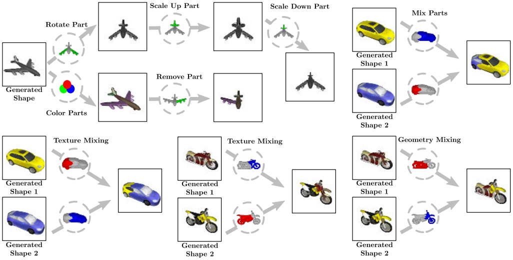
PartNeRF: Generating Part-Aware Editable 3D Shapes without 3D Supervision

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 Guibas [2]

[1] National and Kapodistrian University of Athens [2] Stanford University
[3] Athena RC, Greece [4] Institute of Advanced Research in Artificial Intelligence (IARAI)

https://ktertikas.github.io/part_nerf



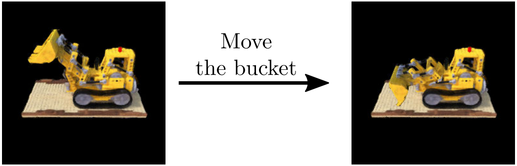
Compositional Representations for Object Synthesis

Shape editing involves **making local changes on the shape and appearance of different regions** of an object. For example, we want to be able to:

Compositional Representations for Object Synthesis

Shape editing involves **making local changes on the shape and appearance of different regions** of an object. For example, we want to be able to:

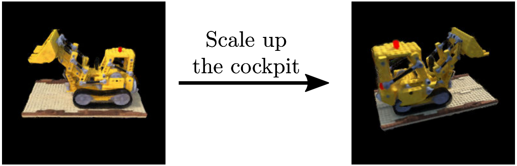
- **Apply rigid and non rigid transformations** on specific parts of the object.



Compositional Representations for Object Synthesis

Shape editing involves **making local changes on the shape and appearance of different regions** of an object. For example, we want to be able to:

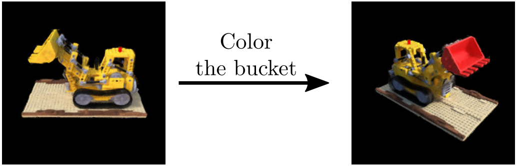
- **Apply rigid and non rigid transformations** on specific parts of the object.



Compositional Representations for Object Synthesis

Shape editing involves **making local changes on the shape and appearance of different regions** of an object. For example, we want to be able to:

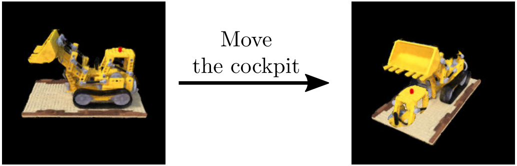
- **Apply rigid and non rigid transformations** on specific parts of the object.
- **Change the appearance** of a specific part of a 3D object.



Compositional Representations for Object Synthesis

Shape editing involves **making local changes on the shape and appearance of different regions** of an object. For example, we want to be able to:

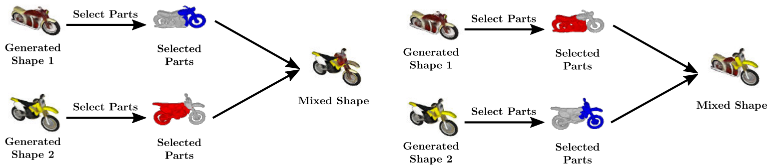
- **Apply rigid and non rigid transformations** on specific parts of the object.
- **Change the appearance** of a specific part of a 3D object.
- **Add or remove** a part.



Compositional Representations for Object Synthesis

Shape editing involves **making local changes on the shape and appearance of different regions** of an object. For example, we want to be able to:

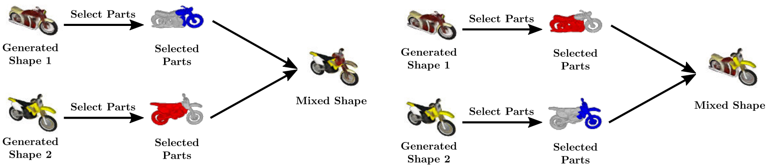
- **Apply rigid and non rigid transformations** on specific parts of the object.
- **Change the appearance** of a specific part of a 3D object.
- **Add or remove** a part.
- **Combine parts from different objects**.



Compositional Representations for Object Synthesis

Shape editing involves **making local changes on the shape and appearance of different regions** of an object. For example, we want to be able to:

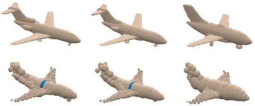
- **Apply rigid and non rigid transformations** on specific parts of the object.
- **Change the appearance** of a specific part of a 3D object.
- **Add or remove** a part.
- **Combine parts from different objects**.



