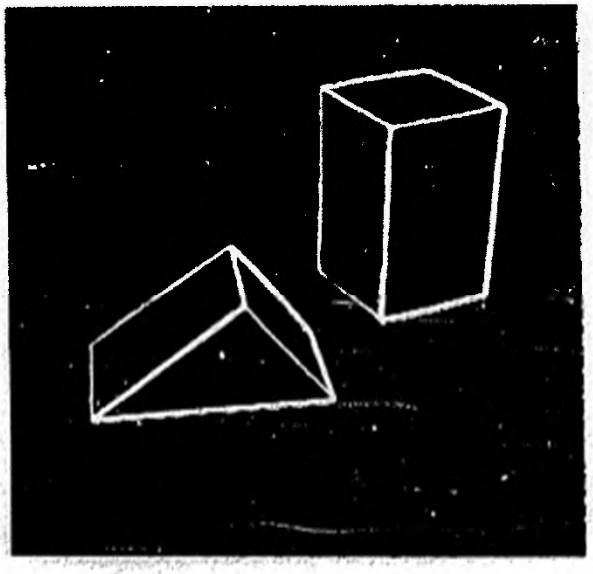
Extrinsic — Intrinsic — Detection Corner Points — Perspective Projection — Known 3D Location



Optimization from Single-view with Prior



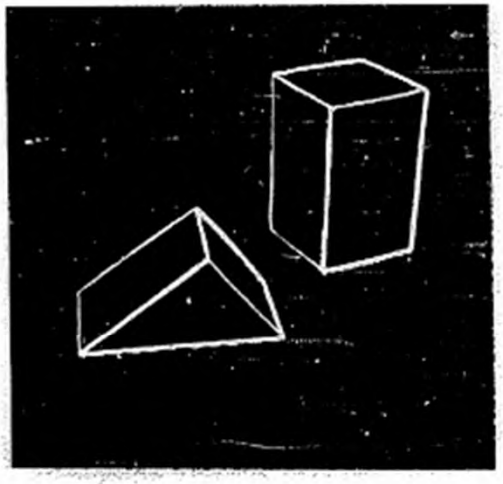
A. Original Picture



B. Differentiated Picture

Machine perception of three-dimensional solids.
Roberts, 1963. PhD Thesis, MIT.

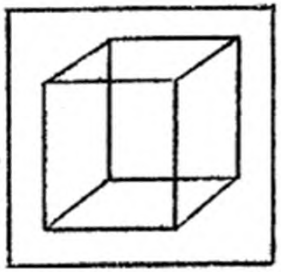
Optimization from Single-view with Prior

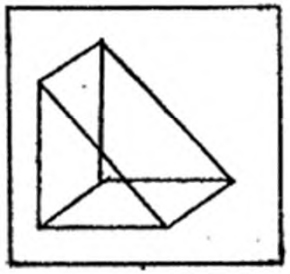


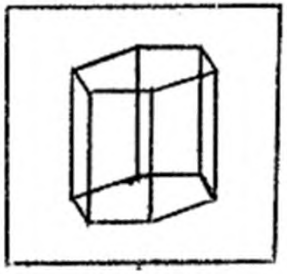
B. Differentiated Picture

Predefined Structures

MODEL STRUCTURES





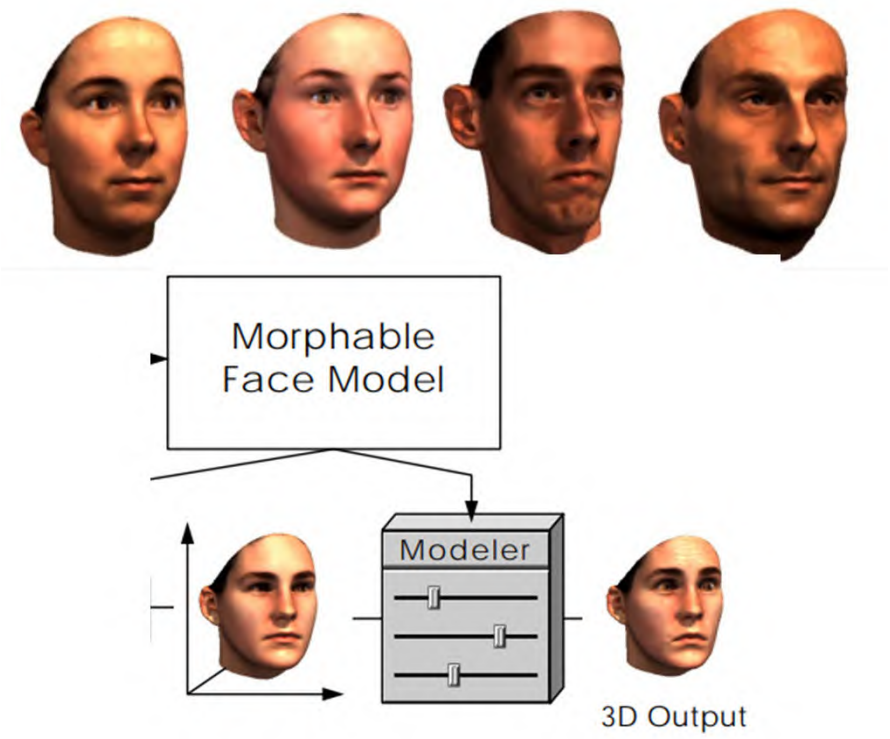


is topology. Basically, we wish to find points in the line drawing which fit a transformation of some model. The polygon structure is used to

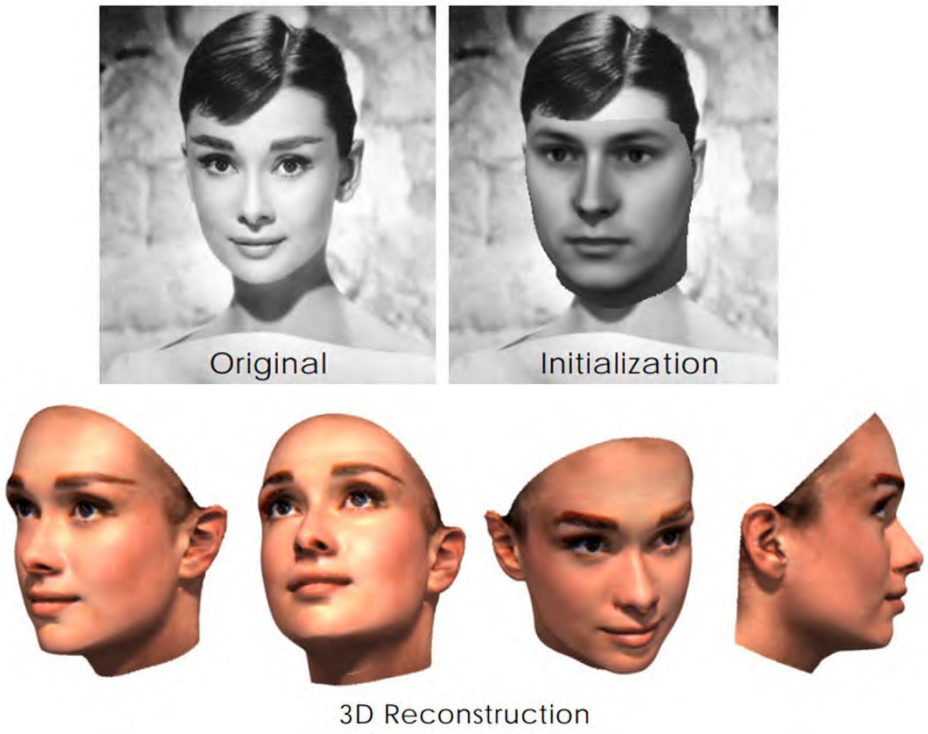
Optimization with Predefined Face Structure

1. Learn a morphable model from 3D data

PCA Model



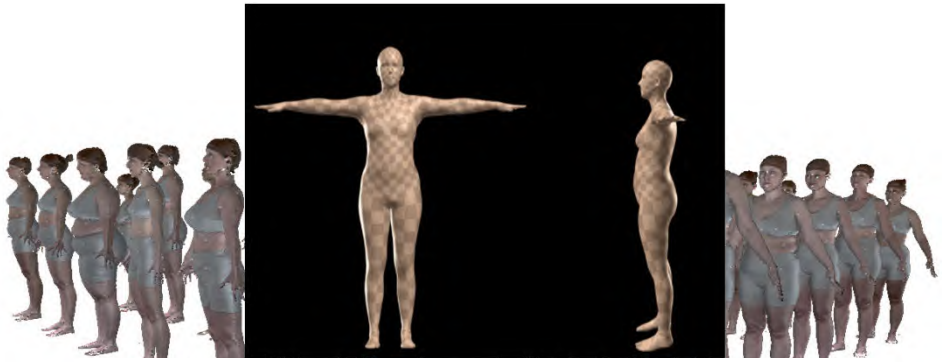
2. Fit the model to images



Blanz & Vetter '99 SIGGRAPH

Optimization with Predefined Body Structure

1. Learn a morphable model from 3D data



+ kinematic tree =SMPL

[Loper et al. SIGGRAPH ASIA '15]

