Generating Part-Aware Editable 3D Shapes without 3D Supervision

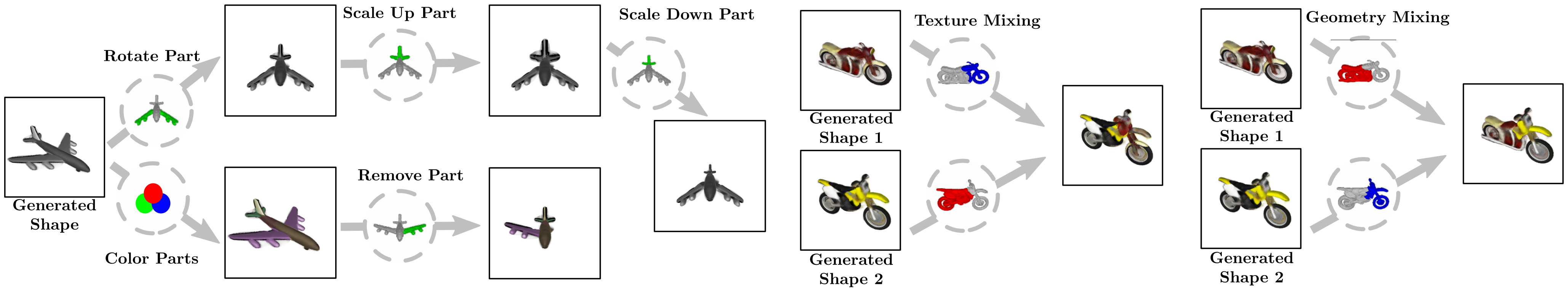
Konstantinos Tertikas [1, 3] Despoina Paschalidou [2] Boxiao Pan [2] Jeong Joon Park [2] Mikaela Angelina Uy [2]

Ioannis Emiris [3, 1] Yannis Avrithis [4] Leonidas J. Guibas [2]

[1] National and Kapodistrian University of Athens [2] Stanford University [3] ATHENA Research Center
[4] Institute of Advanced Research in Artificial Intelligence (IARAI)

CVPR, 2023

TUE-PM-032



Why do we need parts for 3D object generation?

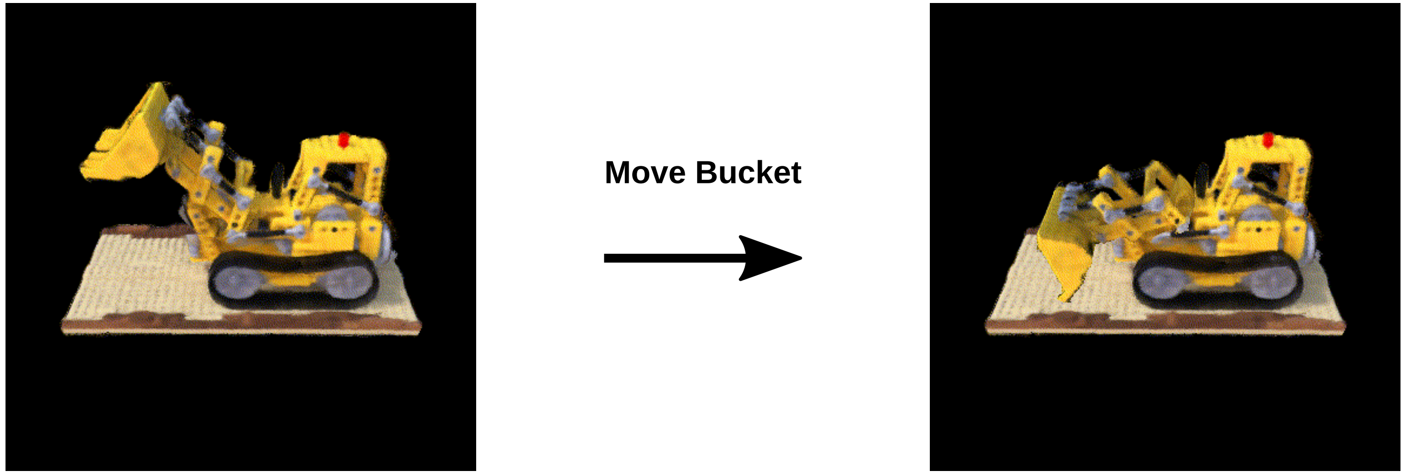
Why do we need parts for 3D object generation?

Shape editing involves **making local changes to the shape and appearance of different regions** of an object. A user may want to:

Why do we need parts for 3D object generation?

Shape editing involves **making local changes to the shape and appearance of different regions** of an object. A user may want to:

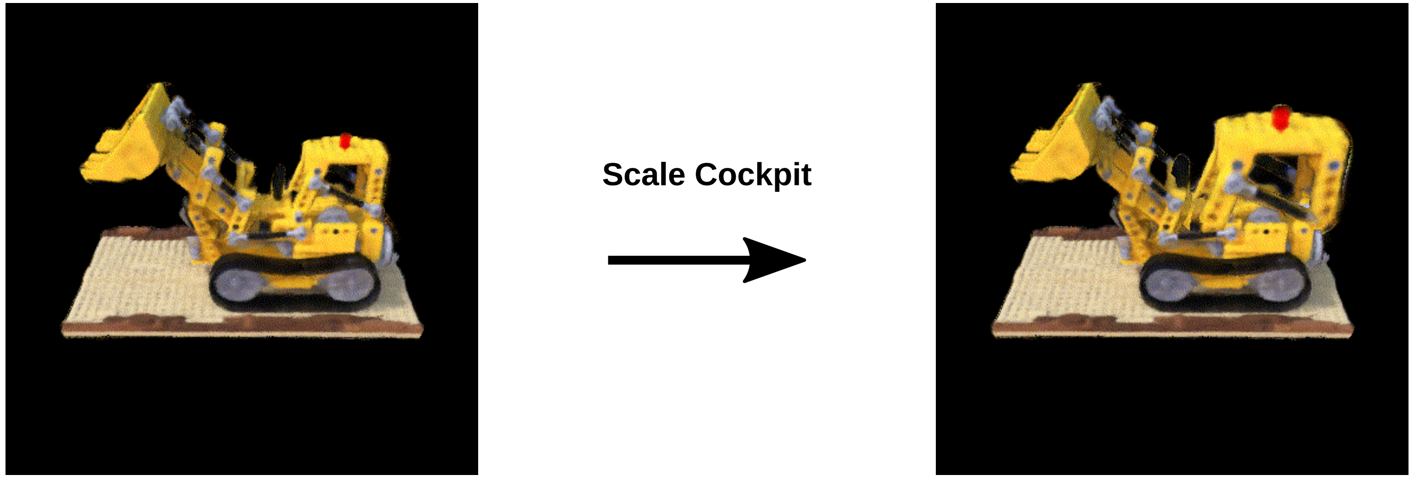
- **Apply rigid & non-rigid transformations** on specific areas of the object.



Why do we need parts for 3D object generation?

Shape editing involves **making local changes to the shape and appearance of different regions** of an object. A user may want to:

- **Apply rigid & non-rigid transformations** on specific areas of the object.



Why do we need parts for 3D object generation?

Shape editing involves **making local changes to the shape and appearance of different regions** of an object. A user may want to:

- **Apply rigid & non-rigid transformations** on specific areas of the object.
- **Change the appearance** of an object part.

Color Bucket

Why do we need parts for 3D object generation?

Shape editing involves **making local changes to the shape and appearance of different regions** of an object. A user may want to:

- **Apply rigid & non-rigid transformations** on specific areas of the object.
- **Change the appearance** of an object part.
- **Combine parts from different objects**.

Why do we need parts for 3D object generation?

Shape editing involves **making local changes to the shape and appearance of different regions** of an object. A user may want to:

- **Apply rigid & non-rigid transformations** on specific areas of the object.
- **Change the appearance** of an object part.
- **Combine parts from different objects**.

Why do we need parts for 3D object generation?

Shape editing involves **making local changes to the shape and appearance of different regions** of an object. A user may want to:

- **Apply rigid & non-rigid transformations** on specific areas of the object.
- **Change the appearance** of an object part.
- **Combine parts from different objects**.

Why do we need parts for 3D object generation?

Shape editing involves **making local changes to the shape and appearance of different regions** of an object. A user may want to:

- **Apply rigid & non-rigid transformations** on specific areas of the object.
- **Change the appearance** of an object part.
- **Combine parts from different objects**.

Why do we need parts for 3D object generation?

Shape editing involves **making local changes to the shape and appearance of different regions** of an object. A user may want to:

- **Apply rigid & non-rigid transformations** on specific areas of the object.
- **Change the appearance** of an object part.
- **Combine parts from different objects**.

Parts enable **local control**!

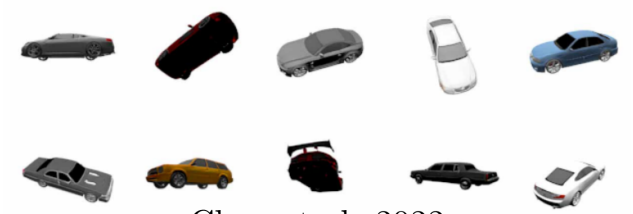
Existing Methods

Existing Methods

NeRF-based Generative Models

Schwarz et al. 2020

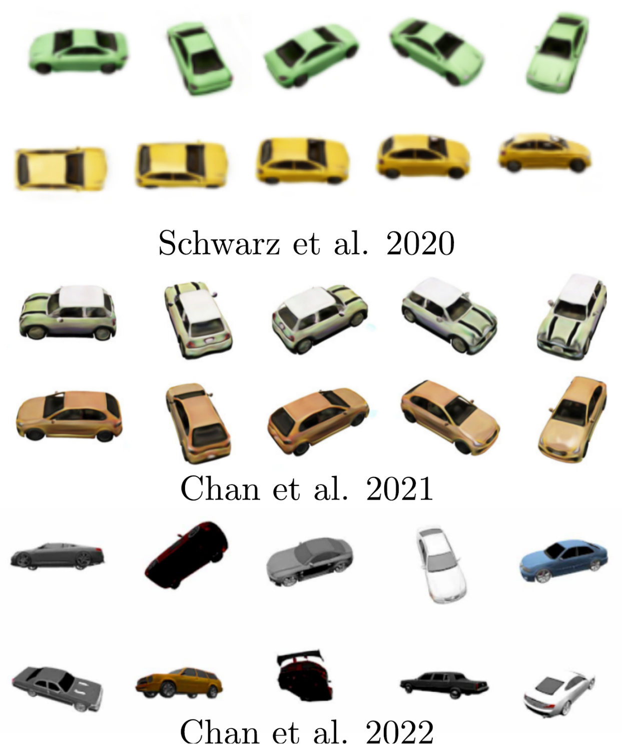
Chan et al. 2021



Chan et al. 2022

Existing Methods

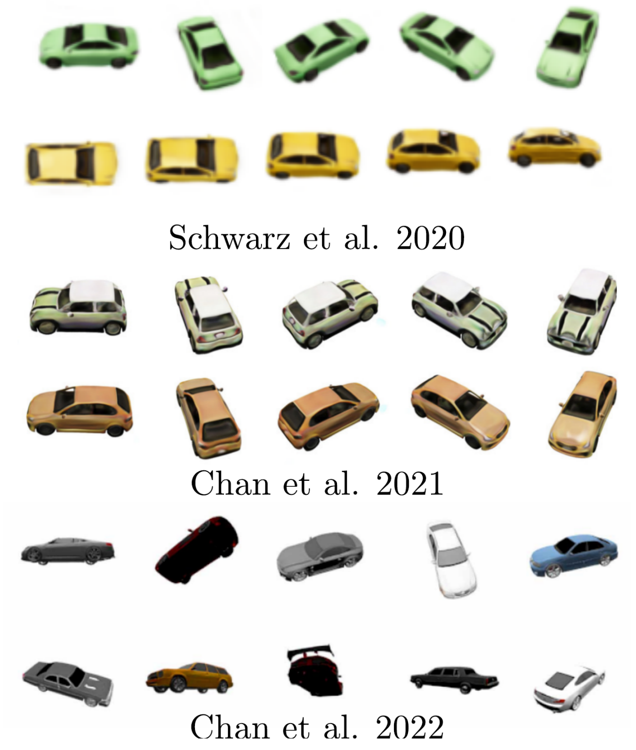
NeRF-based Generative Models



- ✓ High quality 3D meshes
  - ✓ 2D Supervision

Existing Methods

NeRF-based Generative Models



✓ High quality 3D meshes

✓ 2D Supervision

✗ No explicit part-level control

Existing Methods

NeRF-based Generative Models

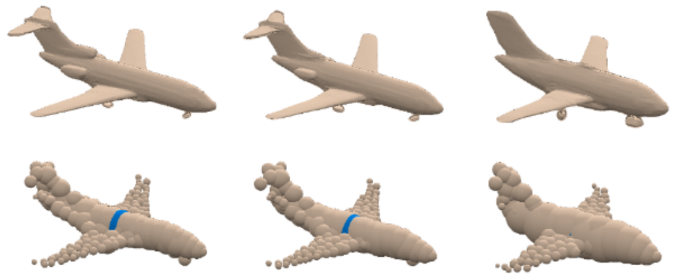
Schwarz et al. 2020

Chan et al. 2021

Chan et al. 2022

- ✓ High quality 3D meshes
- ✓ 2D Supervision
- ✗ No explicit part-level control

Part-based Generative Models



Hao et al. 2020

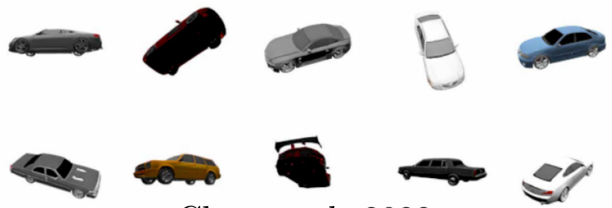
Hertz et al. 2022

Existing Methods

NeRF-based Generative Models

Schwarz et al. 2020

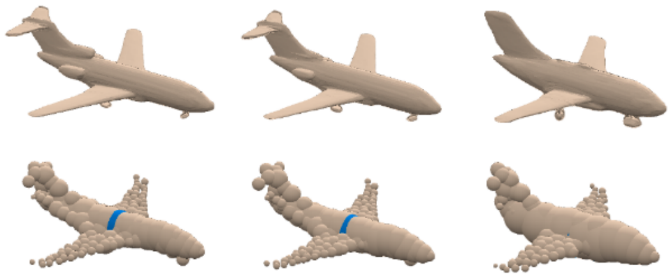
Chan et al. 2021



Chan et al. 2022

- ✓ High quality 3D meshes
- ✓ 2D Supervision
- ✗ No explicit part-level control

Part-based Generative Models



Hao et al. 2020

Hertz et al. 2022

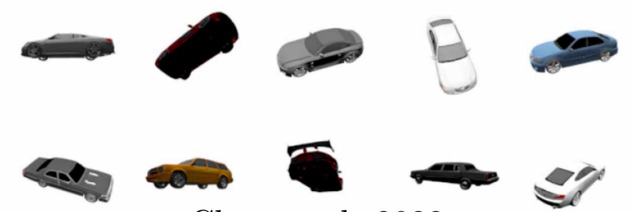
- ✓ Explicit part-level control

Existing Methods

NeRF-based Generative Models

Schwarz et al. 2020

Chan et al. 2021



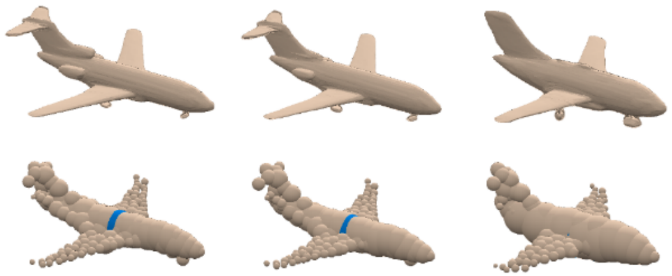
Chan et al. 2022

✓ High quality 3D meshes

✓ 2D Supervision

✗ No explicit part-level control

Part-based Generative Models



Hao et al. 2020

Hertz et al. 2022

✓ Explicit part-level control

✗ 3D Supervision

✗ No texture information

PartNeRF

PartNeRF

- 3D **part-aware** generative model

PartNeRF

- 3D **part-aware** generative model
- Can perform **local shape & appearance edits**