We can specify **what object regions to edit** through parts.

Generative Models for 3D Object Synthesis

Part-based Generative Models

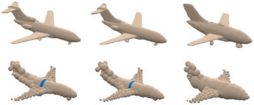


Hao et al. 2020

Hertz et al. 2022

Generative Models for 3D Object Synthesis

Part-based Generative Models



Hao et al. 2020

Hertz et al. 2022

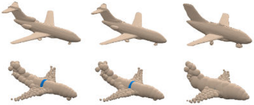
✓ Explicit part-level control

✗ Require 3D supervision

✗ Cannot change the appearance of an object

Generative Models for 3D Object Synthesis

Part-based Generative Models



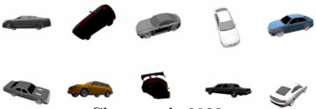
Hao et al. 2020

Hertz et al. 2022

NeRF-based Generative Models

Schwarz et al. 2020

Chan et al. 2021

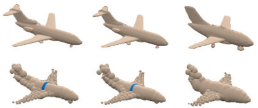


Chan et al. 2022

✓ Explicit part-level control
✕ Require 3D supervision
✕ Cannot change the appearance of an object

Generative Models for 3D Object Synthesis

Part-based Generative Models



Hao et al. 2020

Hertz et al. 2022

NeRF-based Generative Models

Schwarz et al. 2020

Chan et al. 2021

Chan et al. 2022

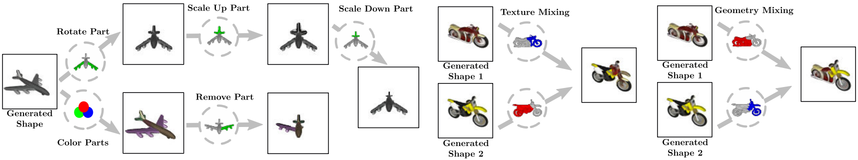
Part-based Generative Models:

✓ Explicit part-level control
✗ Require 3D supervision
✗ Cannot change the appearance of an object

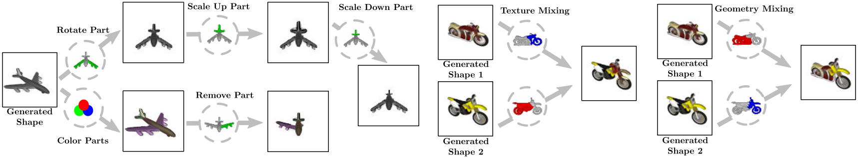
NeRF-based Generative Models:

✓ Can generate high quality 3D meshes
✓ Require 2D supervision during training
✗ No explicit part-level control

Can we learn a **part-aware generative model of 3D objects** capable of performing **local edits** on the **shape and appearance** of the generated 3D object?



Can we learn a **part-aware generative model of 3D objects** capable of performing **local edits** on the **shape and appearance** of the generated 3D object?



Bonus: We want our model to be **trained only from posed images**!!!

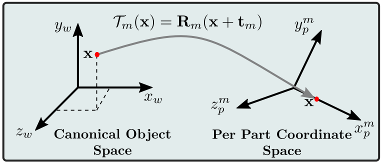
Key Idea: Enable local control through parts

We represent each object using M parts, where **each part is parameterized as a NeRF**. Each part is equipped with:

Key Idea: Enable local control through parts

We represent each object using M parts, where **each part is parameterized as a NeRF**. Each part is equipped with:

1. an *affine transformation* $T_m(\mathbf{x}) = \mathbf{R}_m(\mathbf{x} + \mathbf{t}_m)$ that maps a 3D point $\mathbf{x}$ to the **local coordinate system of the part**, where $\mathbf{t}_m \in \mathbb{R}^3$ is the translation vector and $\mathbf{R}_m \in SO(3)$ is the rotation matrix

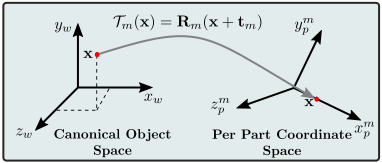


Affine Transformation

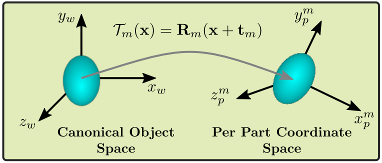
Key Idea: Enable local control through parts

We represent each object using M parts, where **each part is parameterized as a NeRF**. Each part is equipped with:

1. an *affine transformation* $T_m(\mathbf{x}) = \mathbf{R}_m(\mathbf{x} + \mathbf{t}_m)$ that maps a 3D point $\mathbf{x}$ to the **local coordinate system of the part**, where $\mathbf{t}_m \in \mathbb{R}^3$ is the translation vector and $\mathbf{R}_m \in SO(3)$ is the rotation matrix
2. a *scale* vector $\mathbf{s}_m \in \mathbb{R}^3$, representing the **spatial extent of each part**