2. Fit the model to images

Alternatively, if enough data, learn to predict

$$\min_{\beta,\theta,\Pi} \left|\left| \quad - \Pi(\quad) \right|\right|_2^2$$

+ lots of priors

[Bogo and Kanazawa et al ECCV '16]

The best of this before deep learning

Subject: Arnold Old
Original video resolution: 1280x720

Source: https://youtu.be/EgvdhvKreJI
Courtesy of DECC GovUK

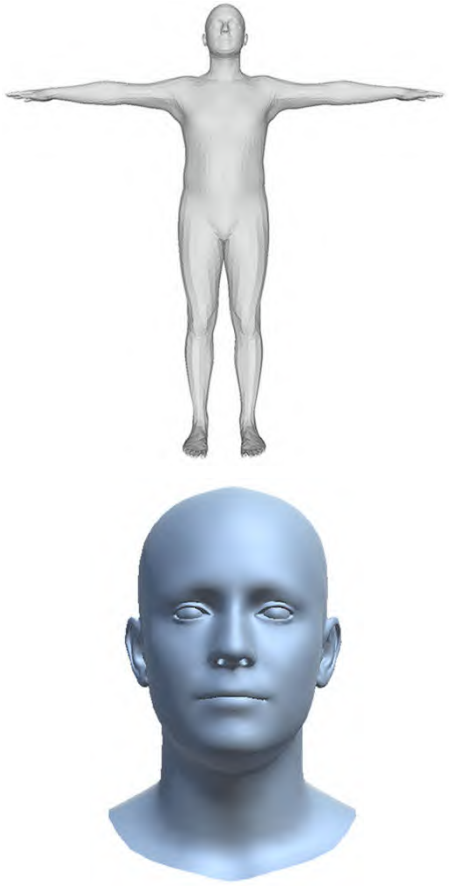
Reconstruction of Personalized 3D Face Rigs from Monocular Video. SIGGRAPH 2016.

The best of this before deep learning

The best of this before deep learning

Photo Wake-Up: 3D Character Animation from a Single Photo. CVPR 2019.

But pre-define structure is difficult

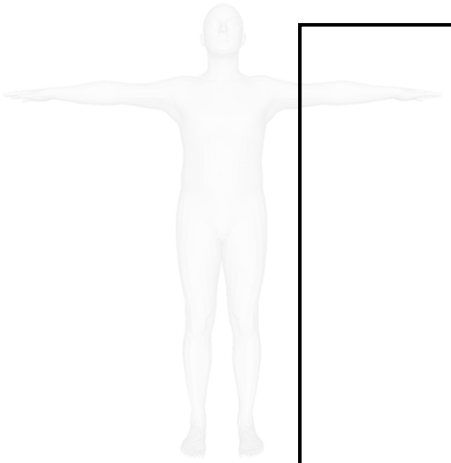


It’s easy to define structure for face / body

But it’s difficult to define it for general objects

Figure 1: We introduce the **Common Objects in 3D (CO3D)** dataset comprising 1.5 million multi-view images of almost 19k objects from 50 MS-COCO categories annotated with accurate cameras and 3D point clouds (visualized above).

But pre-define structure is difficult



It's easy to define structure for face / body

But it's difficult to define it for general objects

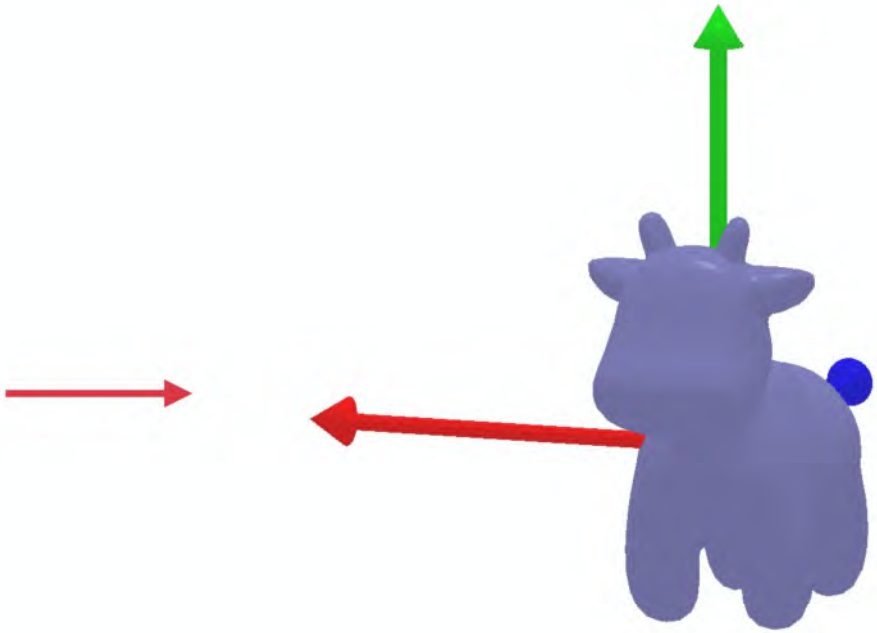
Optimization-based

Turn to Deep Learning

Figure 1: We introduce the **Common Objects in 3D (CO3D)** dataset comprising 1.5 million multi-view images of almost 19k objects from 50 MS-COCO categories annotated with accurate cameras and 3D point clouds (visualized above).

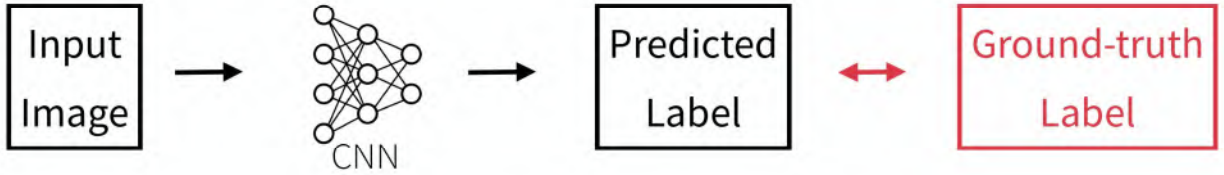
Task

2D Image depicting a
Single Object

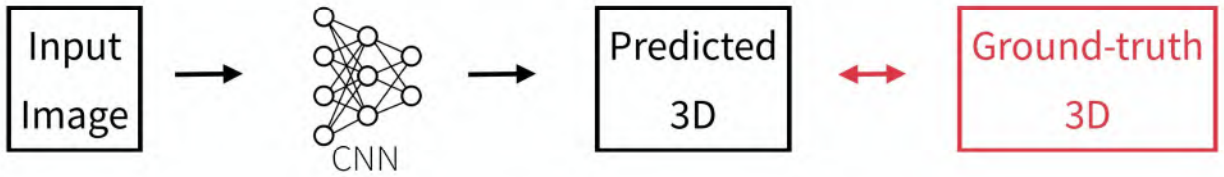


3D in canonical frame

Learning from Direct Supervision



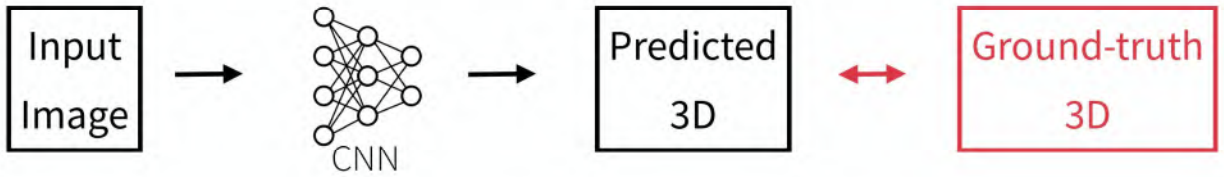
Object-centric 3D: Supervised Learning



A caricature recipe for learning:

- Step 0: Decide on model and objectives
- Step 1: Collect training data (lots of [image, 3D] pairs)
- Step 2: Learn a predictor
  - Step 2a: Wait a few days, drink coffee and watch training curves
- Use the predictor!

