Superquadrics Revisited: Learning 3D Shape Parsing beyond Cuboids

Despoina Paschalidou[1,4] Ali Osman Ulusoy[2] Andreas Geiger[1,3,4]

[1]Autonomous Vision Group, MPI for Intelligent Systems Tübingen

[2]Microsoft [3]University of Tübingen [4]Max Planck ETH Center for Learning Systems

Max Planck Institute for Intelligent Systems Autonomous Vision Group

Eberhard Karls Universität Tübingen

ETH zürich

Microsoft

http://superquadrics.com

CVPR LONG BEACH CALIFORNIA June 16-20, 2019

Motivation

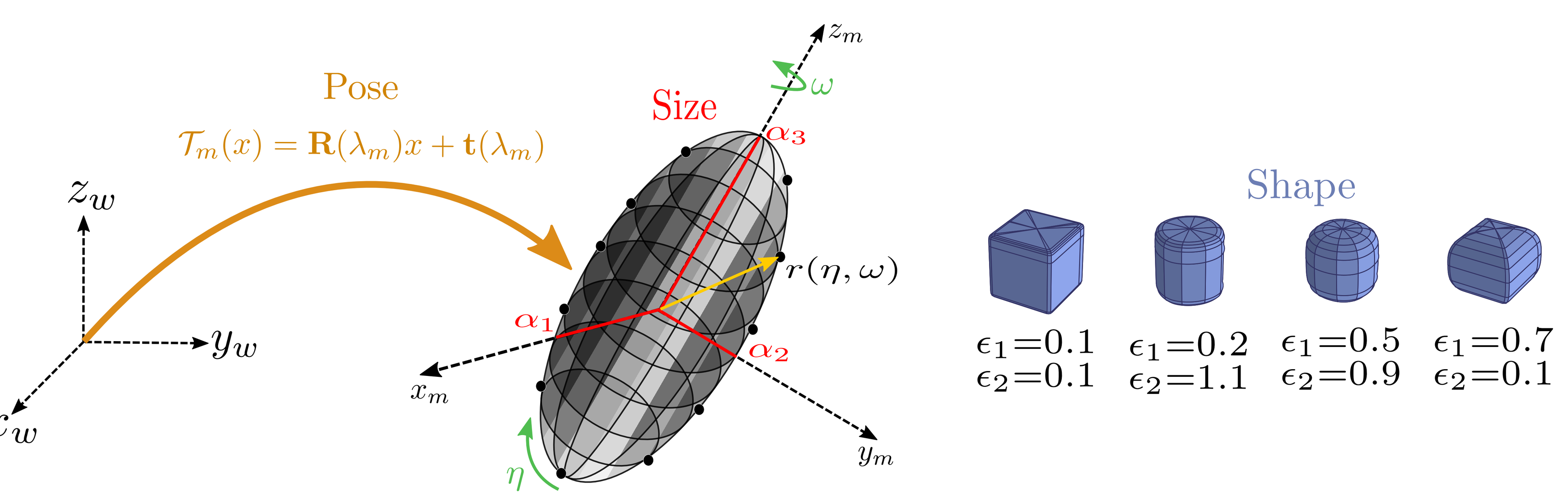
Goal: Learn to **recover the 3D shape** of an object **as a set of primitives without supervision regarding the primitive parameters**

Contributions:

- Use **superquadrics as geometric primitives** for 3D shape parsing
- An **analytical closed-form solution to the Chamfer distance** that can be evaluated in **linear time wrt. the number of primitives**

Superquadrics vs. Cuboids

Superquadrics are a parametric family of surfaces that can represent a diverse class of shapes using a **single continuous parameter space**