Affine Transformation

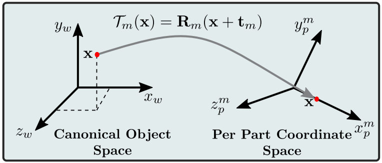


Spatial Extent

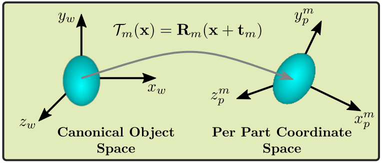
Key Idea: Enable local control through parts

We represent each object using M parts, where **each part is parameterized as a NeRF**. Each part is equipped with:

1. an *affine transformation* $T_m(\mathbf{x}) = \mathbf{R}_m(\mathbf{x} + \mathbf{t}_m)$ that maps a 3D point $\mathbf{x}$ to the **local coordinate system of the part**, where $\mathbf{t}_m \in \mathbb{R}^3$ is the translation vector and $\mathbf{R}_m \in SO(3)$ is the rotation matrix
2. a *scale* vector $\mathbf{s}_m \in \mathbb{R}^3$, representing the **spatial extent of each part**
3. two latent codes: **shape** $\mathbf{z}_m^s \in \mathbb{R}^{L_s}$ and **texture** $\mathbf{z}_m^t \in \mathbb{R}^{L_t}$ that control and shape and the appearance of each part.



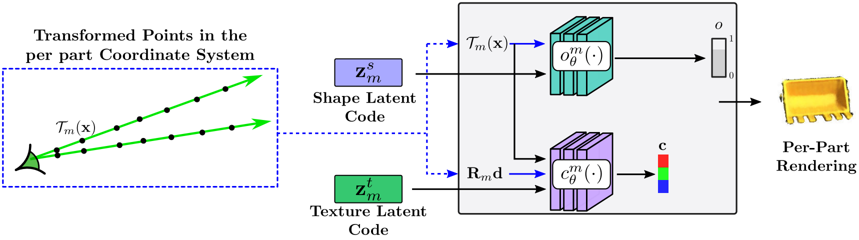
Affine Transformation



Spatial Extent

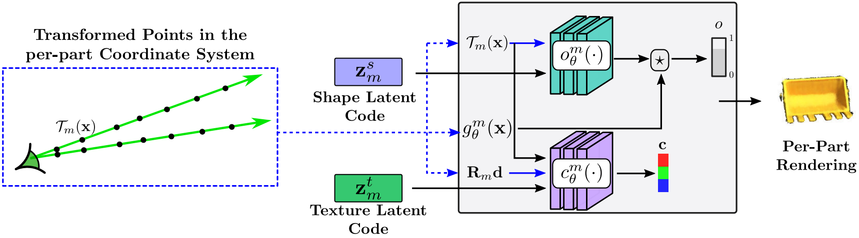
Part Representation

We employ two networks: a **color network** c_θ and an **occupancy network** o_θ to predict the color and the occupancy value respectively.



Part Representation

We employ two networks: a **color network** c_θ and an **occupancy network** o_θ to predict the color and the occupancy value respectively.



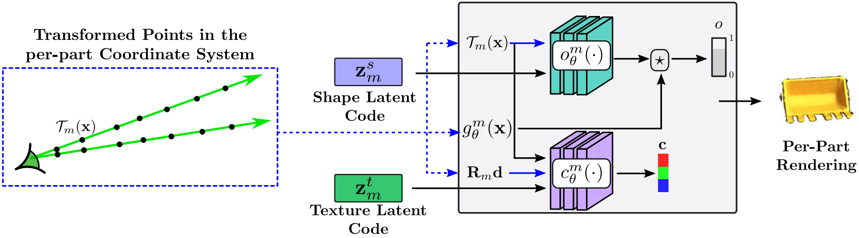
To enforce that **each part only captures continuous regions of the object**, we multiply its occupancy function with **the occupancy function of an axis-aligned 3D ellipsoid**

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

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

Part Representation

We employ two networks: a **color network** c_θ and an **occupancy network** o_θ to predict the color and the occupancy value respectively.



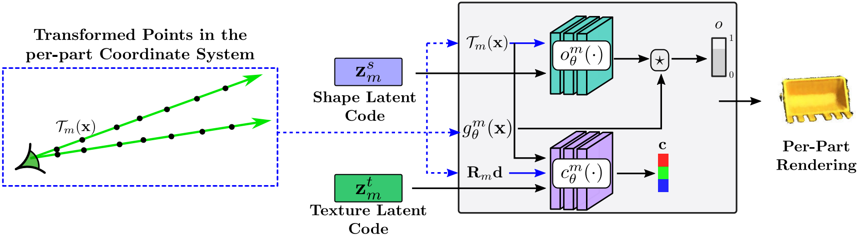
To enforce that **each part only captures continuous regions of the object**, we multiply its occupancy function with **the occupancy function of an axis-aligned 3D ellipsoid**

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

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

Part Rendering

We employ two networks: a **color network** c_θ and an **occupancy network** o_θ to predict the color and the occupancy value respectively.