Object-centric 3D: Supervised Learning



A caricature recipe for learning:

- Step 0: Decide on model and objectives
- **Step 1: Collect training data (lots of [image, 3D] pairs)** (not easy to do)
- Step 2: Learn a predictor
  - Step 2a: Wait a few days, drink coffee and watch training curves
- Use the predictor!

Object-centric 3D: Learning from Synthetic Data

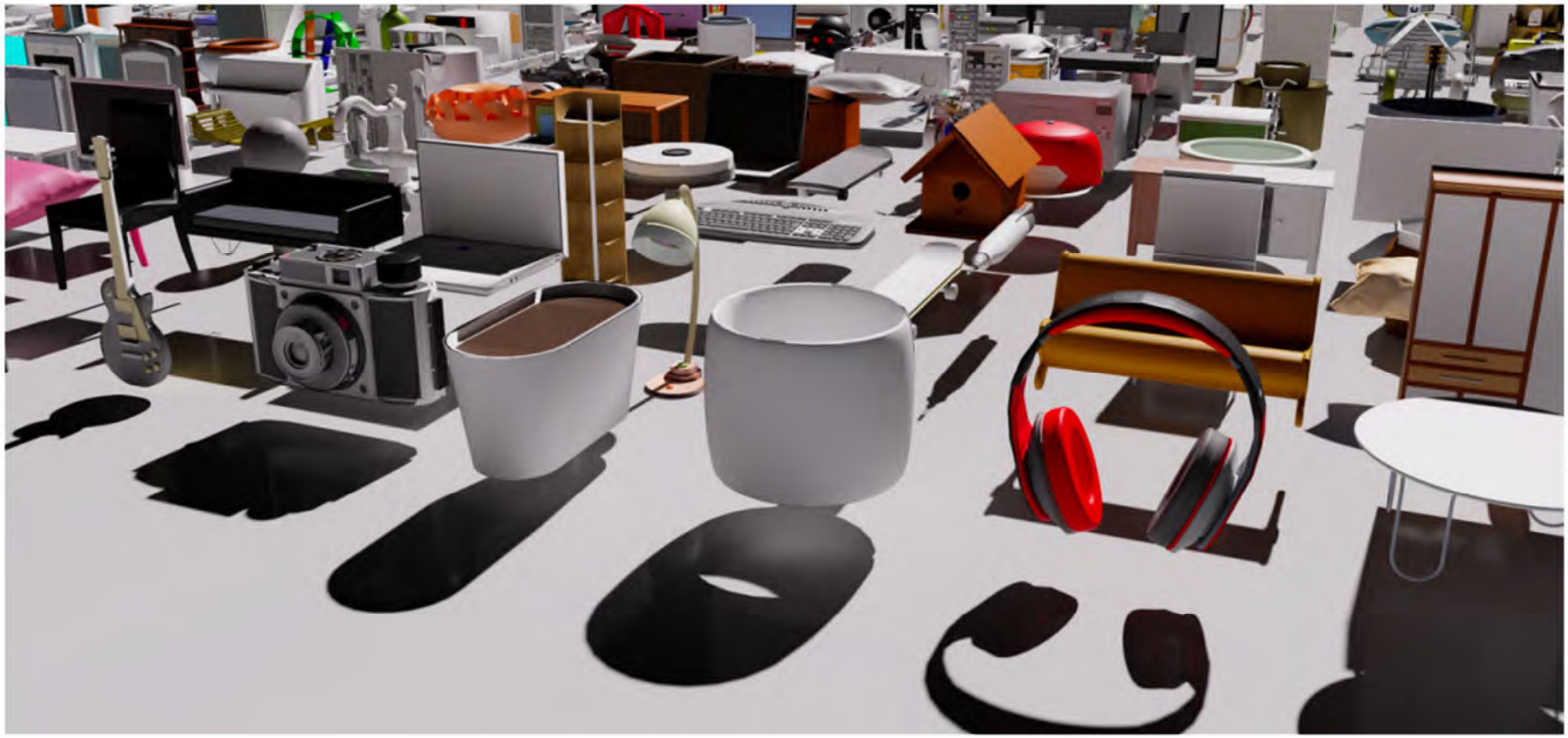
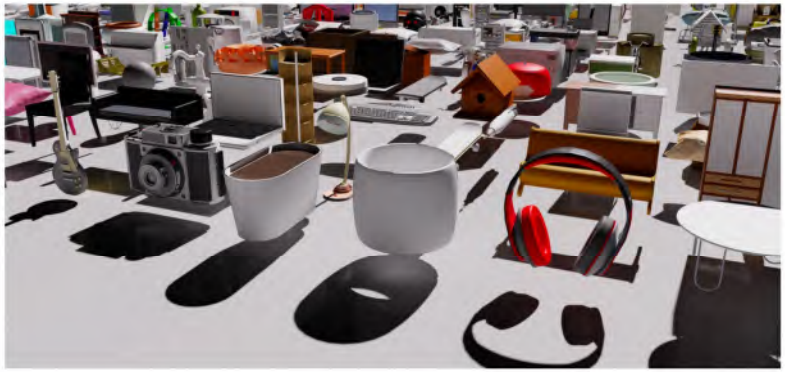


Image Credits: NVIDIA Omniverse ShapeNet importer

Learning from Synthetic Data: A word of caution



- Great for debugging, analyzing and testing ideas

- Not very realistic — typically simple appearance/material

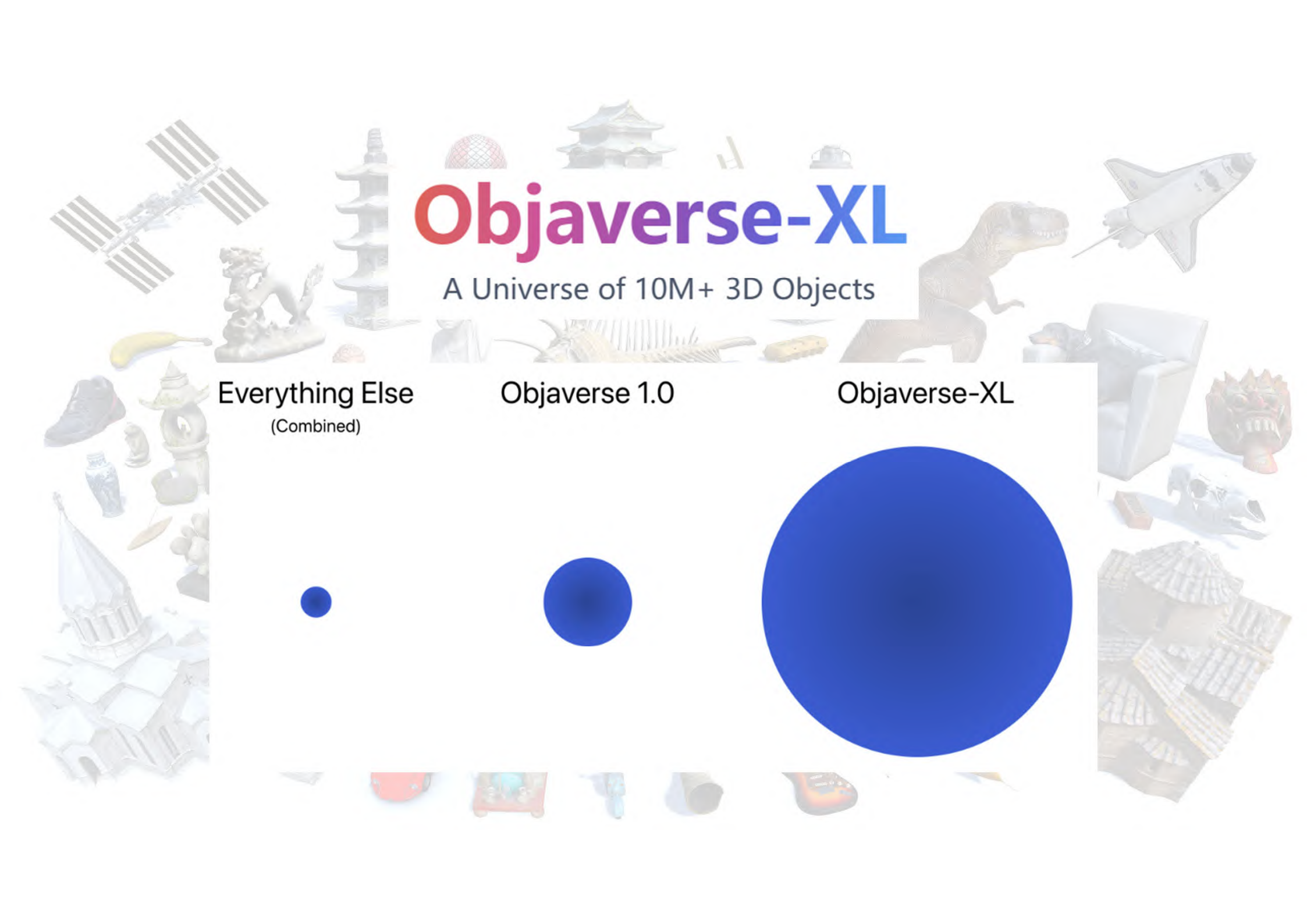
- Biased towards rigid, artificial categories

- Progress on synthetic data is **not** the same as progress on real-world tasks (e.g. did the approach make assumptions specific to synthetic data?)