World Coordinates Primitive-centric Coordinates

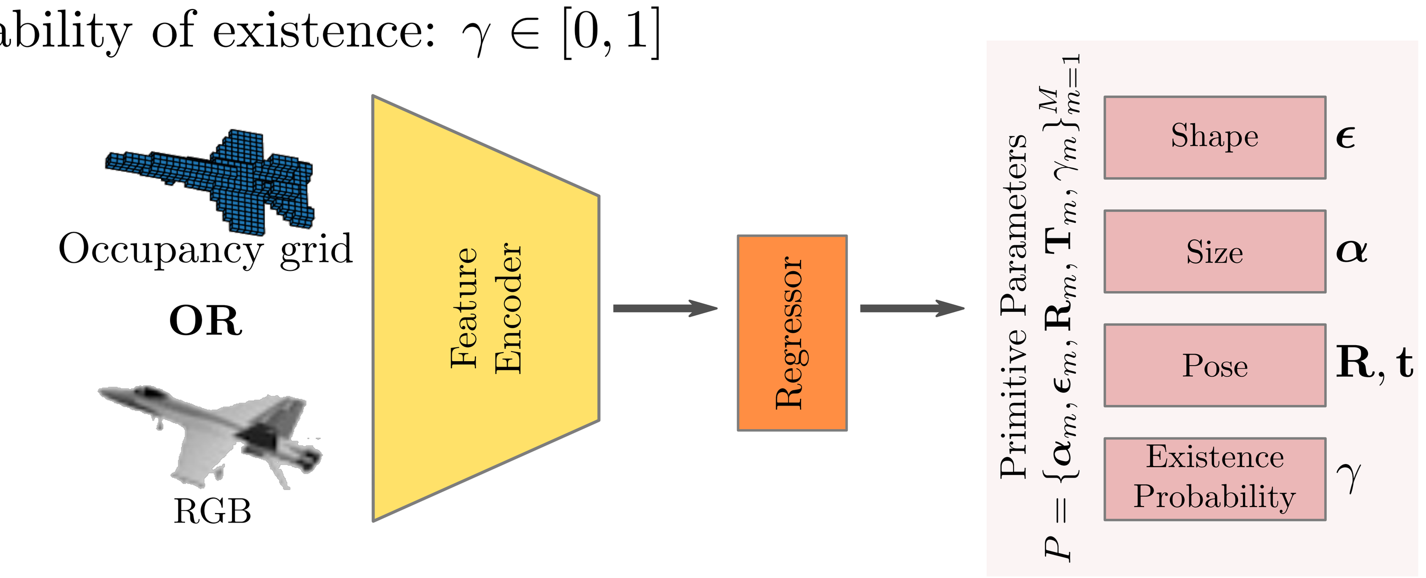
- Superquadrics are a **superset of cuboids**
- The **explicit superquadric** equation defines the surface vector **r** as:

$$\mathbf{r}(\eta,\omega) = \begin{bmatrix} \alpha_1 \cos^{\epsilon_1}\eta \cos^{\epsilon_2}\omega \\ \alpha_2 \cos^{\epsilon_1}\eta \sin^{\epsilon_2}\omega \\ \alpha_3 \sin^{\epsilon_1}\eta \end{bmatrix}$$

Network Architecture and Loss Function

The **Neural Network** encodes the input image/shape and **for each primitive** predicts:

- 11 parameters: 6 for pose $(\mathbf{R}, \mathbf{t})$, 3 for size $\boldsymbol{\alpha}$ and 2 for shape $\boldsymbol{\epsilon}$
- A probability of existence: $\gamma \in [0, 1]$

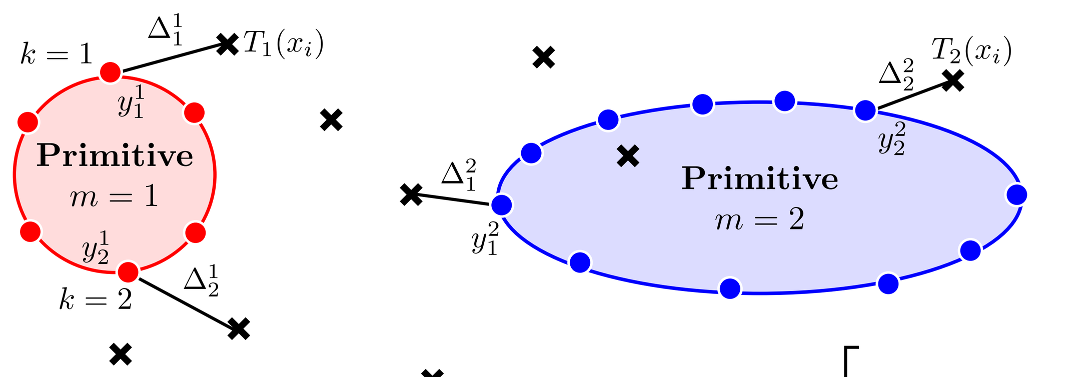


We represent the target pointcloud as a set of 3D points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ and approximate the surface of primitive m by a set of points $\mathbf{Y}_m = \{\mathbf{y}_k^m\}_{k=1}^K$