PartNeRF

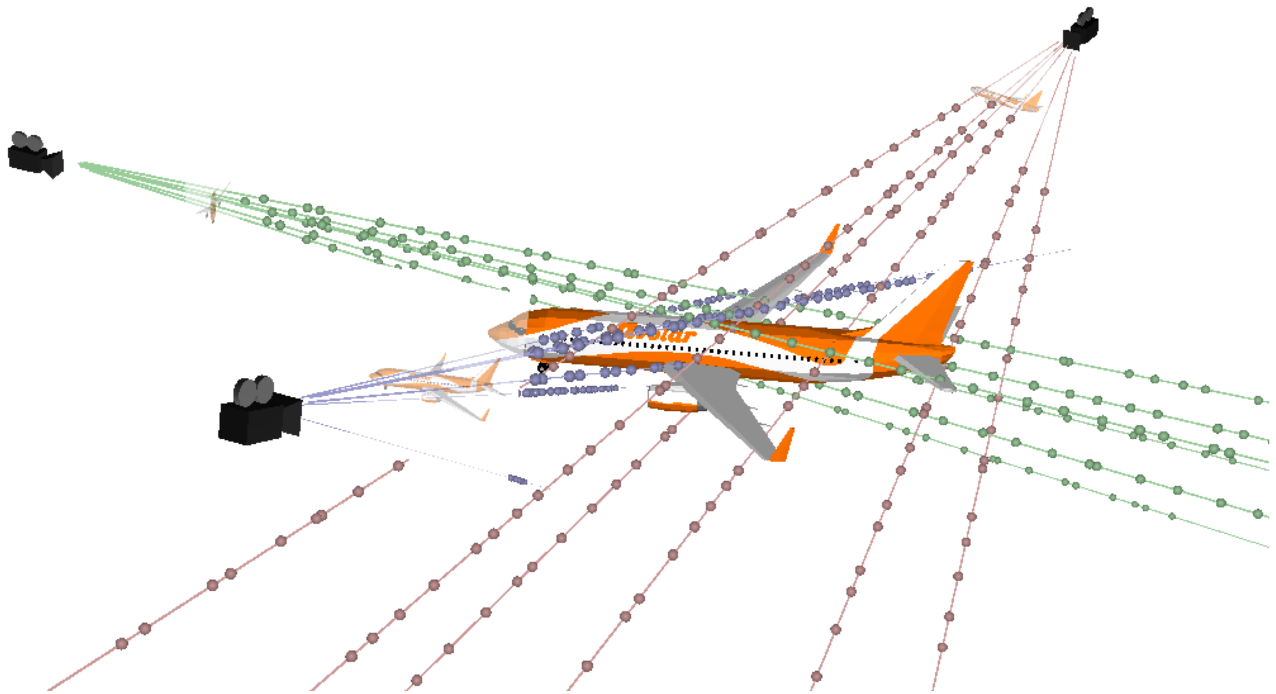
- 3D **part-aware** generative model
- Can perform **local shape & appearance edits**
- Trained only using **posed images & object masks**

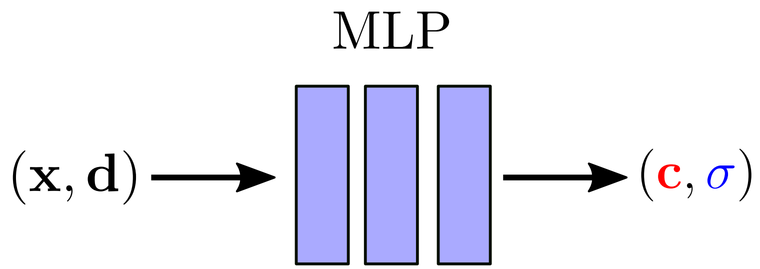
PartNeRF

- 3D **part-aware** generative model
- Can perform **local shape & appearance edits**
- Trained only using **posed images & object masks**

Neural Radiance Fields

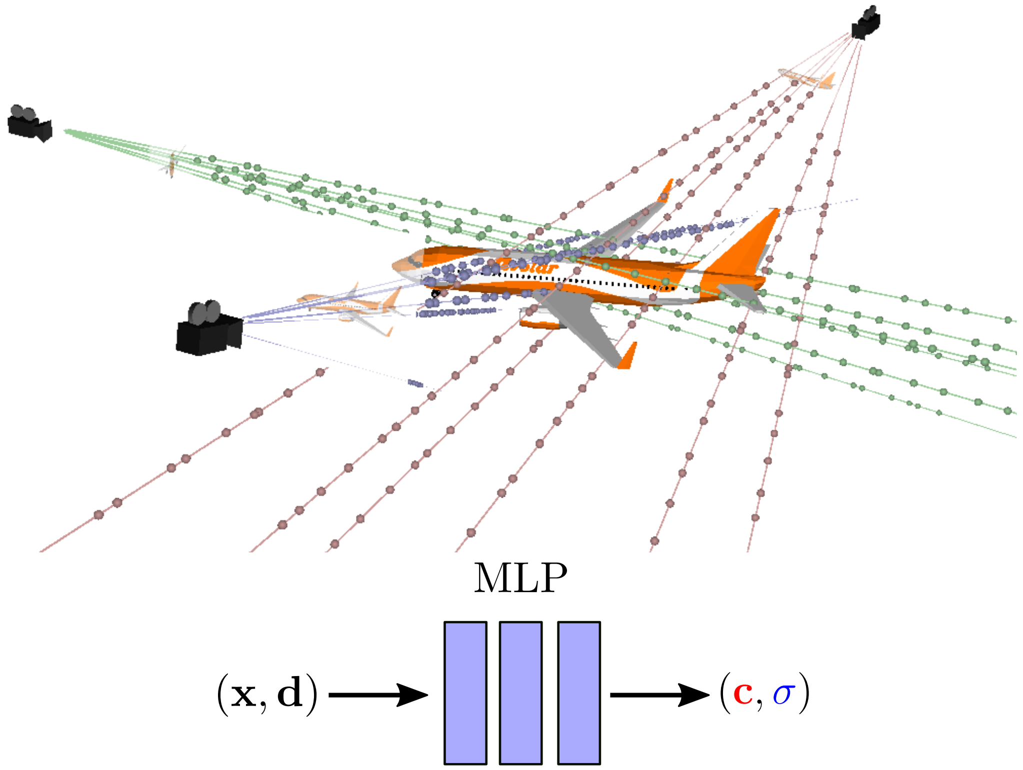
NeRFs map a *3D point* $\mathbf{x} \in \mathbb{R}^3$ and a *viewing direction* $\mathbf{d} \in \mathbb{S}^2$ to a *color* $\mathbf{c} \in \mathbb{R}^3$ and a *volume density* $\sigma \in \mathbb{R}^+$.





Neural Radiance Fields

NeRFs map a *3D point* $\mathbf{x} \in \mathbb{R}^3$ and a *viewing direction* $\mathbf{d} \in \mathbb{S}^2$ to a *color* $\mathbf{c} \in \mathbb{R}^3$ and a *volume density* $\boldsymbol{\sigma} \in \mathbb{R}^+$.

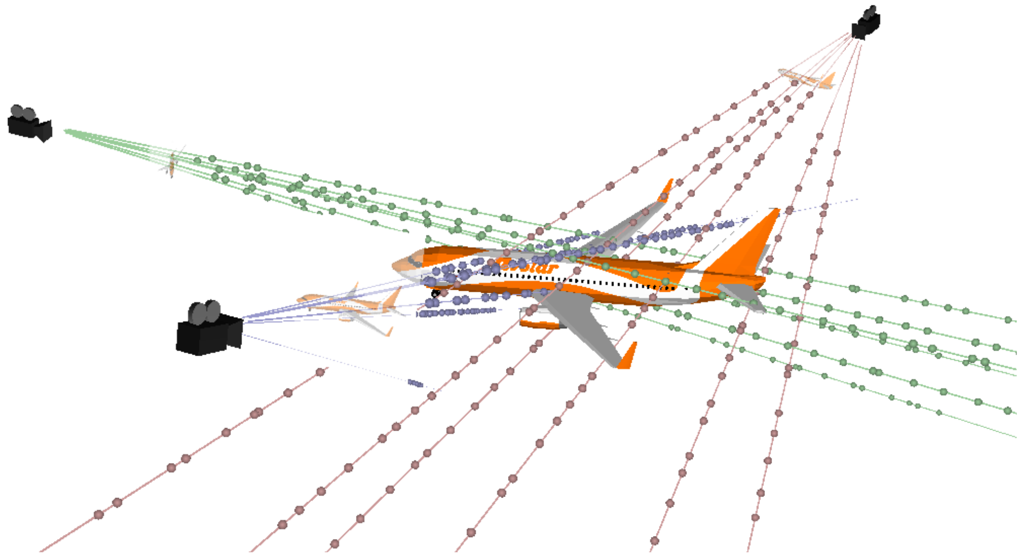


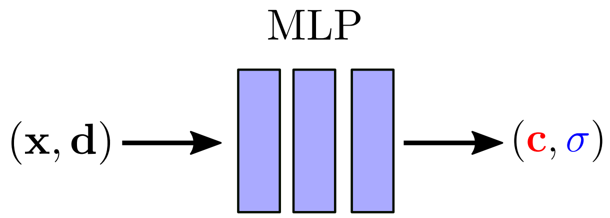
Neural Radiance Fields

NeRFs map a *3D point* $\mathbf{x} \in \mathbb{R}^3$ and a *viewing direction* $\mathbf{d} \in \mathbb{S}^2$ to a *color* $\mathbf{c} \in \mathbb{R}^3$ and a *volume density* $\boldsymbol{\sigma} \in \mathbb{R}^+$.

The **rendered color** for a ray r is calculated by accumulating the predicted $\{\mathbf{c}_i^r, \boldsymbol{\sigma}_i^r\}^N$ for N sampled points $\mathcal{X}_r = \{\mathbf{x}_i^r\}_{i=1}^N$ along r:

$$\hat{C}(r) = \sum_{i=1}^{N} \exp\Big(-\sum_{i<j} \boldsymbol{\sigma}_j^r \delta_j^r\Big)(1 - \exp(-\boldsymbol{\sigma}_i^r \delta_i^r))\mathbf{c}_i^r$$





Neural Radiance Fields

NeRFs map a *3D point* $\mathbf{x} \in \mathbb{R}^3$ and a *viewing direction* $\mathbf{d} \in \mathbb{S}^2$ to a *color* $\mathbf{c} \in \mathbb{R}^3$ and a *volume density* $\boldsymbol{\sigma} \in \mathbb{R}^+$.

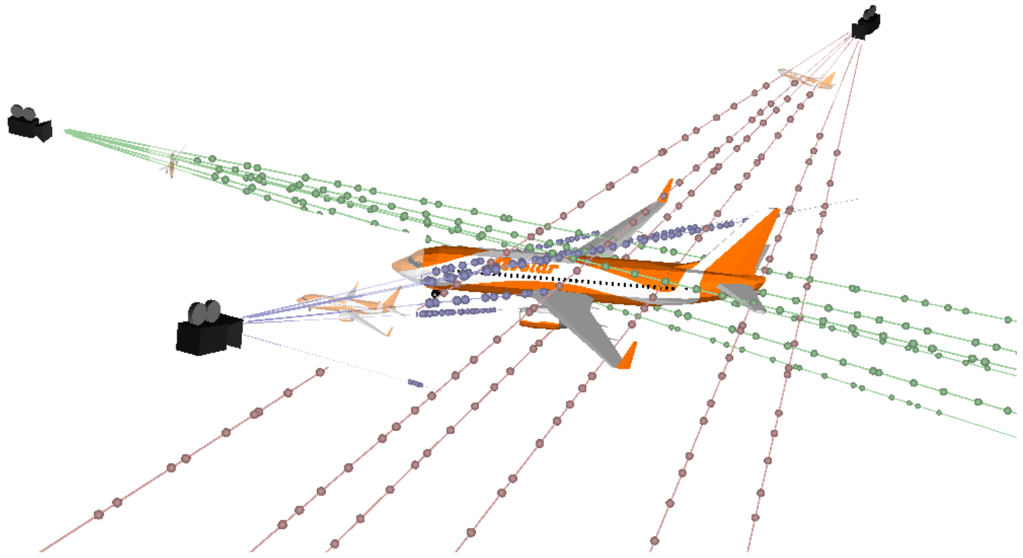
The **rendered color** for a ray r is calculated by accumulating the predicted $\{\mathbf{c}_i^r, \boldsymbol{\sigma}_i^r\}^N$ for N sampled points $\mathcal{X}_r = \{\mathbf{x}_i^r\}_{i=1}^N$ along r:

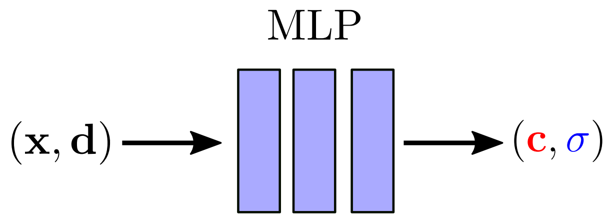
$$\hat{C}(r) = \sum_{i=1}^{N} \exp\left(-\sum_{i<j} \sigma_j^r \delta_j^r\right)\left(1 - \exp(-\sigma_i^r \delta_i^r)\right)\mathbf{c}_i^r$$

or equally

$$\hat{C}(r) = \sum_{i=1}^{N} o_i^r \prod_{j<i} \left(1 - o_j^r\right) \mathbf{c}_i^r$$

with $o_i^r = 1 - \exp(-\sigma_i^r \delta_i^r)$ the **occupancy value** at point $\mathbf{x}_i^r$ and δ_i^r the distance between two adjacent ray points.





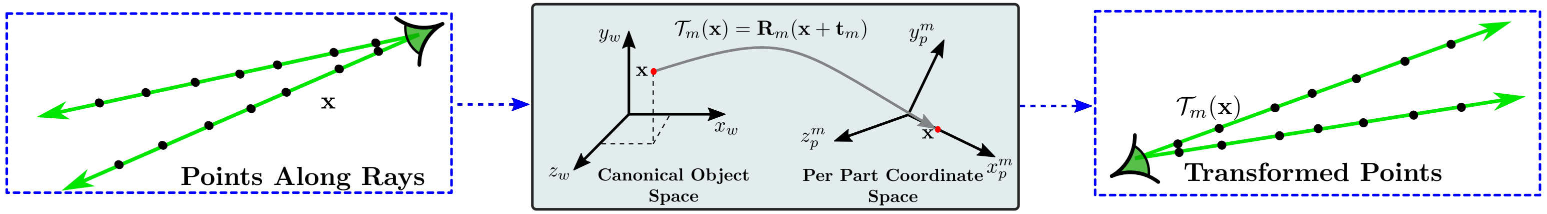
Parts as Neural Radiance Fields

Parts as Neural Radiance Fields

- PartNeRF decomposes a 3D object into M parts, each **parametrized as a NeRF**!