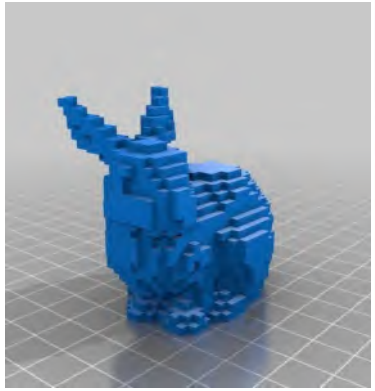
Image Credits: NVIDIA Omniverse ShapeNet importer

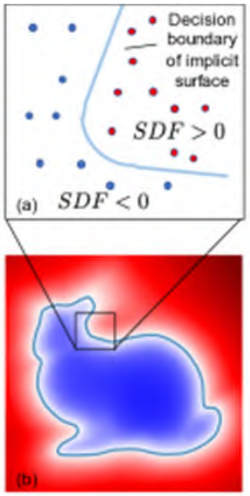
Objaverse-XL

A Universe of 10M+ 3D Objects



2.5D / Image Based Rendering

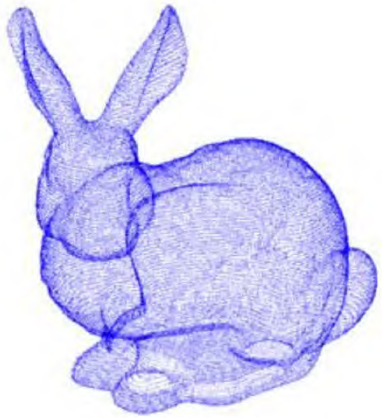




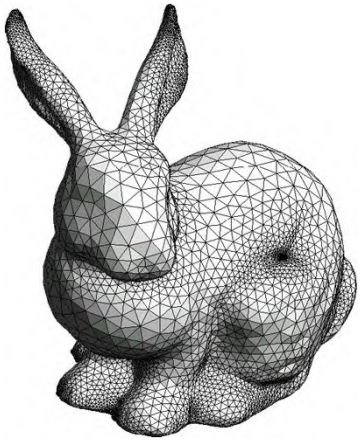
Volumetrics

Explicit

Implicit



Point clouds



Meshes

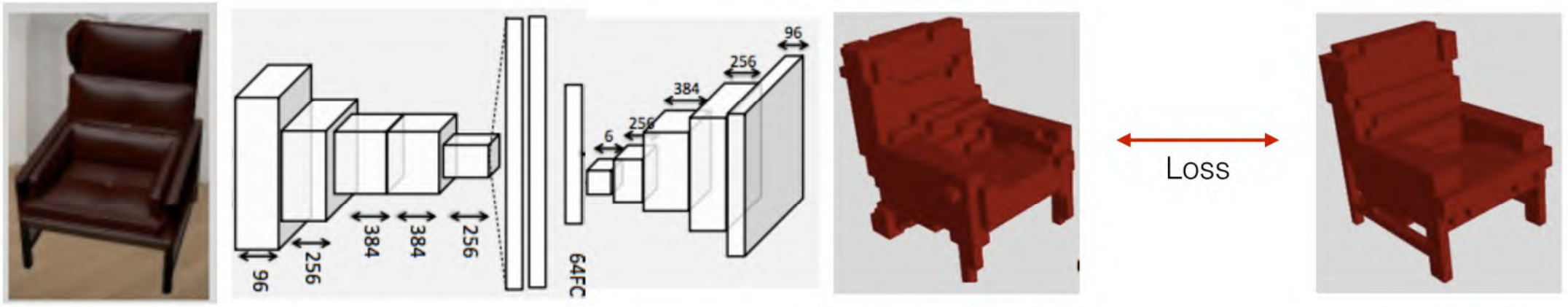
Different output representation, different architectures

Learning to Predict Volumetric 3D

$$V[x, y, z] \in \{0,1\}$$

Image Credits: Learning a Predictable and Generative Vector Representation for Objects. Girdhar et. al., ECCV 2016

Learning to Predict Volumetric 3D



Continuous Probability

Binary Label

Maximize log-likehood of GT label in each voxel:

$$-\frac{1}{N}\sum_{n=1}^{N}[p_n \log \hat{p}_n + (1-p_n)\log(1-\hat{p}_n)]$$

Image Credits: Learning a Predictable and Generative Vector Representation for Objects. Girdhar et. al., ECCV 2016

Learning to Predict Volumetric 3D

Fig. 3: Sample renderings used to train our network. We render each training model into 72 views over a random background each epoch of training.

Training data: Renderings of CAD models on random backgrounds (helps transfer to real images)

Image Credits: Learning a Predictable and Generative Vector Representation for Objects. Girdhar et. al., ECCV 2016

Learning to Predict Volumetric 3D

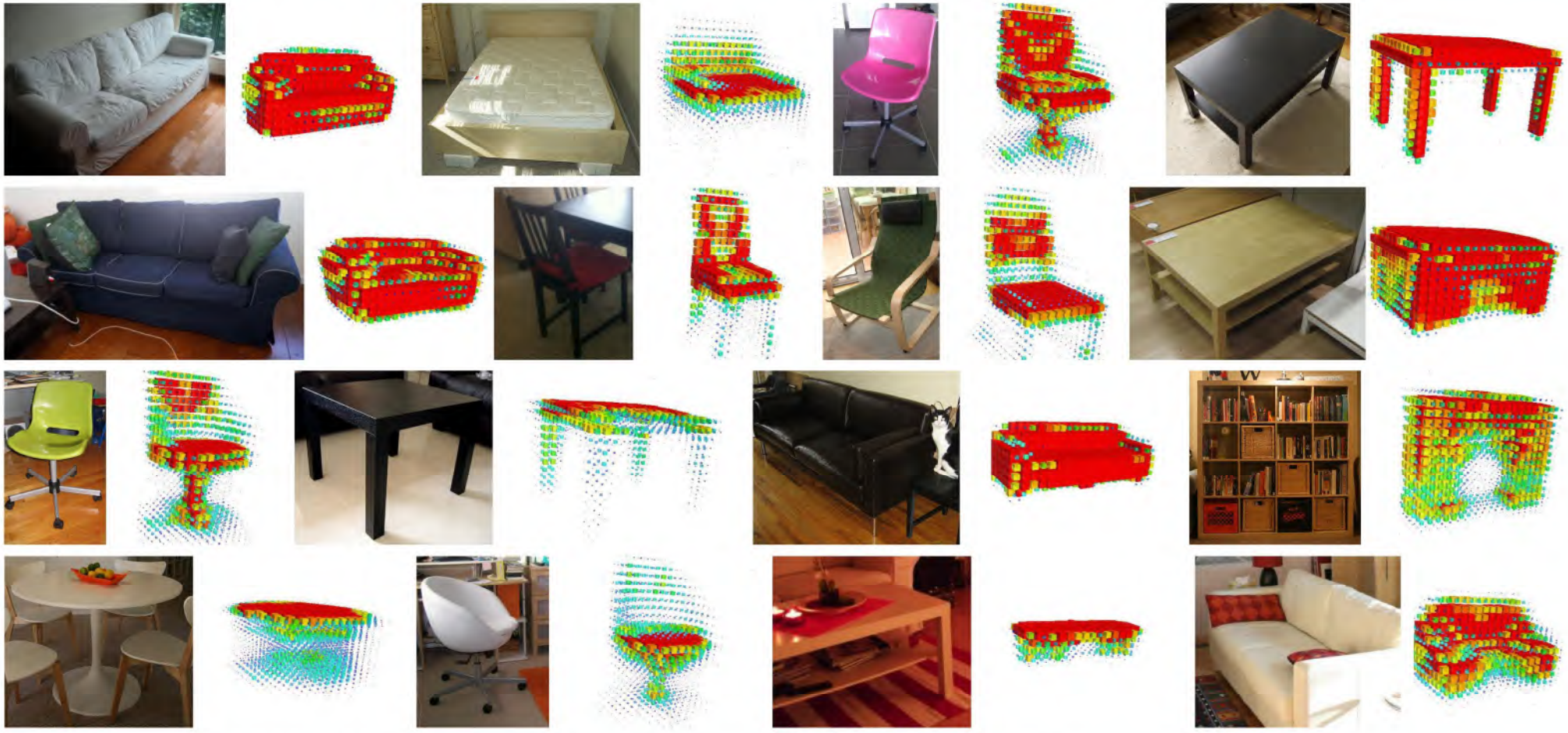
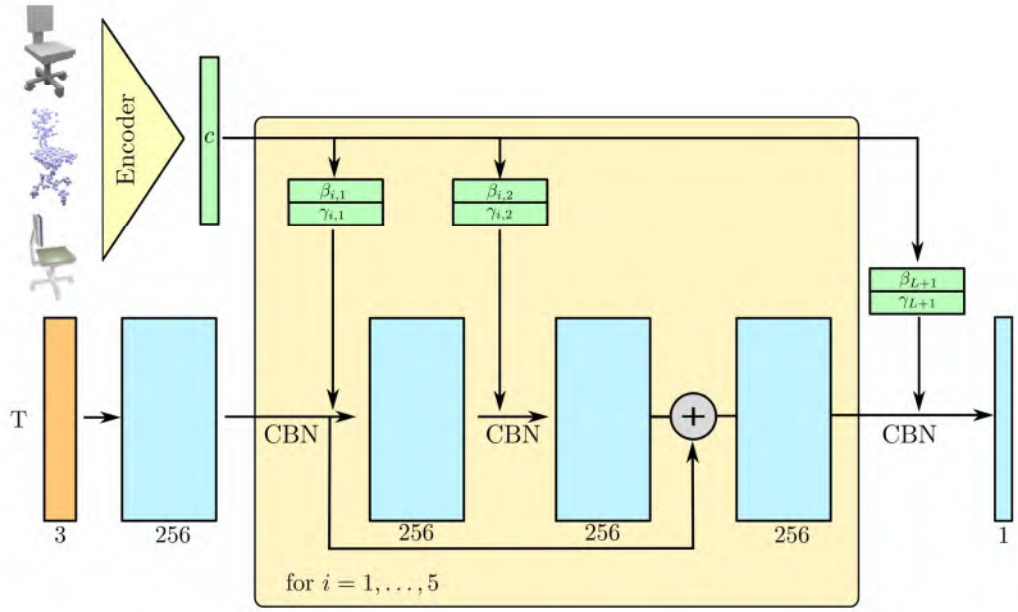