Overall Loss: Measure the discrepancy between the target and the predicted shape

$$\mathcal{L}_D(\mathbf{P}, \mathbf{X}) = \underbrace{\mathcal{L}_{P\rightarrow X}(\mathbf{P}, \mathbf{X})}_{\text{Primitive-to-Pointcloud}} + \underbrace{\mathcal{L}_{X\rightarrow P}(\mathbf{X}, \mathbf{P})}_{\text{Pointcloud-to-Primitive}} + \underbrace{\mathcal{L}_{\gamma}(\mathbf{P})}_{\text{Parsimony}}$$

Primitive-to-Pointcloud Loss



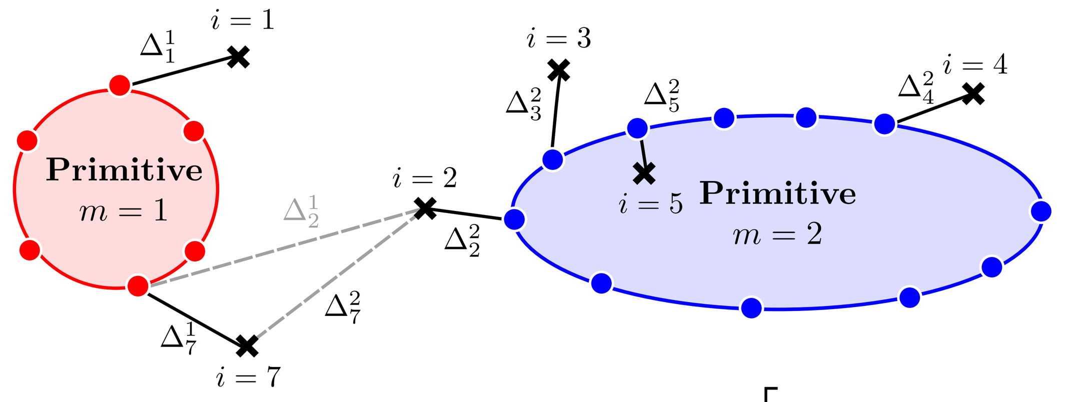
Minimum distance from point y_k^m on primitive m to the target pointcloud

$$\Delta_k^m = \min_{i=1,\dots,N} \|\mathcal{T}_m(\mathbf{x}_i) - \mathbf{y}_k^m\|_2$$

$$\mathcal{L}^m_{P\rightarrow X}(\mathbf{P}, \mathbf{X}) = \frac{1}{K}\sum_{k=1}^{K} \Delta_k^m$$

$$\mathcal{L}_{P\rightarrow X}(\mathbf{P}, \mathbf{X}) = \mathbb{E}_{p(\mathbf{z})}\left[\sum_{m|z_m=1} \mathcal{L}^m_{P\rightarrow X}(\mathbf{P}, \mathbf{X})\right] = \sum_{m=1}^{M} \gamma_m \mathcal{L}^m_{P\rightarrow X}(\mathbf{P}, \mathbf{X})$$

Pointcloud-to-Primitive Loss



Minimum distance from point x_i to the predicted shape

$$\Delta_i^m = \min_{k=1,\dots,K} \|\mathcal{T}_m(\mathbf{x}_i) - \mathbf{y}_k^m\|_2$$

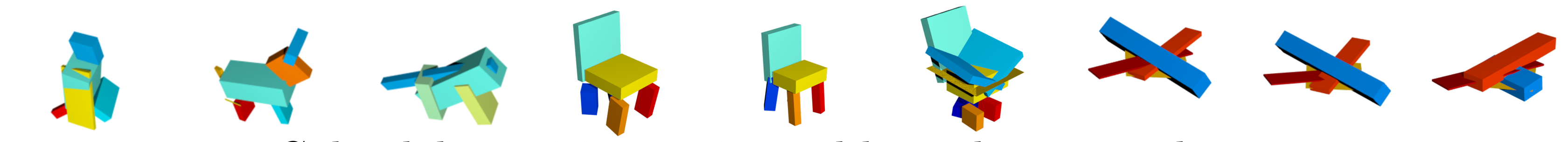
$$\mathcal{L}^i_{X\rightarrow P}(\mathbf{X}, \mathbf{P}) = \min_{m|z_m=1} \Delta_i^m$$

$$\mathcal{L}_{X\rightarrow P}(\mathbf{X}, \mathbf{P}) = \mathbb{E}_{p(\mathbf{z})}\left[\sum_{\mathbf{x}_i\in\mathbf{X}} \mathcal{L}^i_{X\rightarrow P}(\mathbf{X}, \mathbf{P})\right] = \sum_{\mathbf{x}_i\in\mathbf{X}} \sum_{m=1}^{M} \Delta_i^m \gamma_m \prod_{\bar{m}=1}^{m-1}(1-\gamma_{\bar{m}})$$