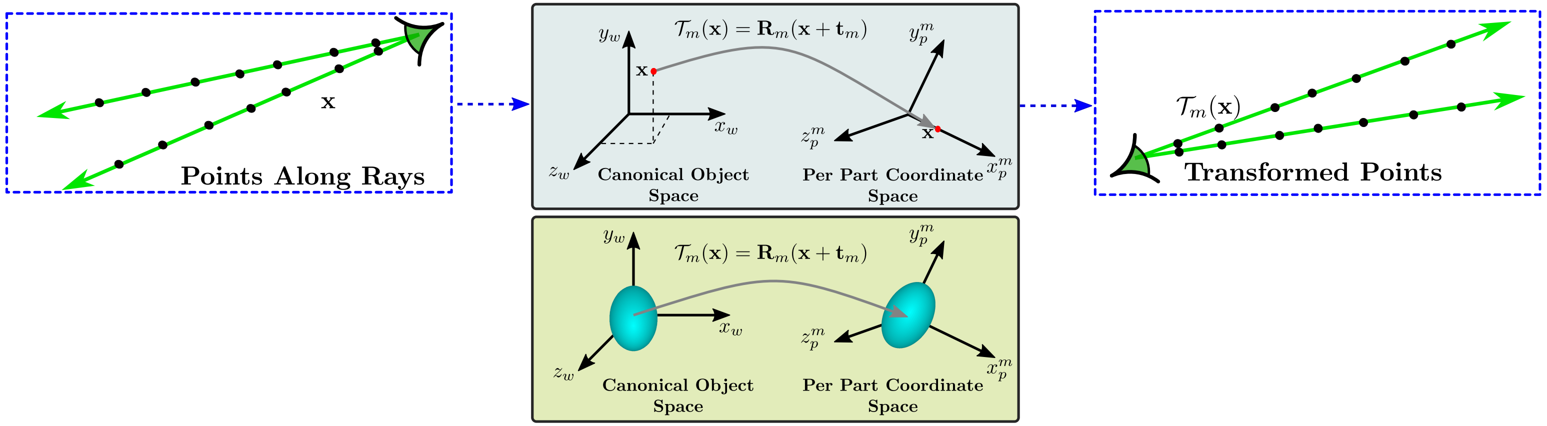
Parts as Neural Radiance Fields

- PartNeRF decomposes a 3D object into *M* parts, each **parametrized as a NeRF**!
- Each part *m* consists of:
    1. An *affine transformation* $\mathcal{T}_m(\mathbf{x}) = \mathbf{R}_m(\mathbf{x} - \mathbf{t}_m)$ mapping a point $\mathbf{x}$ to the *local coordinate system of the part*, where $\mathbf{t}_m \in \mathbb{R}^3$ a translation vector and $\mathbf{R}_m \in \mathrm{SO}(3)$ a rotation matrix.



Parts as Neural Radiance Fields

- PartNeRF decomposes a 3D object into *M* parts, each **parametrized as a NeRF**!
- Each part *m* consists of:
    1. An *affine transformation* $\mathcal{T}_m(\mathbf{x}) = \mathbf{R}_m(\mathbf{x} - \mathbf{t}_m)$ mapping a point $\mathbf{x}$ to the *local coordinate system of the part*, where $\mathbf{t}_m \in \mathbb{R}^3$ a translation vector and $\mathbf{R}_m \in \mathrm{SO}(3)$ a rotation matrix.
    2. A scale vector $\mathbf{s}_m \in \mathbb{R}^3$.



Parts as Neural Radiance Fields

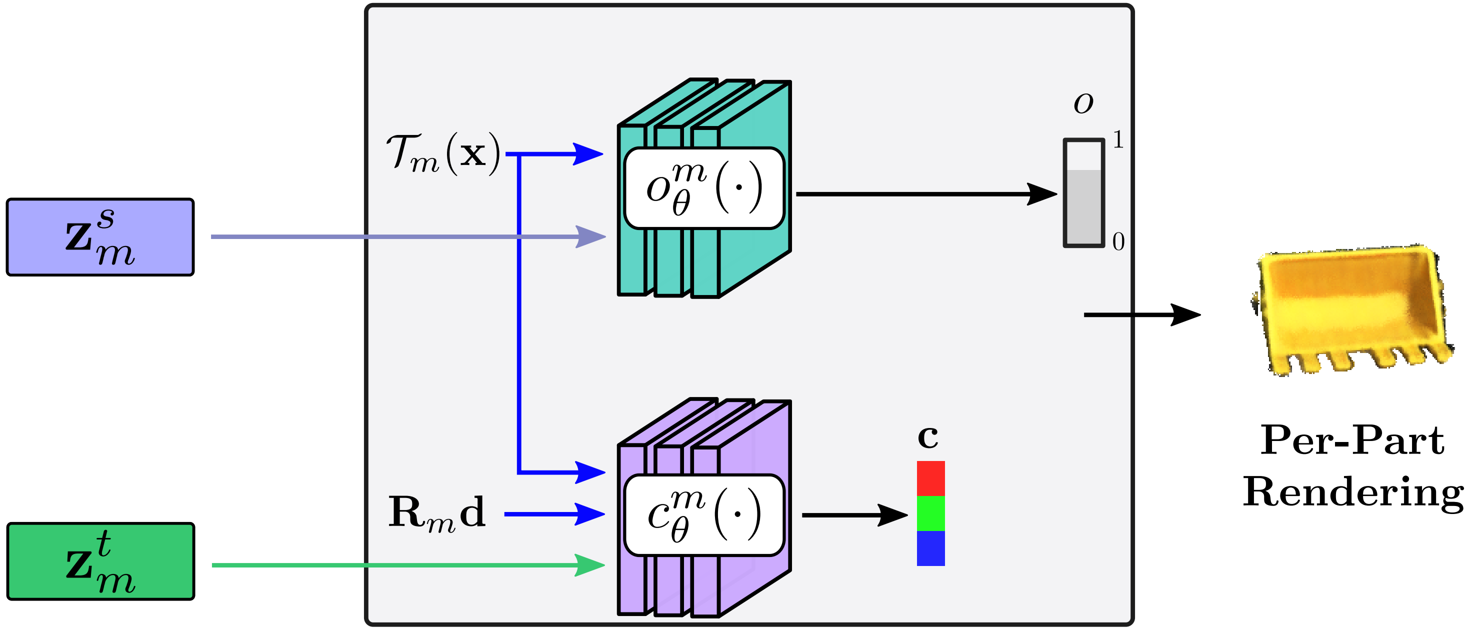
- PartNeRF decomposes a 3D object into *M* parts, each **parametrized as a NeRF**!
- Each part *m* consists of:
    1. An *affine transformation* $\mathcal{T}_m(\mathbf{x}) = \mathbf{R}_m(\mathbf{x} - \mathbf{t}_m)$ mapping a point $\mathbf{x}$ to the *local coordinate system of the part*, where $\mathbf{t}_m \in \mathbb{R}^3$ a translation vector and $\mathbf{R}_m \in \mathrm{SO}(3)$ a rotation matrix.
    2. A scale vector $\mathbf{s}_m \in \mathbb{R}^3$.
    3. Two latent codes: **shape** $\mathbf{z}_m^s \in \mathbb{R}^{L_s}$ and **texture** $\mathbf{z}_m^t \in \mathbb{R}^{L_t}$.

$\mathbf{z}_m^s$

$\mathbf{z}_m^t$

Parts as Neural Radiance Fields

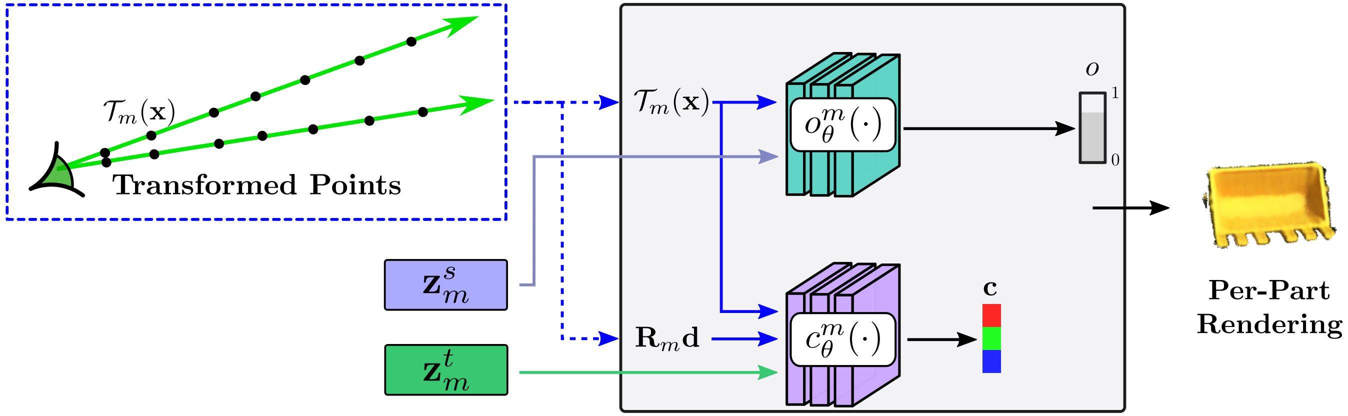
- PartNeRF decomposes a 3D object into *M* parts, each **parametrized as a NeRF**!
- Each part *m* consists of:
    1. An *affine transformation* $\mathcal{T}_m(\mathbf{x}) = \mathbf{R}_m(\mathbf{x} - \mathbf{t}_m)$ mapping a point $\mathbf{x}$ to the *local coordinate system of the part*, where $\mathbf{t}_m \in \mathbb{R}^3$ a translation vector and $\mathbf{R}_m \in \mathrm{SO}(3)$ a rotation matrix.
    2. A scale vector $\mathbf{s}_m \in \mathbb{R}^3$.
    3. Two latent codes: **shape** $\mathbf{z}_m^s \in \mathbb{R}^{L_s}$ and **texture** $\mathbf{z}_m^t \in \mathbb{R}^{L_t}$.
    4. Two separate networks, a **color network** c_θ^m and an **occupancy network** o_θ^m.



Parts as Neural Radiance Fields

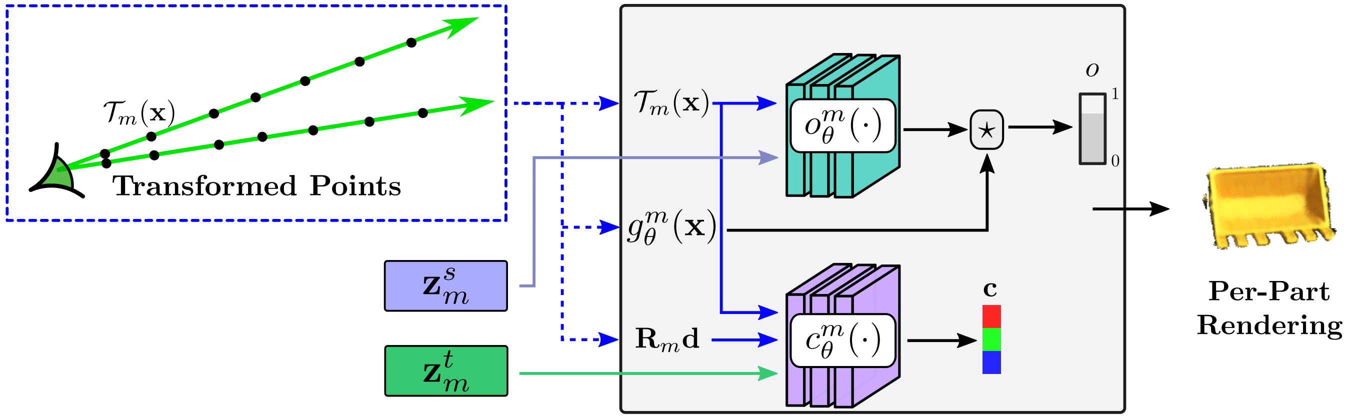
Parts as Neural Radiance Fields

- The **color network** c_θ^m and the **occupancy network** o_θ^m predict the color and occupancy value respectively.



Parts as Neural Radiance Fields

- The **color network** c_θ^m and the **occupancy network** o_θ^m predict the color and occupancy value respectively.

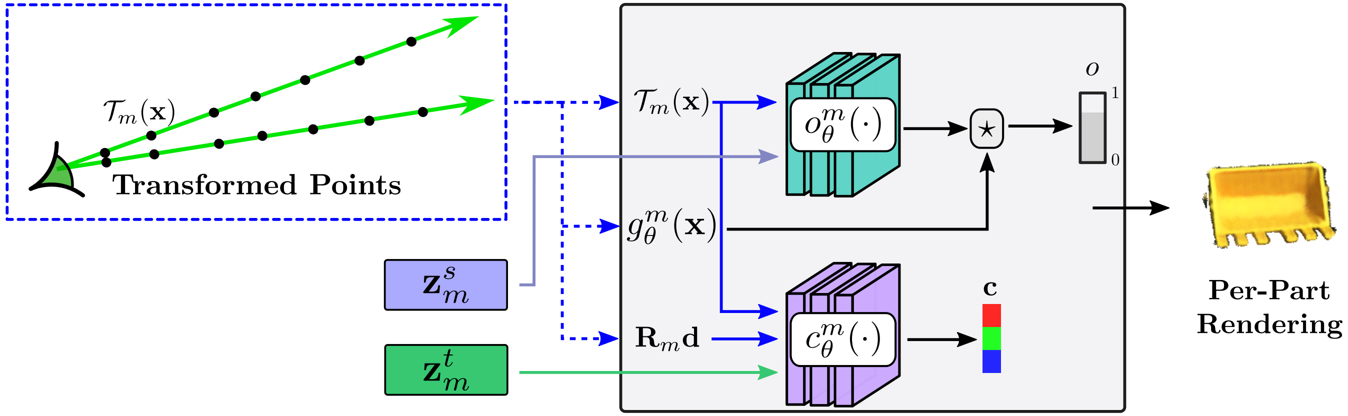


- To enforce that **each part captures local and continuous regions of the object**, we multiply the occupancy function with an **axis-aligned 3D ellipsoid occupancy function**:

$$h_\theta^m(\mathbf{x}) = o_\theta^m(\mathbf{x})g_\theta^m(\mathbf{x}),$$

Parts as Neural Radiance Fields

- The **color network** c_θ^m and the **occupancy network** o_θ^m predict the color and occupancy value respectively.



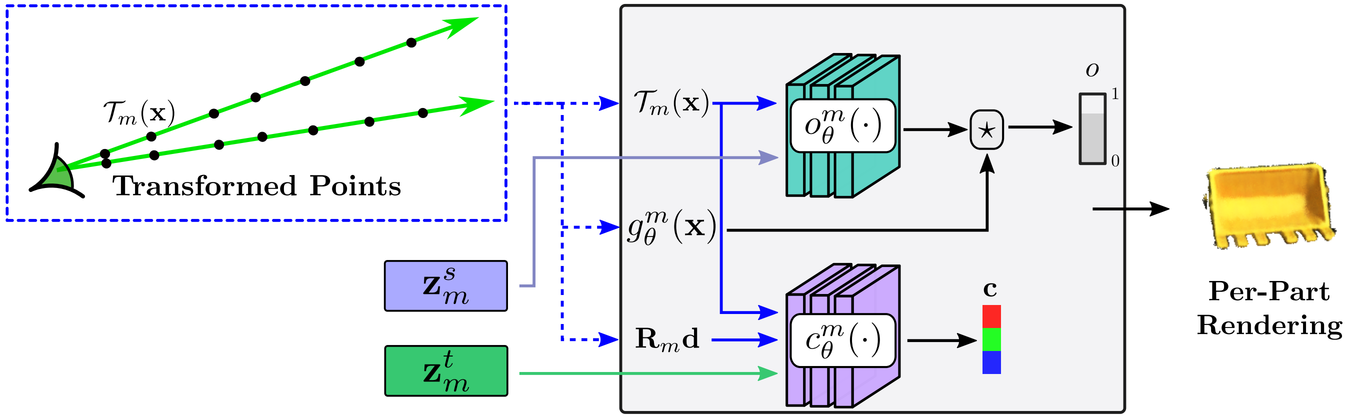
- To enforce that **each part captures local and continuous regions of the object**, we multiply the occupancy function with an **axis-aligned 3D ellipsoid occupancy function**:

$$h_\theta^m(\mathbf{x}) = o_\theta^m(\mathbf{x}) g_\theta^m(\mathbf{x}),$$

where $g_\theta^m(\mathbf{x}) = g(\mathcal{T}_m(\mathbf{x}), \mathbf{s}_m)$ is the **occupancy function of the** m-th **ellipsoid**.