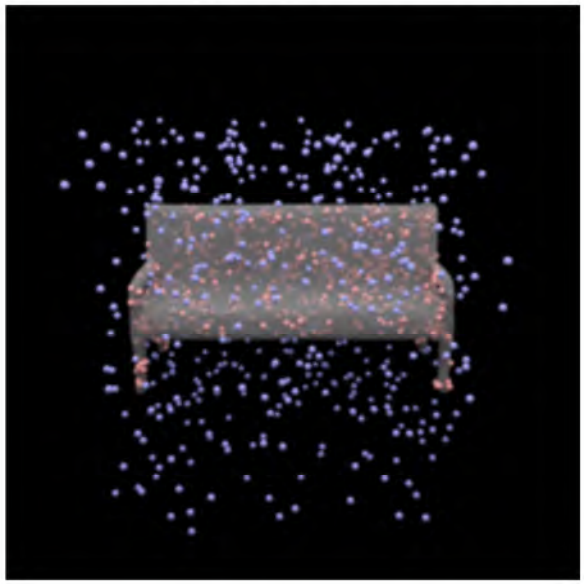
Image Credits: Learning a Predictable and Generative Vector Representation for Objects. Girdhar et. al., ECCV 2016

Predicting Implicit Volumetric Representations



Instead of decoding a volumetric representation, answer occupancy query of an **arbitrary** point

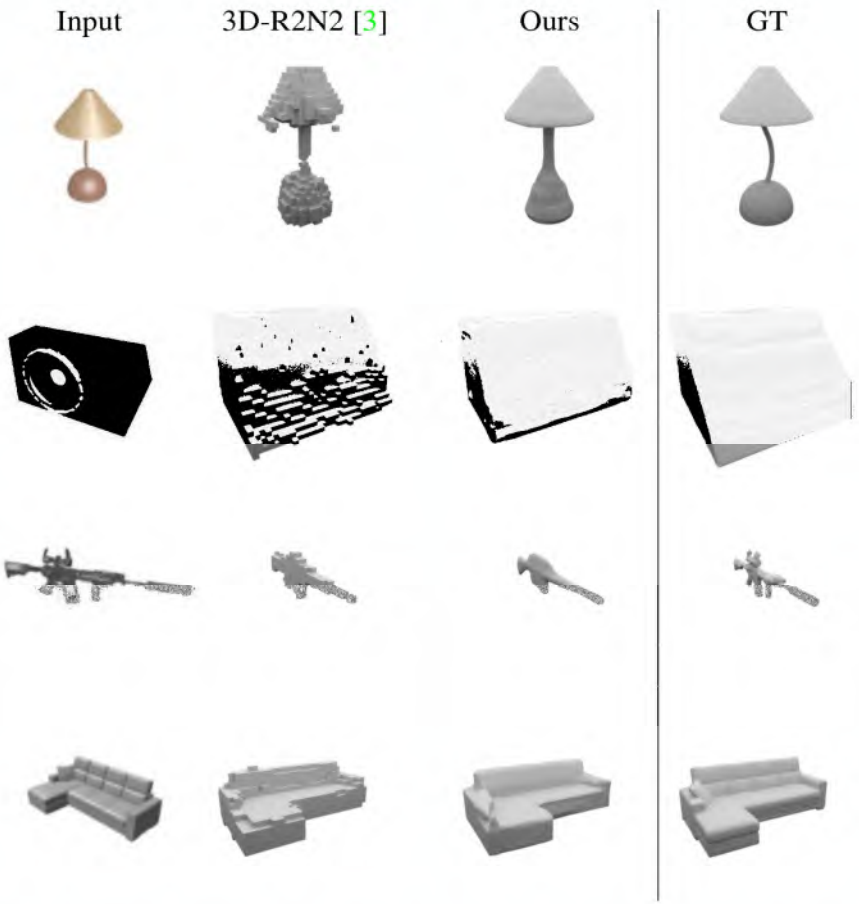
Similar learning objective as before!



extra credit question for Assignment 2!

Image Credits: Occupancy Networks: : Learning 3D Reconstruction in Function Space. Mescheder et. al.

Predicting Implicit Volumetric Representations



Can query occupancy at arbitrarily fine resolution at inference

Q: how to recover a mesh?

Image Credits: Occupancy Networks: : Learning 3D Reconstruction in Function Space. Mescheder et. al.

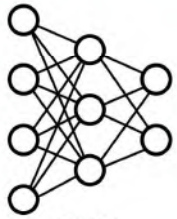
Learning to Predict Volumetric 3D

Some Takeaways

1. Simple binary-cross entropy loss between prediction and GT
2. Resolution is a bottleneck. Using implicit occupancy is an alternative
3. Similar encoders, varying decoders across approaches

Learning to Predict Point Clouds

Input Image



CNN

Predicted points
(N=1024)

Sampled GT points
(N=1024)

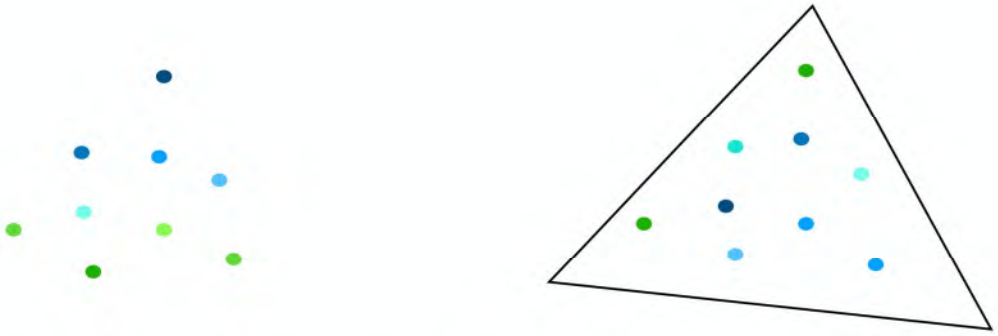
Say our CNN gives a Nx3 output. We can then sample N points from GT mesh

$$L(P, \hat{P}) = \sum_i \|P_i - \hat{P}_i\|^2$$

What is wrong with this objective?