Experiments on ShapeNet

- Train **a model per-object category**, using maximally $M = 20$ primitives
- Associate every primitive with a unique color, thus **primitives illustrated with the same color correspond to the same object part**

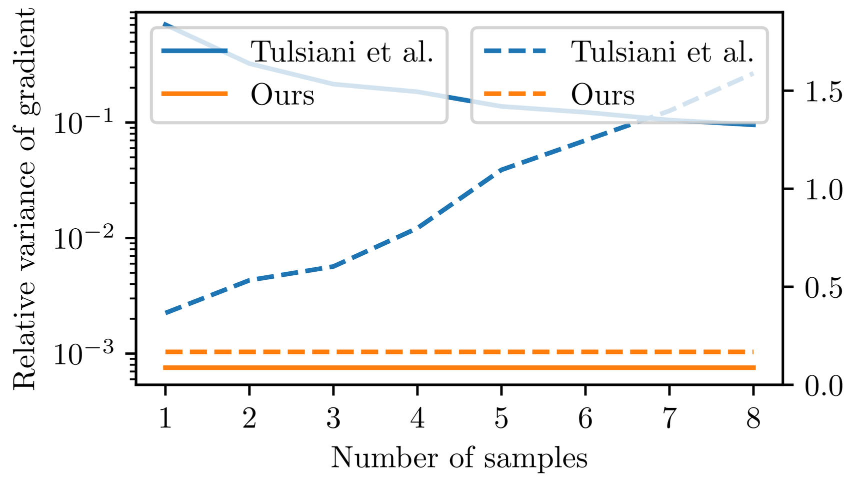
Ground-truth mesh



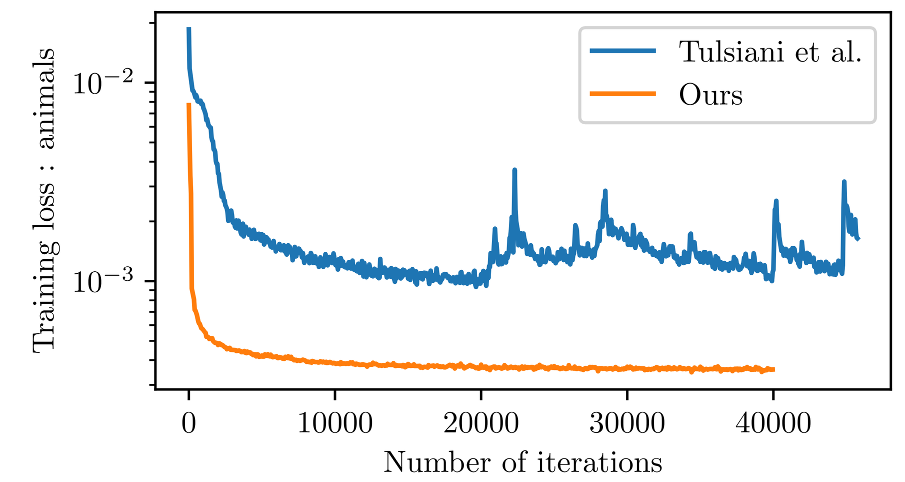
Cuboidal primitives estimated by Tulsiani et. al.

Superquadric surfaces (**Ours**)

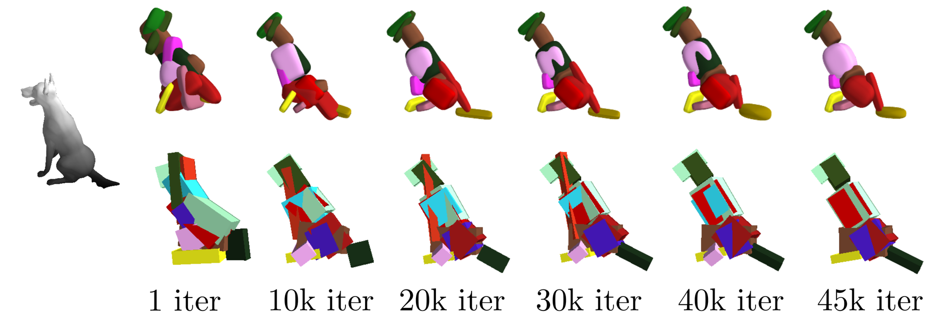
Gradient Variance and Iteration Time



Evolution of Training Loss



Training Evolution Superquadrics vs. Cuboids



Experiments on SURREAL