Parts as Neural Radiance Fields

- The **color network** c_θ^m and the **occupancy network** o_θ^m predict the color and occupancy value respectively.



- To enforce that **each part captures local and continuous regions of the object**, we multiply the occupancy function with an **axis-aligned 3D ellipsoid occupancy function**:

$$h_\theta^m(\mathbf{x}) = o_\theta^m(\mathbf{x}) g_\theta^m(\mathbf{x}),$$

where $g_\theta^m(\mathbf{x}) = g(\mathcal{T}_m(\mathbf{x}), \mathbf{s}_m)$ is the **occupancy function of the** m-th **ellipsoid**.

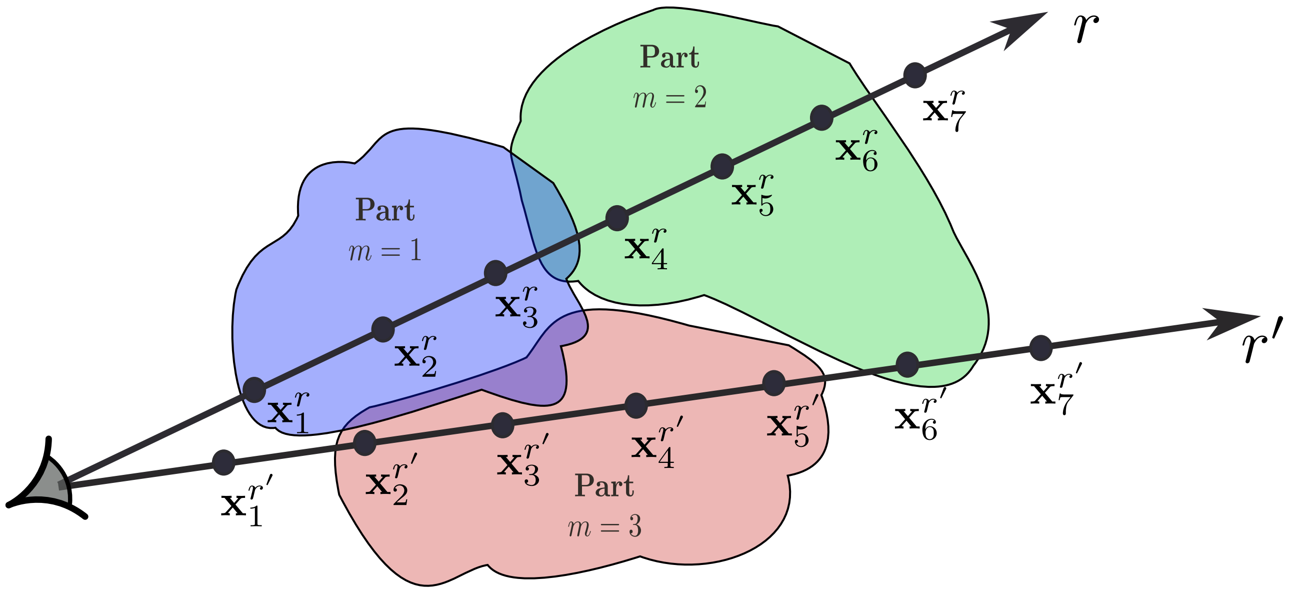
- The **per-part rendering equation** now becomes:

$$\hat{C}_m(r) = \sum_{i=1}^{N} h_\theta^m(\mathbf{x}_i^r) \prod_{j<i} \left(1 - h_\theta^m(\mathbf{x}_j^r)\right) \mathbf{c}_\theta^m(\mathbf{x}_i^r)$$

Hard Assignment between Rays and Parts

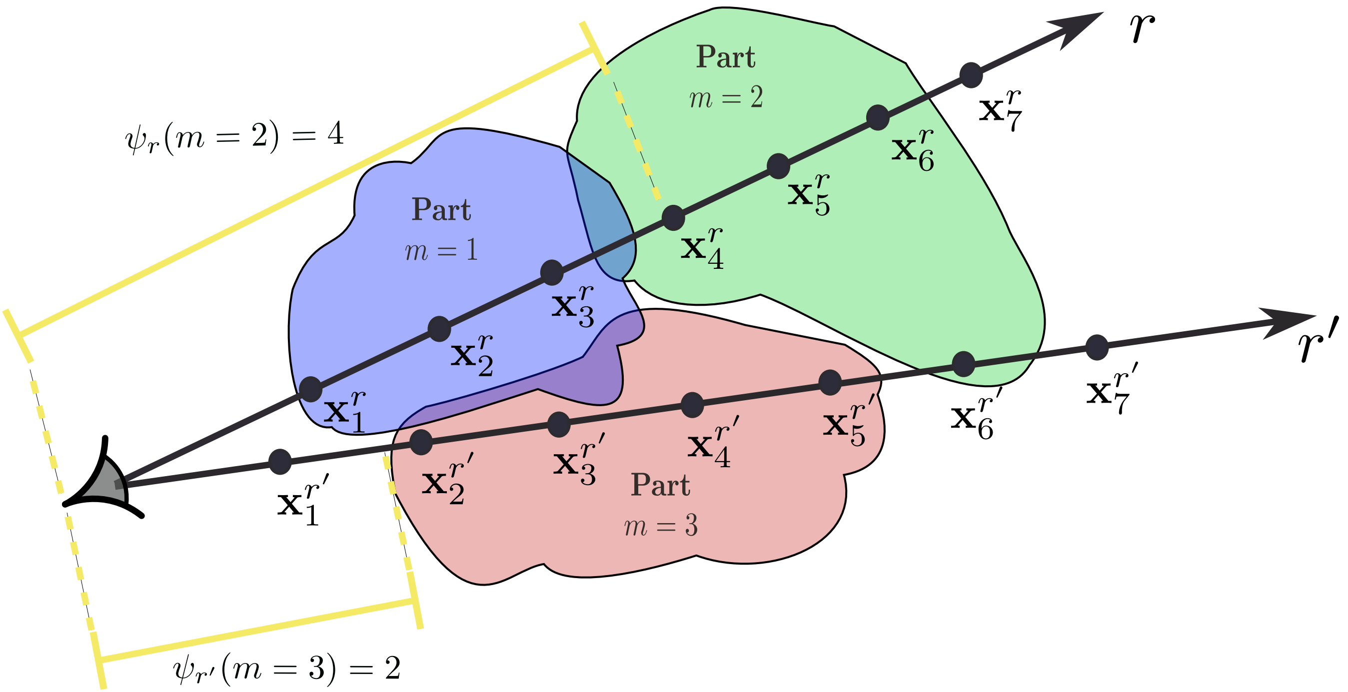
Hard Assignment between Rays and Parts

We impose a hard assignment between rays and parts, **associating a ray with the first part it itersects**.



Hard Assignment between Rays and Parts

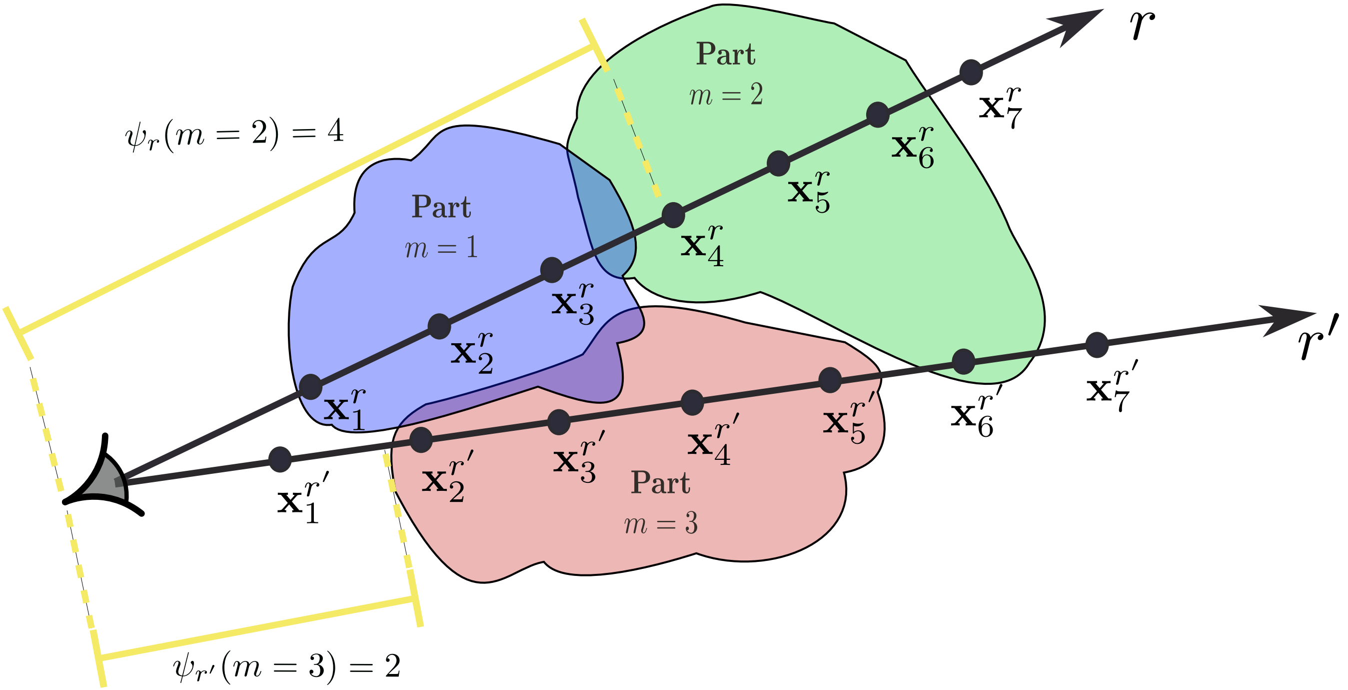
We impose a hard assignment between rays and parts, **associating a ray with the first part it itersects**.



$$\psi_r(m) = \min\{i \in \{1, \ldots, N\} \, : \, h_\theta^m(\mathbf{x}_i^r) \geq \tau\}$$

Hard Assignment between Rays and Parts

We impose a hard assignment between rays and parts, **associating a ray with the first part it itersects**.



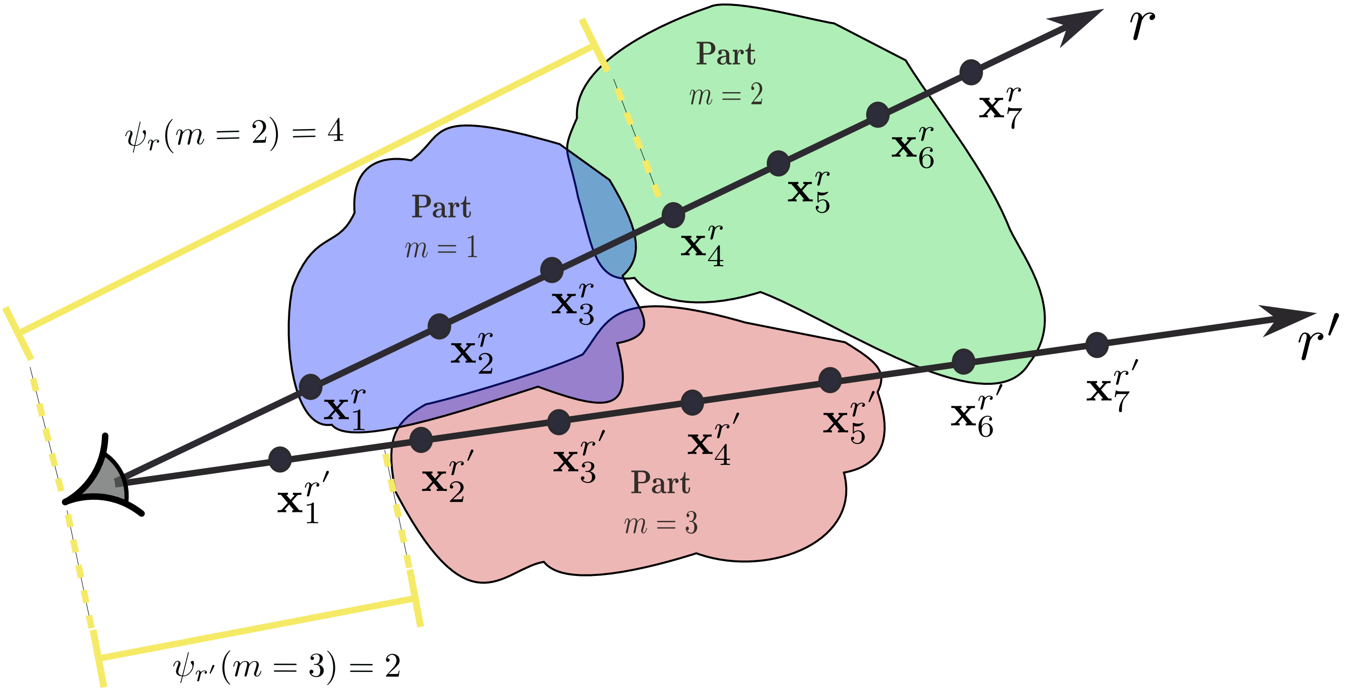
$$\psi_r(m) = \min\{i \in \{1, \ldots, N\} \,:\, h_\theta^m(\mathbf{x}_i^r) \geq \tau\}$$

We define the set of rays $\mathcal{R}_m$ as the **set of rays that first intersect with the** m-th **part**:

$$\mathcal{R}_m = \left\{ r \in \mathcal{R} \,:\, m = \operatorname*{arg\,min}_{k \in \{0 \ldots M\}} \psi_r(k) \right\}$$

Hard Assignment between Rays and Parts

We impose a hard assignment between rays and parts, **associating a ray with the first part it itersects**.



$$\psi_r(m) = \min\{i \in \{1, \ldots, N\} \; : \; h_\theta^m(\mathbf{x}_i^r) \geq \tau\}$$

The **rendering equation for the entire object** using M **NeRFs** becomes

$$\hat{C}(r) = \sum_{m=1}^{M} \mathbb{1}_{r \in \mathcal{R}_m} \hat{C}_m(r).$$

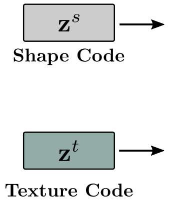
3D Object Generation

3D Object Generation

We are given a collection of **posed 2D images** of objects in a semantic class, along with the respective **object masks**.

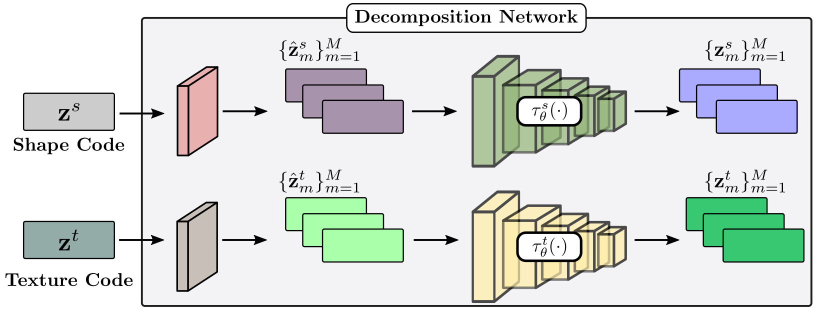
3D Object Generation

We are given a collection of **posed 2D images** of objects in a semantic class, along with the respective **object masks**.