Assumes k^{th} predicted and sampled point **correspond**

Learning to Predict Point Clouds



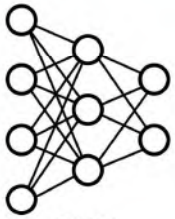
Say blue to green represents prediction/sampling order.
No reason to enforce correspondence between the light blue prediction and GT.

Need **permutation invariant** learning objectives

Assumes k^{th} predicted and sampled point **correspond**

Learning to Predict Point Clouds: Chamfer Distance

Input Image



CNN

Predicted points

Sampled GT points

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2$$

Can handle different number of points in (prediction, GT)

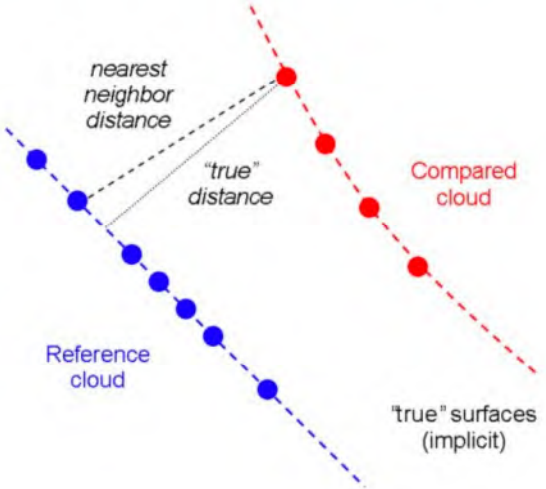
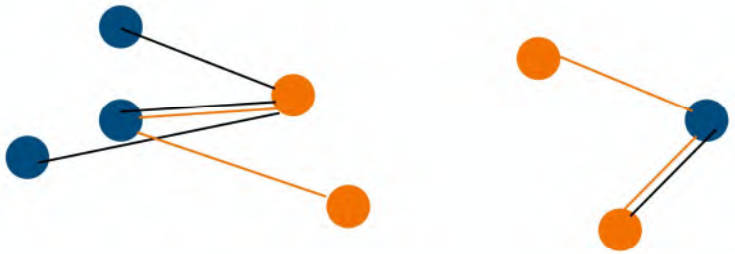


Image courtesy: cloudcompare

Chamfer Distance



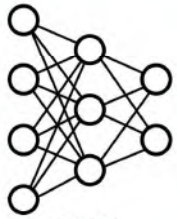
Asymmetric matching — one point (e.g. left-most orange) can be the mimima for multiple in the other set

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2$$

Can handle different number of points in (prediction, GT)

Learning to Predict Point Clouds

Input Image



CNN

Predicted points

Sampled GT points

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2$$

Image courtesy: cloudcompare

Learning to Predict Point Clouds

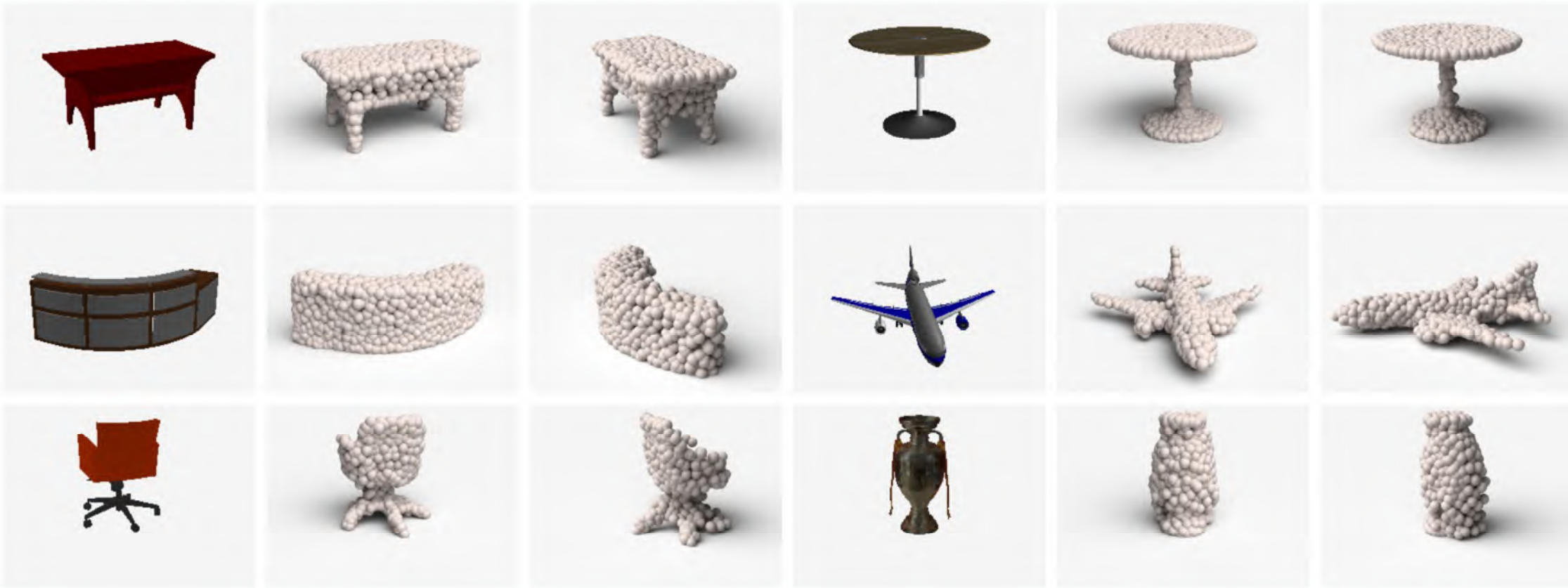
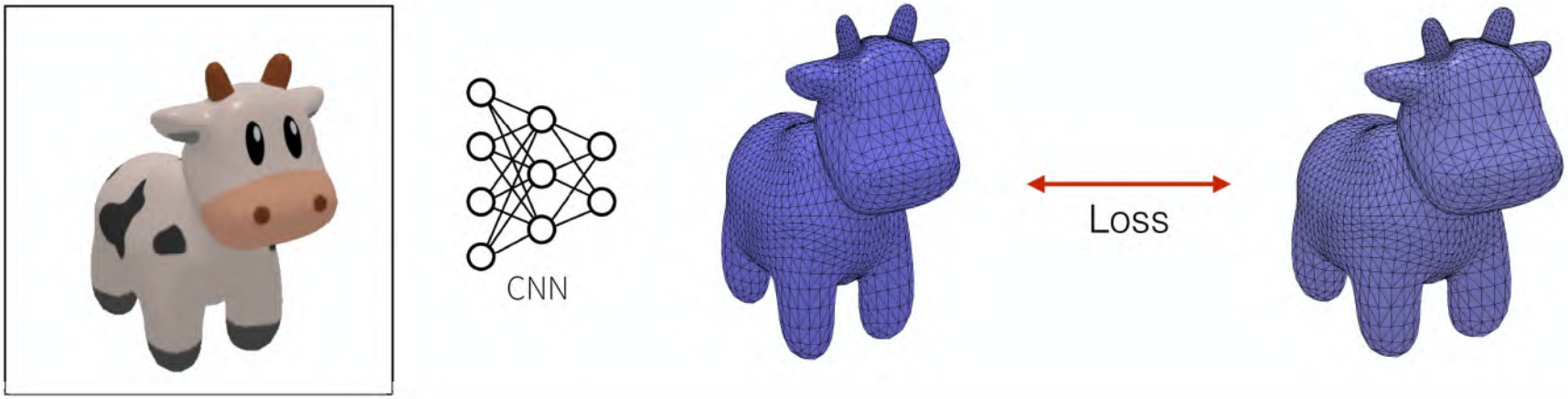


Image courtesy: A Point Set Generation Network for 3D Object Reconstruction from a Single Image. Fan et. al.

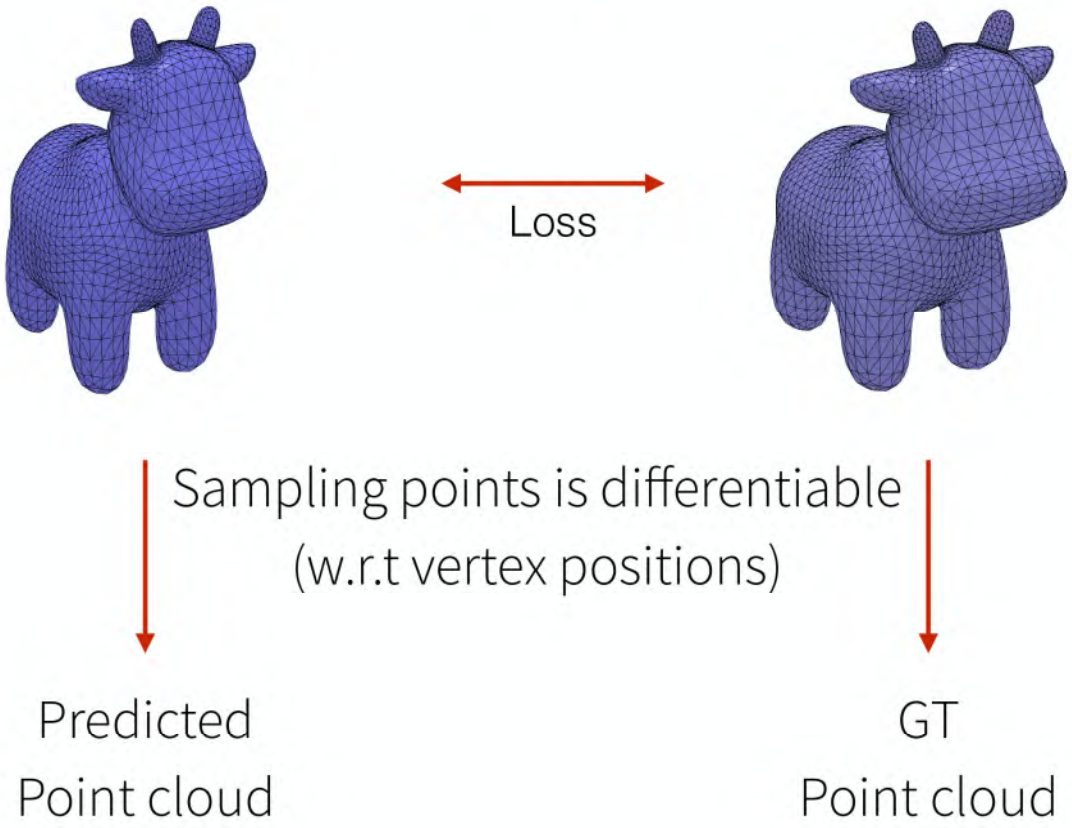
Learning to Predict Meshes



How to infer meshes (vertices and faces)?

Training objectives

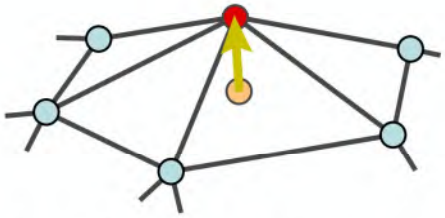
Training Objective: Data Term



Training Regularization: Smoothness

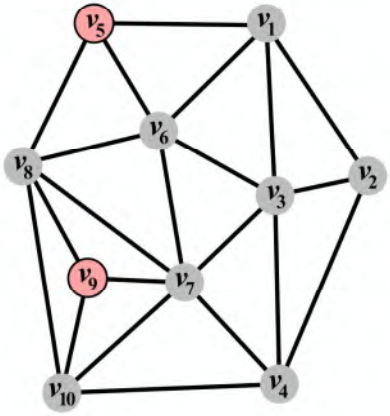
$$\|LV\|^2; \quad V : N \times 3, L : N \times N$$

Can be reformulated as minimizing squared norm of LV, where L is the laplacian matrix



$$\delta_i = \frac{1}{d_i} \sum_{j \in N(i)} (\mathbf{v}_i - \mathbf{v}_j)$$

A vertex should move towards average of its neighbors



The mesh

4	−1	−1		−1	−1				
−1	3	−1	−1						
−1	−1	5	−1		−1	−1			
	−1	−1	4			−1			−1
−1				3	−1		−1		
−1		−1			4	−1	−1		
		−1	−1		−1	6	−1	−1	−1
				−1	−1	−1	6	−1	−1
						−1	−1	3	−1
			−1			−1	−1	−1	4

The symmetric Laplacian L

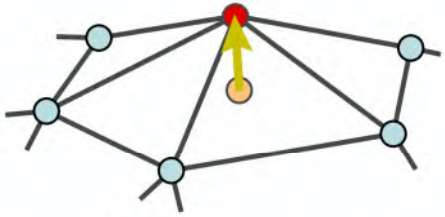
(divide each row by degree of vertex to match our definition)

'Uniform' laplacian assigns equal weight to each edge

(but other non-uniform weighting schemes exist)

Image courtesy: Laplacian Mesh Processing. Olga Sorkine

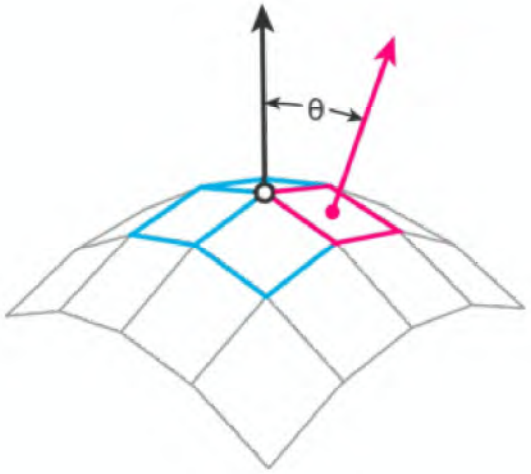
Training Regularization: Smoothness



$$\delta_i = \frac{1}{d_i} \sum_{j \in N(i)} (\mathbf{v}_i - \mathbf{v}_j)$$

$$\|LV\|^2; \quad V : N \times 3, L : N \times N$$

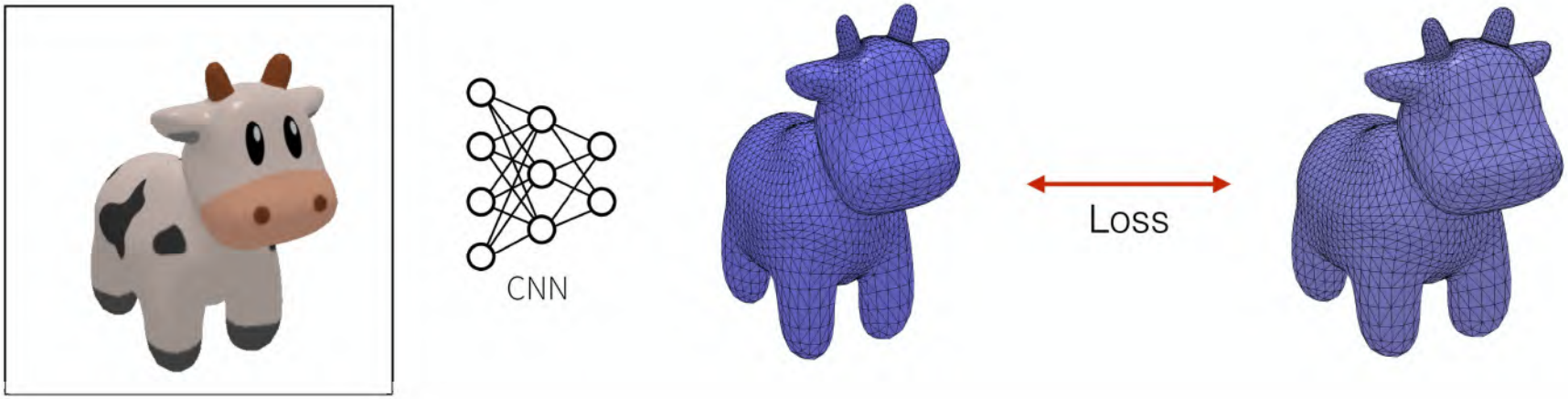
A vertex should move towards average of its neighbors



Surface normals of adjacent faces should align

Image courtesy: Stepan Jirka.

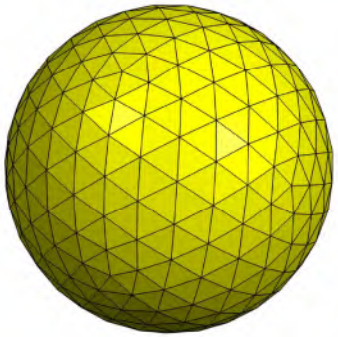
Learning to Predict Meshes

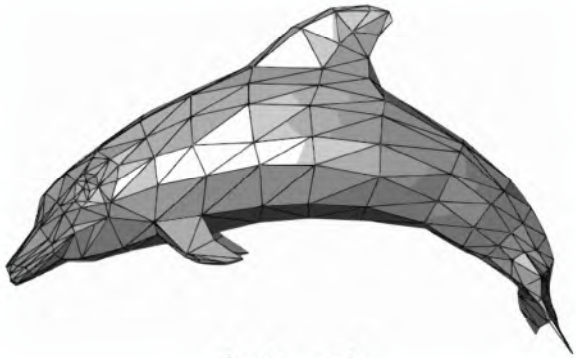


How to infer meshes (vertices and faces)?