3D Object Generation

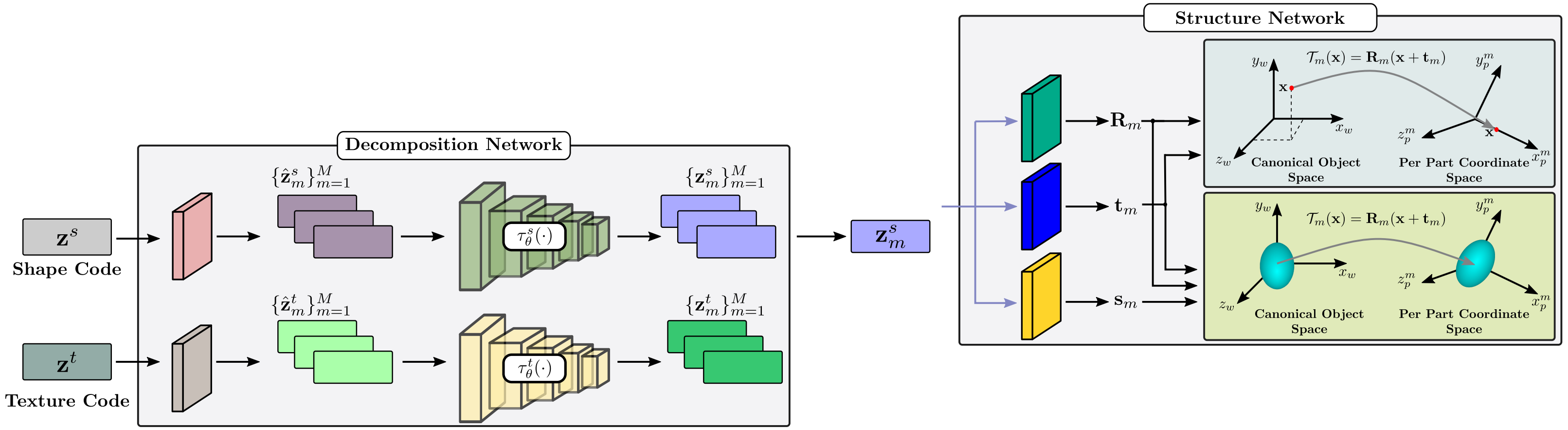
We are given a collection of **posed 2D images** of objects in a semantic class, along with the respective **object masks**.



- **Decomposition Network:** Maps $\mathbf{z}^s$ and $\mathbf{z}^t$ to M latent codes that control the per-part shape and texture.

3D Object Generation

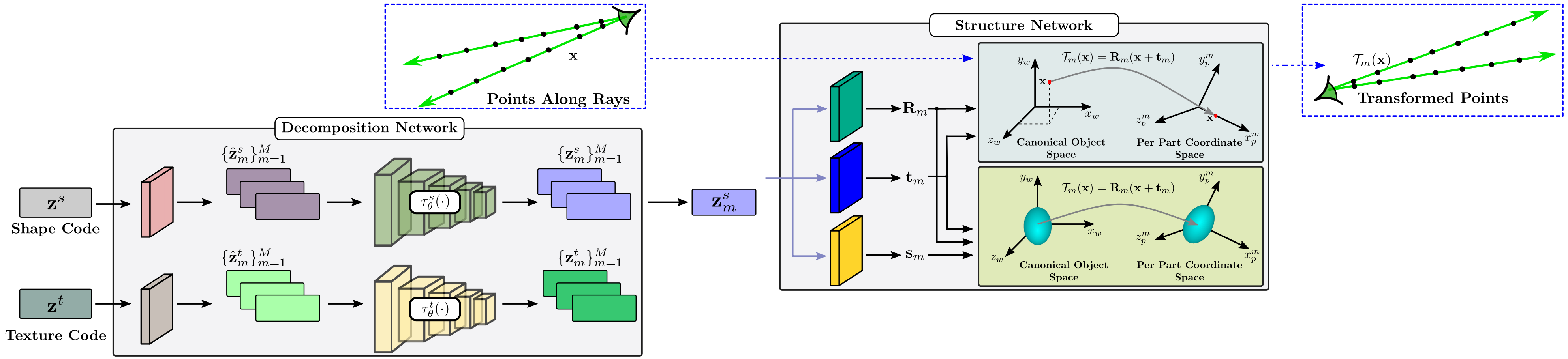
We are given a collection of **posed 2D images** of objects in a semantic class, along with the respective **object masks**.



- **Decomposition Network:** Maps $\mathbf{z}^s$ and $\mathbf{z}^t$ to M latent codes that control the per-part shape and texture.
- **Structure Network:** Predicts the pose and scale for each part m.

3D Object Generation

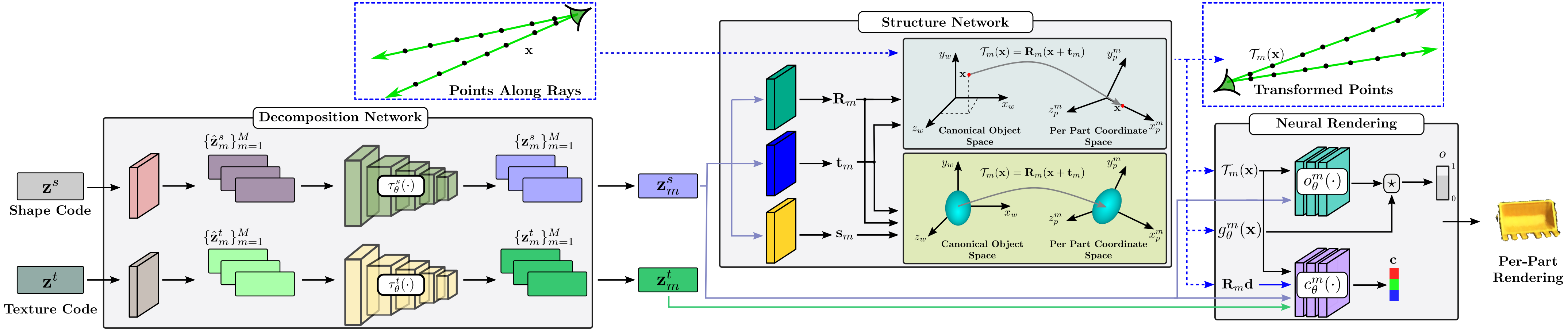
We are given a collection of **posed 2D images** of objects in a semantic class, along with the respective **object masks**.



- **Decomposition Network:** Maps $\mathbf{z}^s$ and $\mathbf{z}^t$ to M latent codes that control the per-part shape and texture.
- **Structure Network:** Predicts the pose and scale for each part m.

3D Object Generation

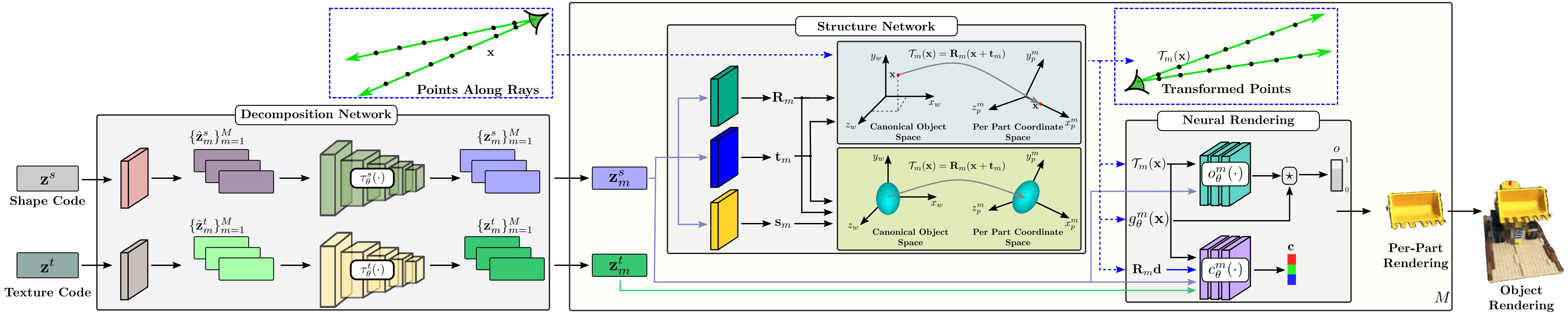
We are given a collection of **posed 2D images** of objects in a semantic class, along with the respective **object masks**.



- **Decomposition Network:** Maps $\mathbf{z}^s$ and $\mathbf{z}^t$ to M latent codes that control the per-part shape and texture.
- **Structure Network:** Predicts the pose and scale for each part m.
- **Neural Rendering Network:** Renders a 2D image using M locally defined NeRFs.

3D Object Generation

We are given a collection of **posed 2D images** of objects in a semantic class, along with the respective **object masks**.



- **Decomposition Network:** Maps $\mathbf{z}^s$ and $\mathbf{z}^t$ to M latent codes that control the per-part shape and texture.
- **Structure Network:** Predicts the pose and scale for each part m.
- **Neural Rendering Network:** Renders a 2D image using M locally defined NeRFs.

Training

Training

Our objective function $\mathcal{L}$ is composed of 6 terms, and 2 regularizers on the shape and texture embeddings $\mathbf{z}^s, \mathbf{z}^t$:

Training

Our objective function $\mathcal{L}$ is composed of 6 terms, and 2 regularizers on the shape and texture embeddings $\mathbf{z}^s, \mathbf{z}^t$:

$$\mathcal{L} = \mathcal{L}_{rgb}(\mathcal{R})$$

- **Reconstruction Loss:** The rendered and ground truth image colors should match.

Training

Our objective function $\mathcal{L}$ is composed of 6 terms, and 2 regularizers on the shape and texture embeddings $\mathbf{z}^s, \mathbf{z}^t$:

$$\mathcal{L} = \mathcal{L}_{rgb}(\mathcal{R}) + \mathcal{L}_{mask}(\mathcal{R})$$

- **Reconstruction Loss:** The rendered and ground truth image colors should match.
- **Mask Loss:** The rendered and ground truth mask values should match.

Training

Our objective function $\mathcal{L}$ is composed of 6 terms, and 2 regularizers on the shape and texture embeddings $\mathbf{z}^s, \mathbf{z}^t$:

$$\mathcal{L} = \mathcal{L}_{rgb}(\mathcal{R}) + \mathcal{L}_{mask}(\mathcal{R}) + \mathcal{L}_{occ}(\mathcal{R})$$

- **Reconstruction Loss:** The rendered and ground truth image colors should match.
- **Mask Loss:** The rendered and ground truth mask values should match.
- **Occupancy Loss:** The generated shape should occupy space.

Training

Our objective function $\mathcal{L}$ is composed of 6 terms, and 2 regularizers on the shape and texture embeddings $\mathbf{z}^s, \mathbf{z}^t$:

$$\mathcal{L} = \mathcal{L}_{rgb}(\mathcal{R}) + \mathcal{L}_{mask}(\mathcal{R}) + \mathcal{L}_{occ}(\mathcal{R}) + \mathcal{L}_{cov}(\mathcal{R})$$

- **Reconstruction Loss:** The rendered and ground truth image colors should match.
- **Mask Loss:** The rendered and ground truth mask values should match.
- **Occupancy Loss:** The generated shape should occupy space.
- **Coverage Loss:** The parts should be distributed along the object.

Training

Our objective function $\mathcal{L}$ is composed of 6 terms, and 2 regularizers on the shape and texture embeddings $\mathbf{z}^s, \mathbf{z}^t$:

$$\mathcal{L} = \mathcal{L}_{rgb}(\mathcal{R}) + \mathcal{L}_{mask}(\mathcal{R}) + \mathcal{L}_{occ}(\mathcal{R}) + \mathcal{L}_{cov}(\mathcal{R}) + \mathcal{L}_{overlap}(\mathcal{R})$$

- **Reconstruction Loss:** The rendered and ground truth image colors should match.
- **Mask Loss:** The rendered and ground truth mask values should match.
- **Occupancy Loss:** The generated shape should occupy space.
- **Coverage Loss:** The parts should be distributed along the object.
- **Overlapping Loss:** The parts should not be overlapping.

Training

Our objective function $\mathcal{L}$ is composed of 6 terms, and 2 regularizers on the shape and texture embeddings $\mathbf{z}^s, \mathbf{z}^t$:

$$\mathcal{L} = \mathcal{L}_{rgb}(\mathcal{R}) + \mathcal{L}_{mask}(\mathcal{R}) + \mathcal{L}_{occ}(\mathcal{R}) + \mathcal{L}_{cov}(\mathcal{R}) + \mathcal{L}_{overlap}(\mathcal{R}) + \mathcal{L}_{control}(\mathcal{R})$$

- **Reconstruction Loss:** The rendered and ground truth image colors should match.
- **Mask Loss:** The rendered and ground truth mask values should match.
- **Occupancy Loss:** The generated shape should occupy space.
- **Coverage Loss:** The parts should be distributed along the object.
- **Overlapping Loss:** The parts should not be overlapping.
- **Control Loss:** The parts should be of similar 3D volume.

Training

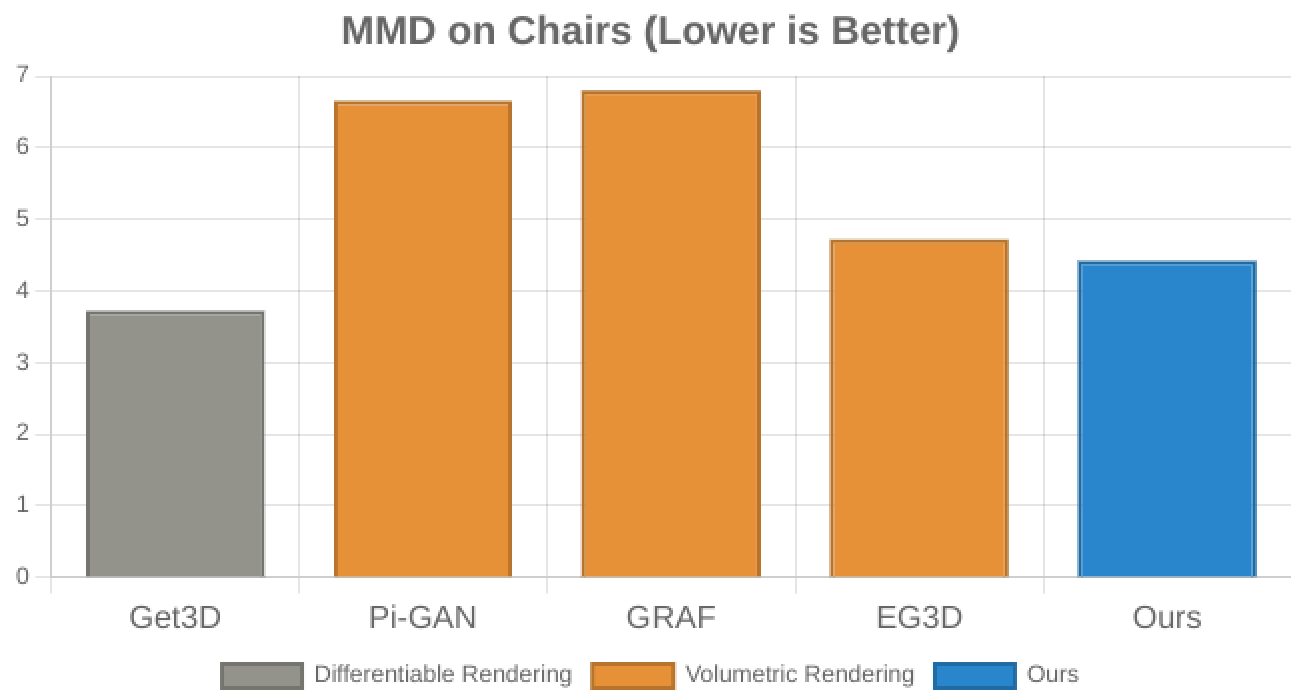
Our objective function $\mathcal{L}$ is composed of 6 terms, and 2 regularizers on the shape and texture embeddings $\mathbf{z}^s, \mathbf{z}^t$:

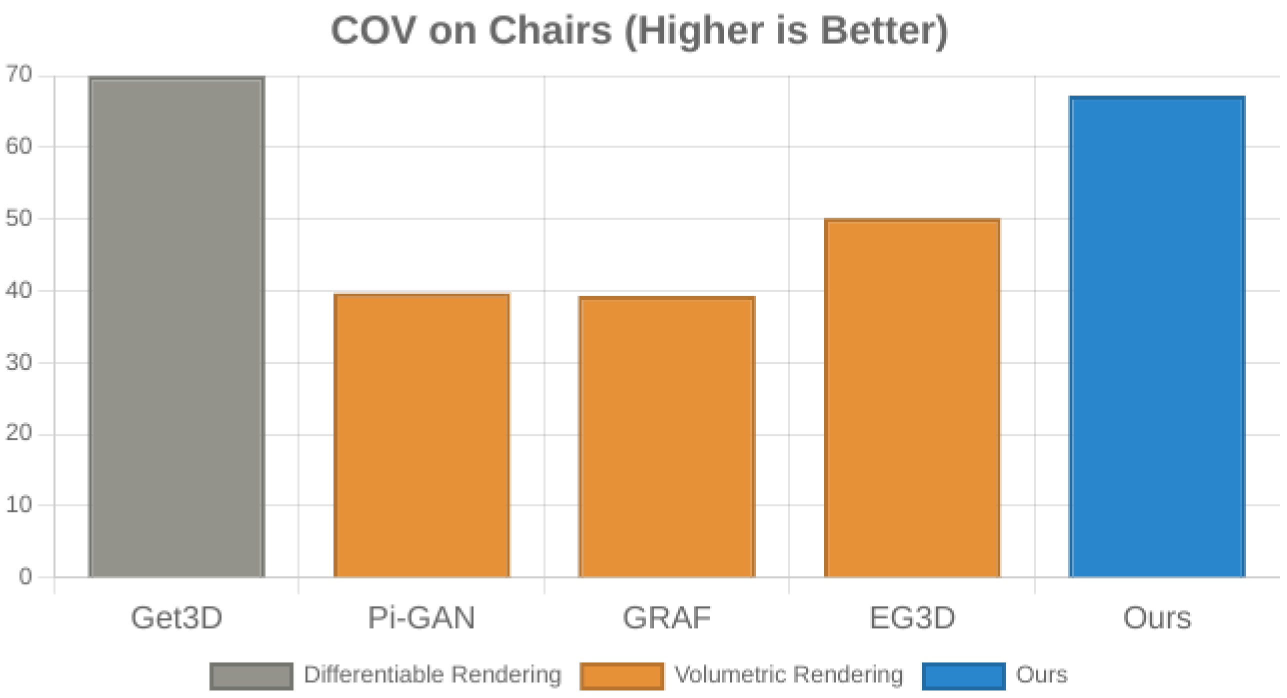
$$\mathcal{L} = \mathcal{L}_{rgb}(\mathcal{R}) + \mathcal{L}_{mask}(\mathcal{R}) + \mathcal{L}_{occ}(\mathcal{R}) + \mathcal{L}_{cov}(\mathcal{R}) + \mathcal{L}_{overlap}(\mathcal{R}) + \mathcal{L}_{control}(\mathcal{R}) + \ \|\mathbf{z}^s\|_2 + \|\mathbf{z}^t\|_2$$

- **Reconstruction Loss:** The rendered and ground truth image colors should match.
- **Mask Loss:** The rendered and ground truth mask values should match.
- **Occupancy Loss:** The generated shape should occupy space.
- **Coverage Loss:** The parts should be distributed along the object.
- **Overlapping Loss:** The parts should not be overlapping.
- **Control Loss:** The parts should be of similar 3D volume.

Quantitative Results

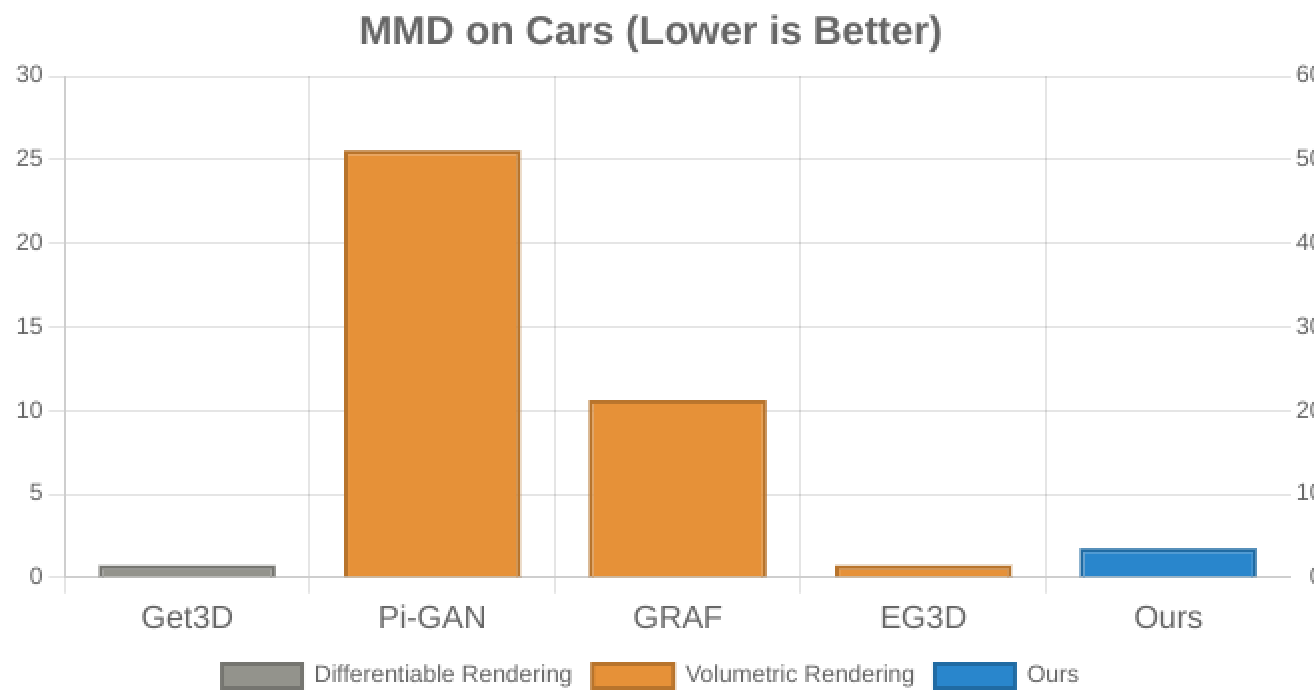
Comparisons with NeRF-based 3D Generative Methods

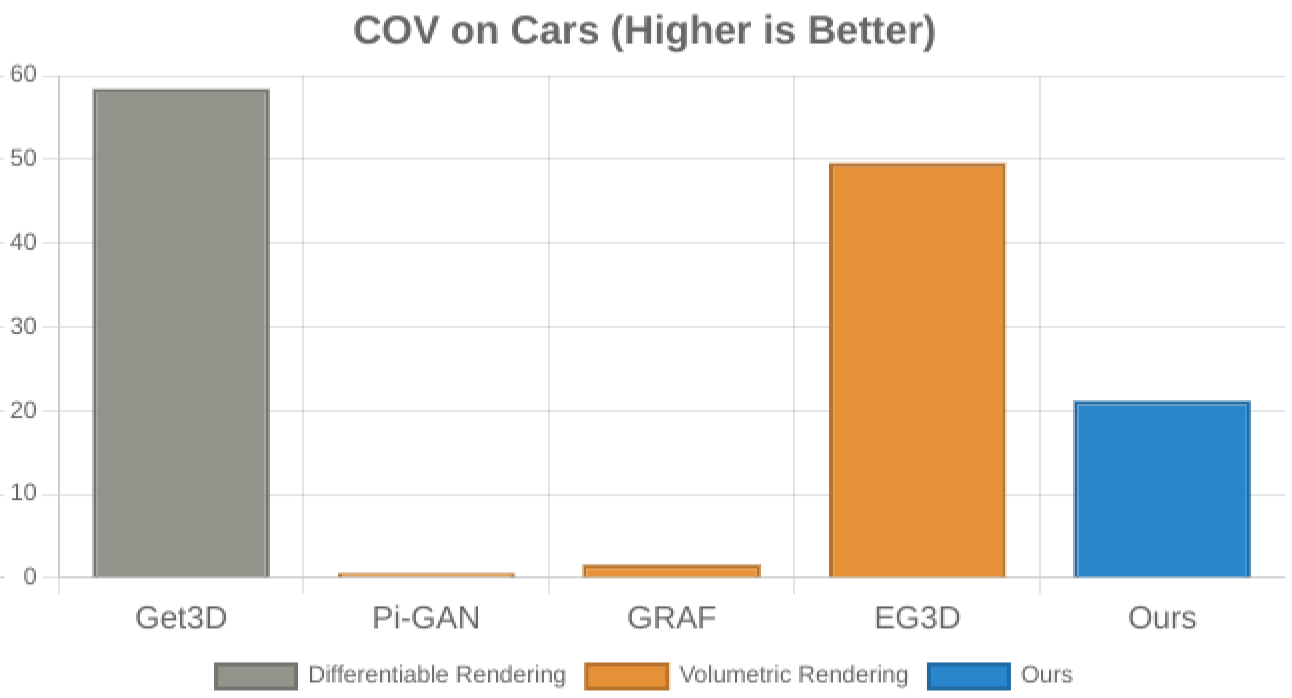




Quantitative Results

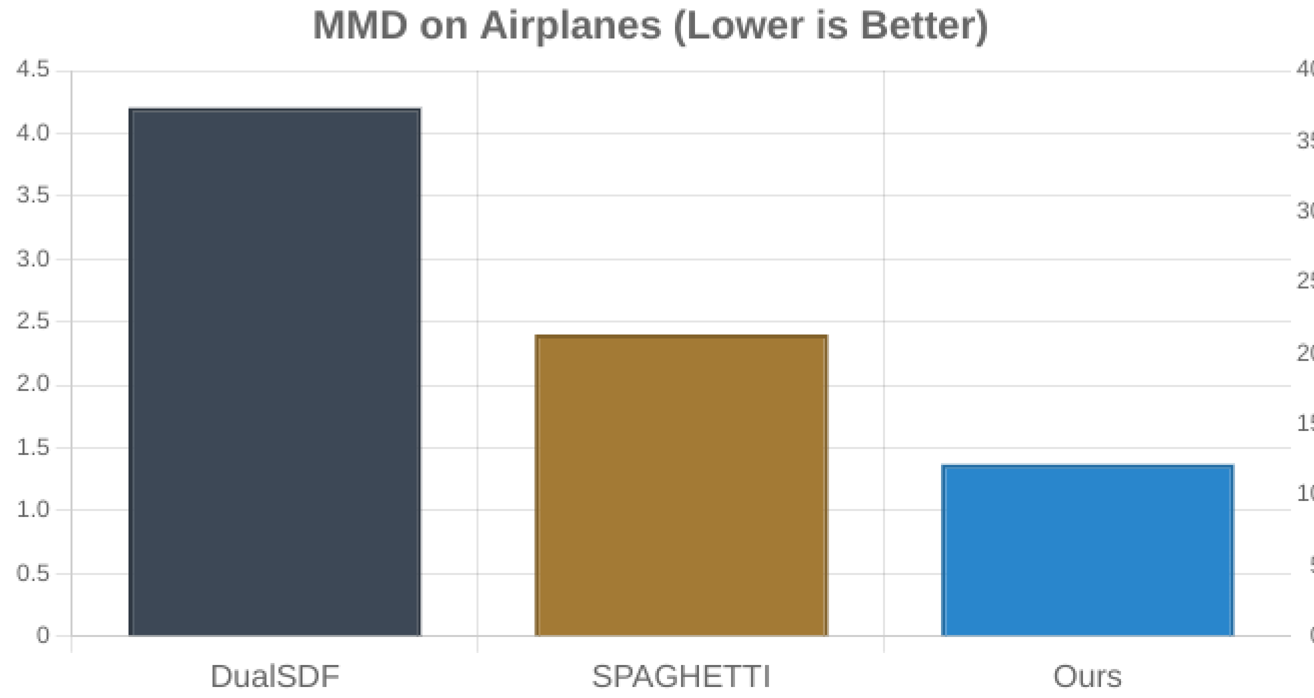
Comparisons with NeRF-based 3D Generative Methods