Training objectives

Learning to ~~Predict~~ Meshes
Deform





(V , F)

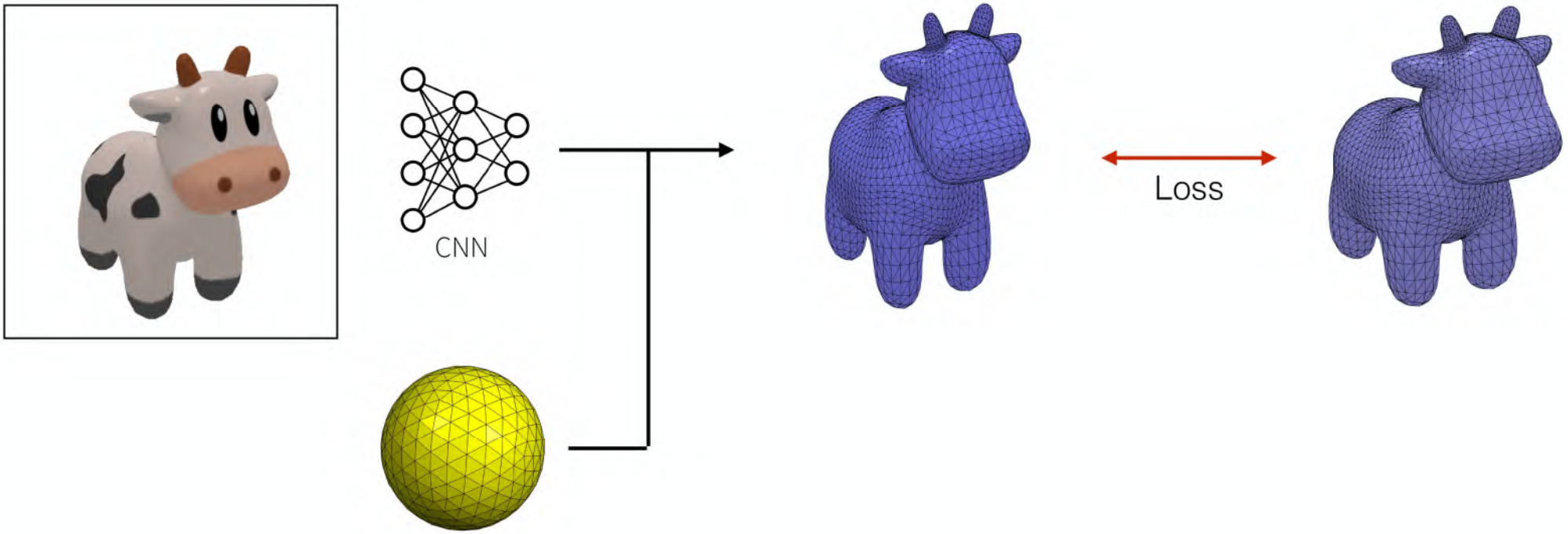
(V' , F)

Start with a known mesh

Predict new vertex positions (V x 3 dimensional)

Cannot predict/change connectivity

Learning to ~~Predict~~ Meshes
Deform

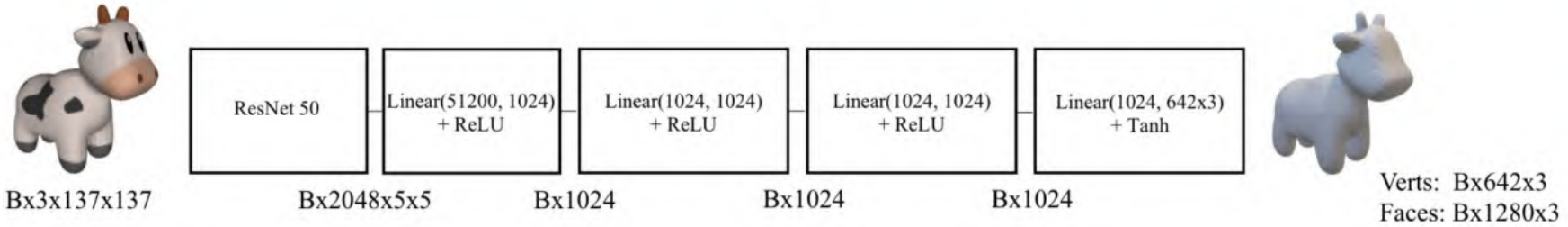


Q: What makes this different from point cloud prediction?

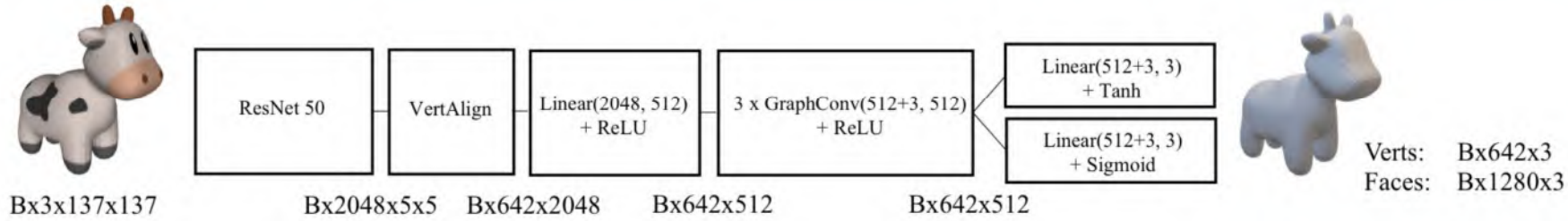
Smoothness objectives respect connectivity

Can sample varying number of points

Learning to Predict Meshes: Sample Architectures



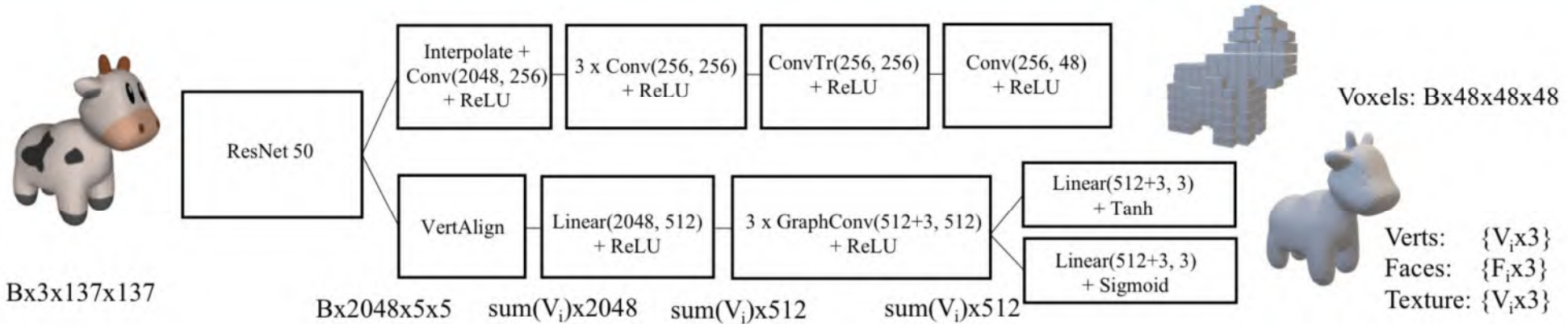
Simple feed-forward network



Including graph convolution layers that leverage connectivity

Image courtesy: Accelerating 3D Deep Learning with Pytorch3d. Ravi et. al.

Learning to Predict Meshes: Sample Architectures



First predict a volumetric 3D representation

Deform from a ‘meshified’ voxel representation

Image courtesy: Accelerating 3D Deep Learning with Pytorch3d. Ravi et. al.

Learning to Predict Meshes

Some Takeaways

1. Data terms in objectives are similar to point cloud prediction
2. Architectures and objectives can use connectivity
3. Difficult to predict connectivity (often assumed fixed)

Course Summary

Single Recon needs Priors

Optimization based methods needs predefined structure

So suitable for objects with well-defined structure (faces/bodies)

Deep learning based needs large-scale 3D data (difficult to obtain)

Different architecture for different representation

中国科学技术大学

University of Science and Technology of China

谢谢观看!