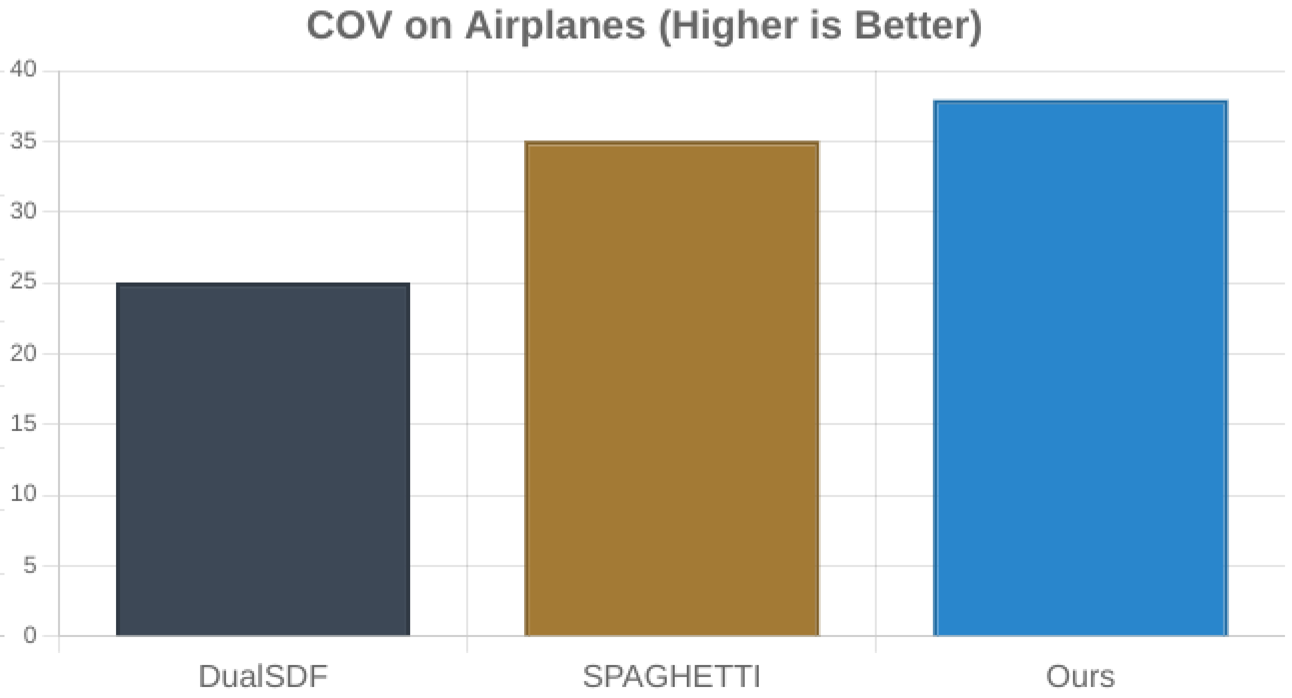




Quantitative Results

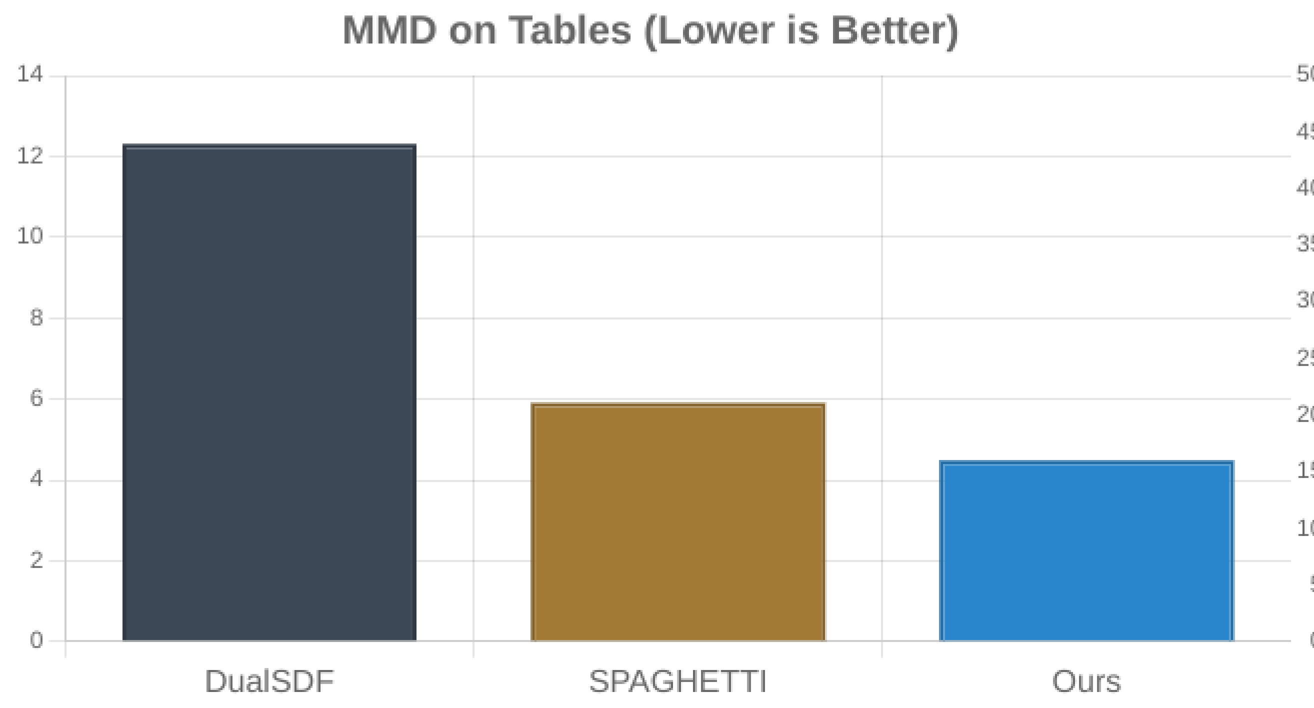
Comparisons with Part-Based 3D Generative Methods

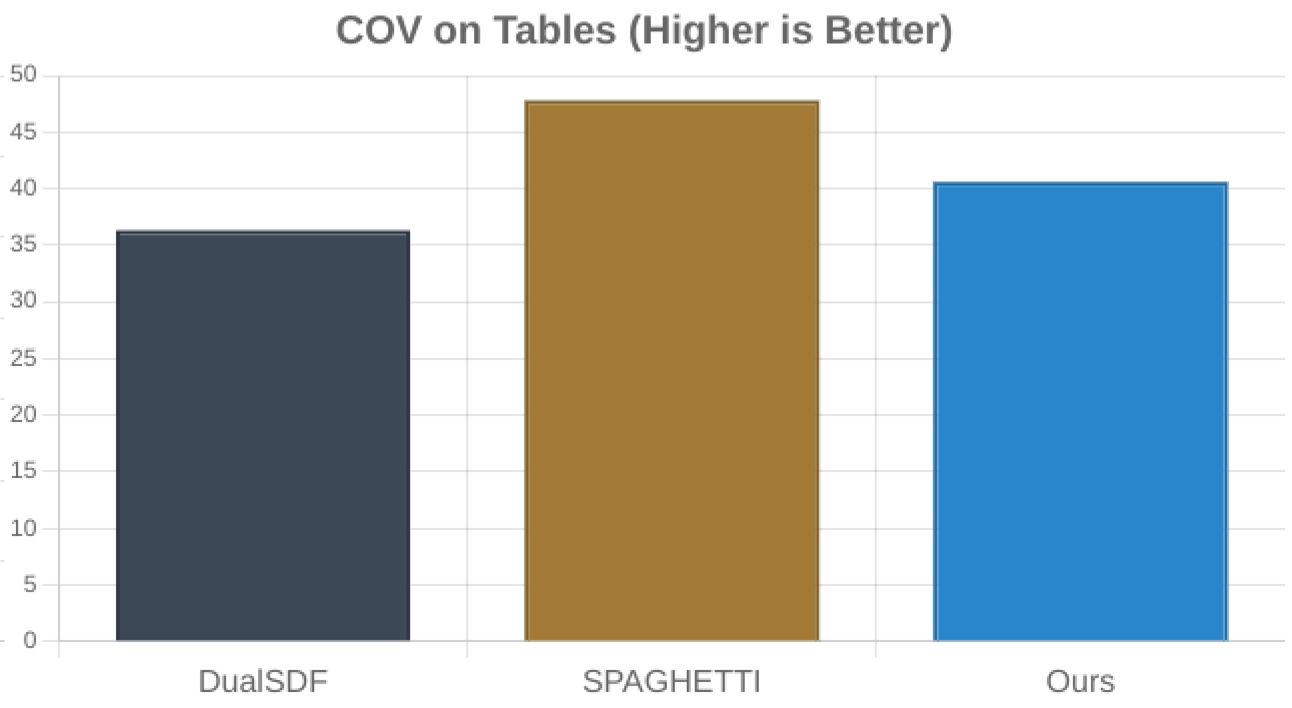




Quantitative Results

Comparisons with Part-Based 3D Generative Methods





Scene-Specific Editing

No editing

Rotation

Translation

Scaling

Scaling

Colorization

Shape Mixing

Shape Mixing

Shape #1

Shape Mixing

Shape #1

Shape #2

Shape Mixing

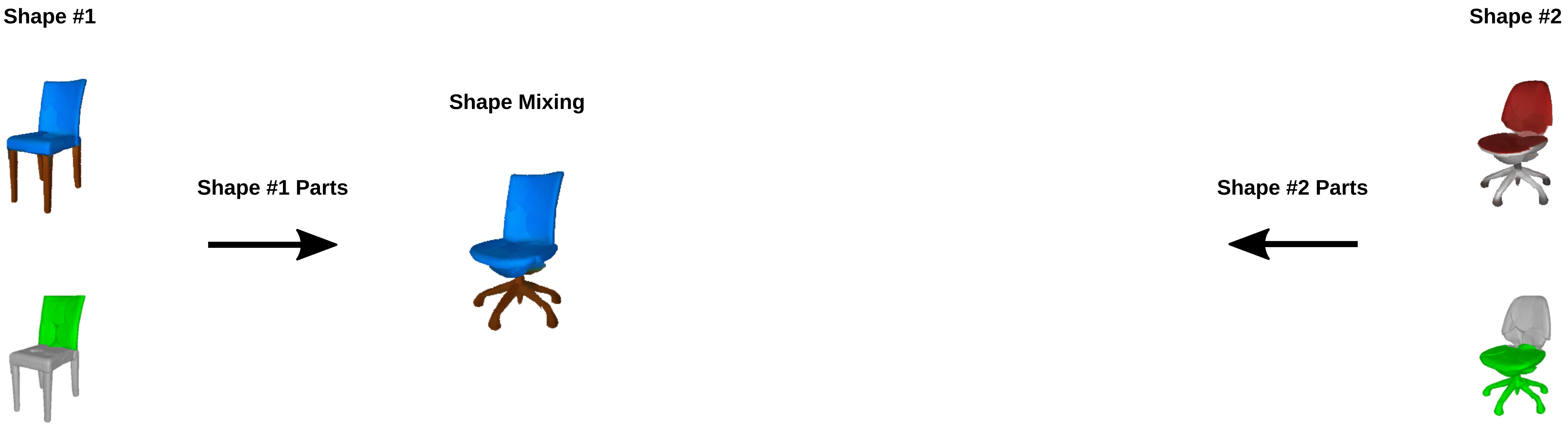
Shape #1

Shape #1 Parts

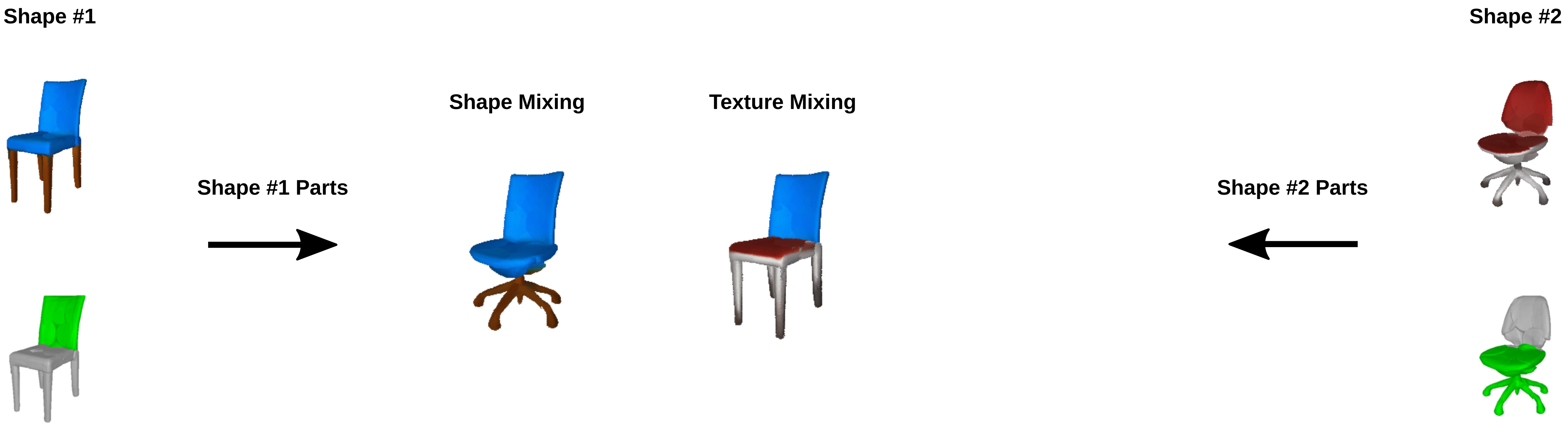
Shape #2

Shape #2 Parts

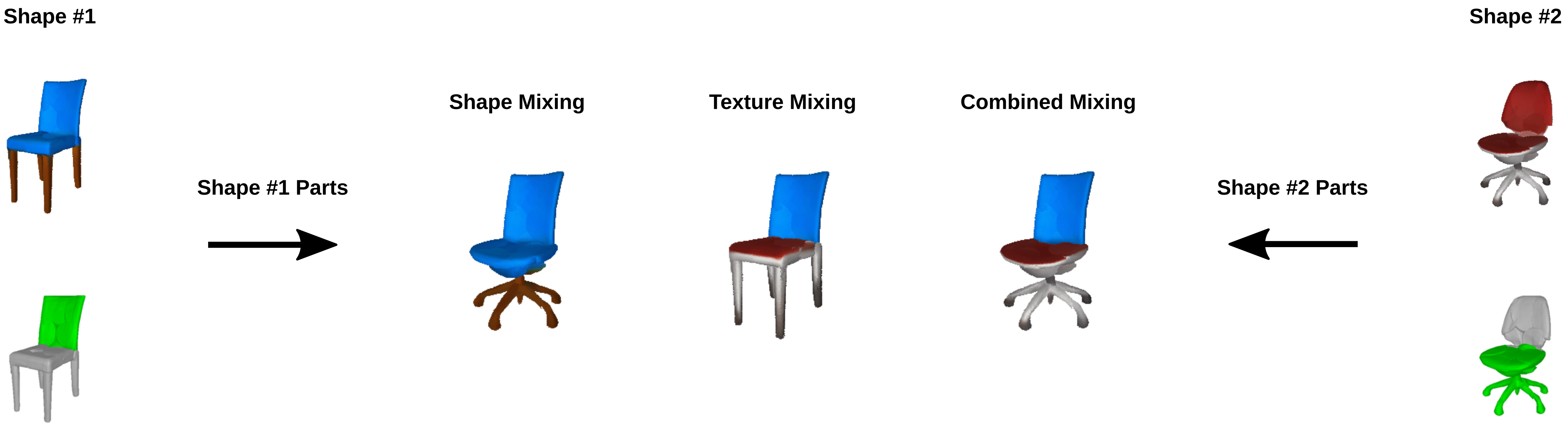
Shape Mixing



Shape Mixing



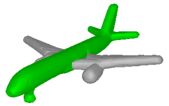
Shape Mixing



Shape Mixing

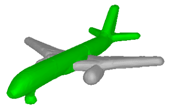
Shape Mixing

Shape #3

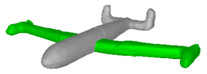


Shape Mixing

Shape #3

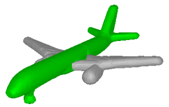


Shape #4



Shape Mixing

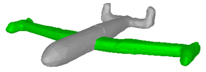
Shape #3



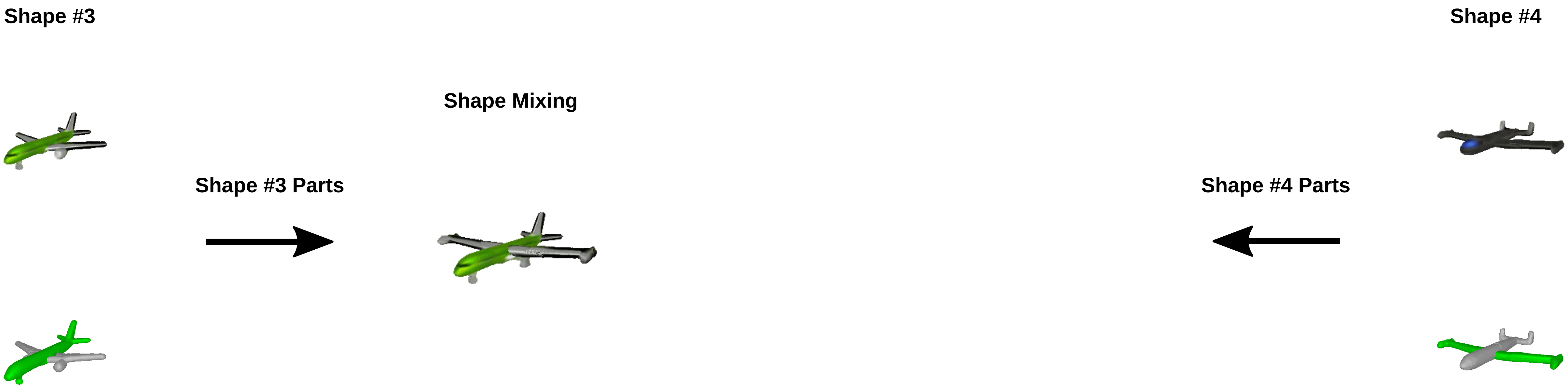
Shape #3 Parts

Shape #4 Parts

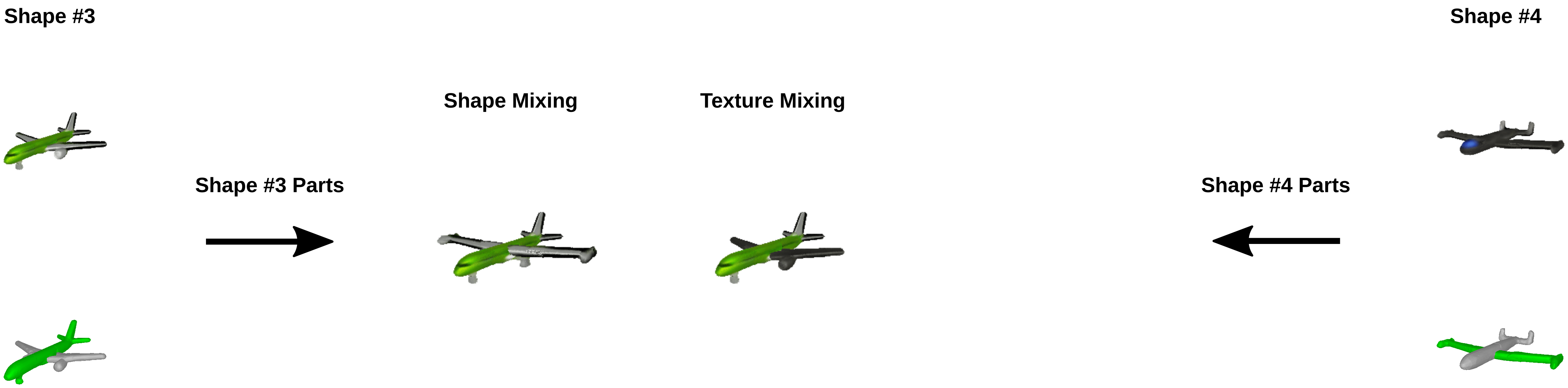
Shape #4



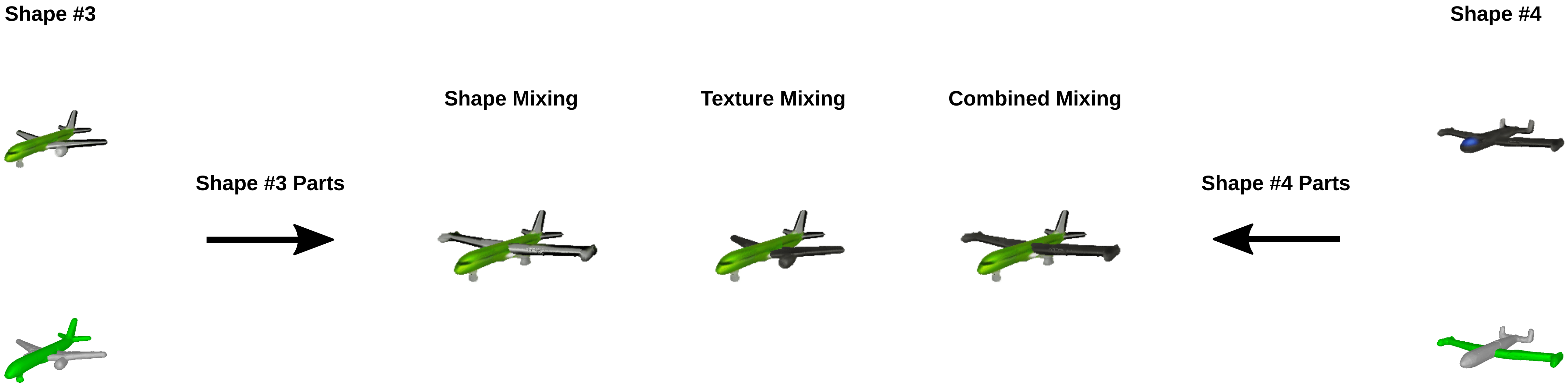
Shape Mixing



Shape Mixing



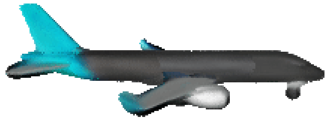
Shape Mixing



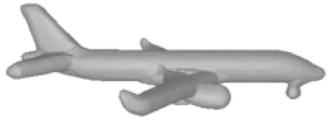
Shape Editing

Shape Editing

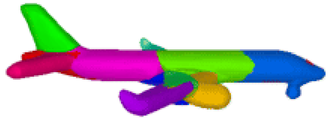
Renders



Geometry



Parts

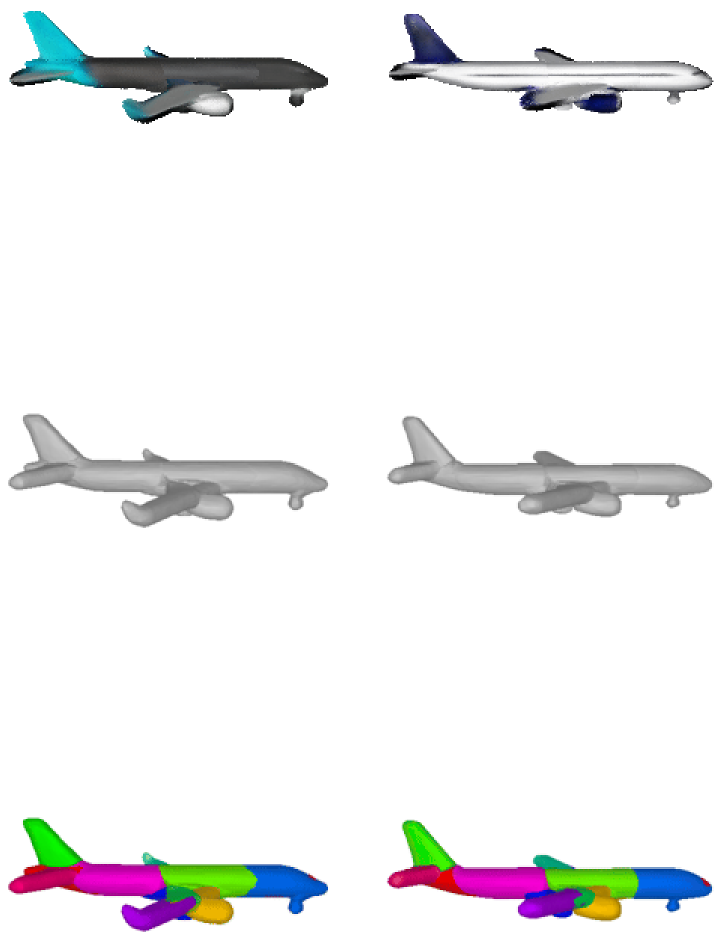


Shape Editing

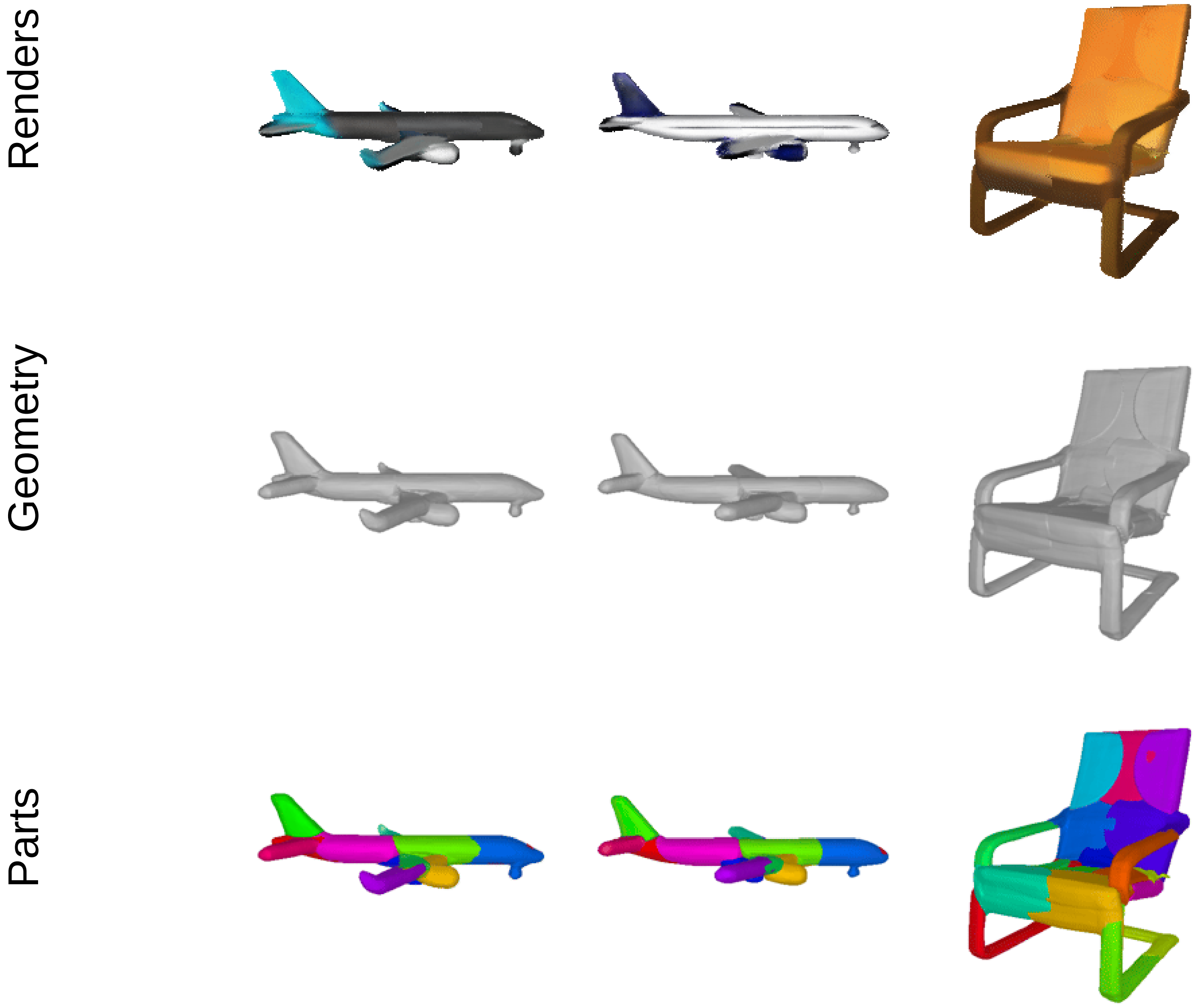
Renders

Geometry

Parts

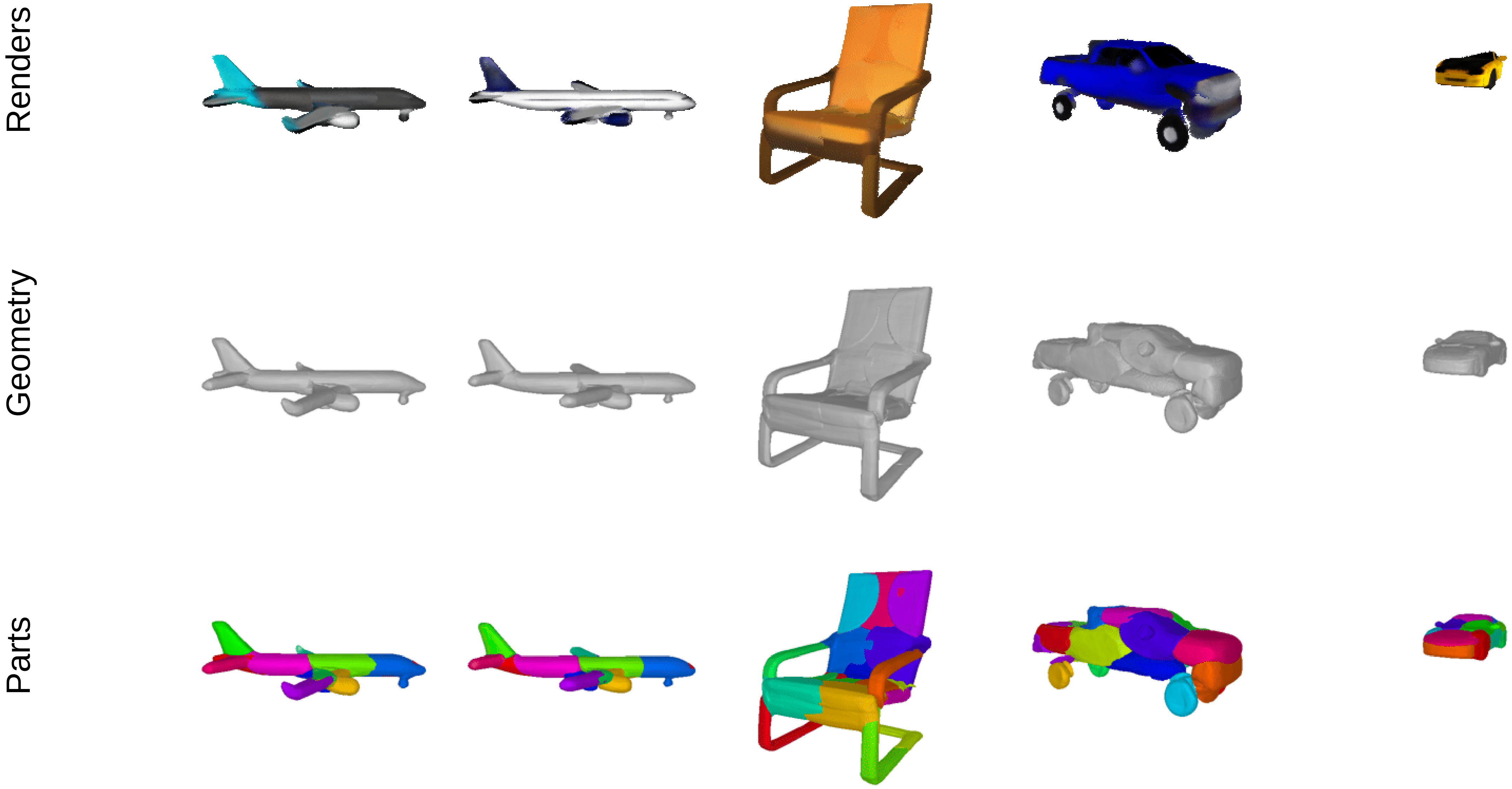


Shape Editing



Shape Editing

Shape Editing



Shape Generation

Summary & Limitations

Summary & Limitations

- The **first part-aware generative model that parameterizes parts as NeRFs**.

Summary & Limitations

- The **first part-aware generative model that parameterizes parts as NeRFs**.
- Our model enables **intuitive part-level control** and a broad range of editing operations not previously possible.

Summary & Limitations

- The **first part-aware generative model that parameterizes parts as NeRFs**.
- Our model enables **intuitive part-level control** and a broad range of editing operations not previously possible.
- Our model is **trained without explicit 3D supervision**.

Summary & Limitations

- The **first part-aware generative model that parameterizes parts as NeRFs**.
- Our model enables **intuitive part-level control** and a broad range of editing operations not previously possible.
- Our model is **trained without explicit 3D supervision**.

Limitations:

Summary & Limitations

- The **first part-aware generative model that parameterizes parts as NeRFs**.
- Our model enables **intuitive part-level control** and a broad range of editing operations not previously possible.
- Our model is **trained without explicit 3D supervision**.

Limitations:

- The quality of generated textures could be improved by employing GAN losses or triplane representations.

Summary & Limitations

- The **first part-aware generative model that parameterizes parts as NeRFs**.
- Our model enables **intuitive part-level control** and a broad range of editing operations not previously possible.
- Our model is **trained without explicit 3D supervision**.

Limitations:

- The quality of generated textures could be improved by employing GAN losses or triplane representations.
- The generated parts are not necessarily semantic.

Summary & Limitations

- The **first part-aware generative model that parameterizes parts as NeRFs**.
- Our model enables **intuitive part-level control** and a broad range of editing operations not previously possible.
- Our model is **trained without explicit 3D supervision**.

Limitations:

- The quality of generated textures could be improved by employing GAN losses or triplane representations.
- The generated parts are not necessarily semantic.

Project Page: https://ktertikas.github.io/part_nerf

CVPR Poster: *TUE-PM-032*