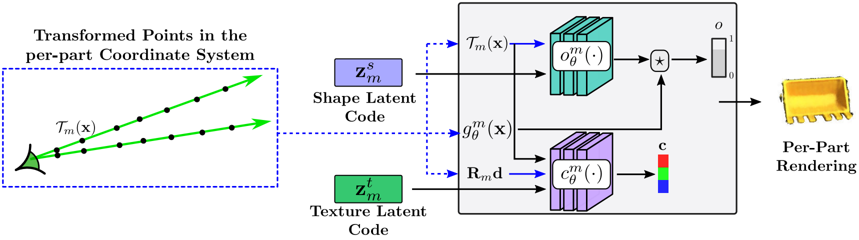


Instead of predicting volume densities, we predict occupancy values and the **rendering equation** of the m-th part becomes:

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

Part Rendering

We employ two networks: a **color network** c_θ and an **occupancy network** o_θ to predict the color and the occupancy value respectively.



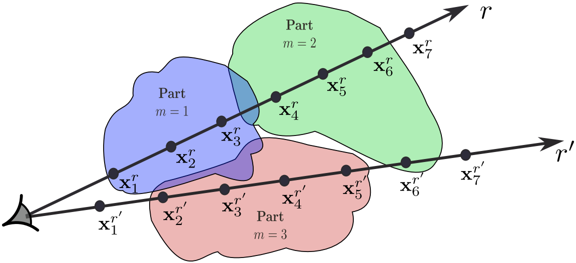
Instead of predicting volume densities, we predict occupancy values and the **rendering equation** of the m-th part becomes:

$$\hat{C}_m(r) = \sum_{i=1}^{N} h_\theta^m(\mathbf{x}_i^r) \prod_{j<i} (1 - h_\theta^m(\mathbf{x}_j^r))\, c_\theta^m(\mathbf{x}_i^r, \mathbf{d}^r)$$

where $h_\theta^m(\mathbf{x}_i^r)$ is the occupancy value at point $\mathbf{x}_i^r$ and $c_\theta^m(\mathbf{x}_i^r, \mathbf{d}^r)$ its color.

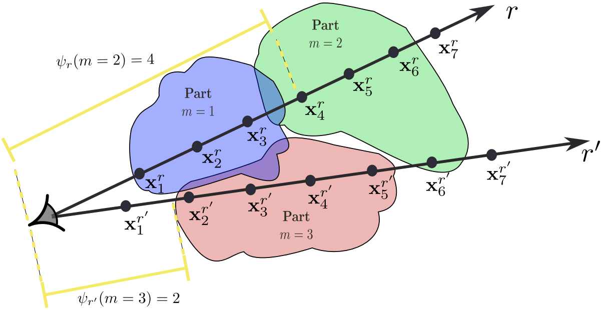
Hard Assignment between Rays and Parts

Given the ordered set of points $\mathcal{X}_r$ sampled along ray r, we define a hard assignment between rays and parts, by **associating a ray with the first part it intersects**.



Hard Assignment between Rays and Parts

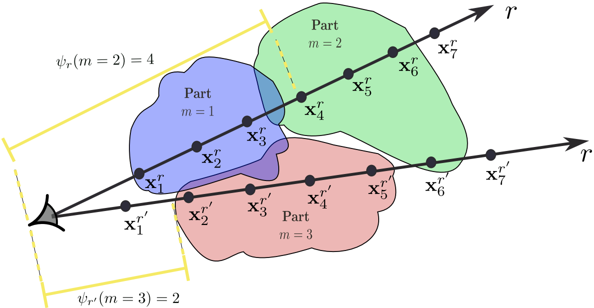
Given the ordered set of points $\mathcal{X}_r$ sampled along ray r, we define a hard assignment between rays and parts, by **associating a ray with the first part it intersects**.



$$\underbrace{\psi_r(m)}_{\text{Index of the first point inside each part intersecting with each ray}} = \min\left\{i \in \{1,\ldots,N\} \,:\, h_\theta^m(\mathbf{x}_i^r) \geq \tau\right\}$$

Hard Assignment between Rays and Parts

Given the ordered set of points $\mathcal{X}_r$ sampled along ray r, we define a hard assignment between rays and parts, by **associating a ray with the first part it intersects**.

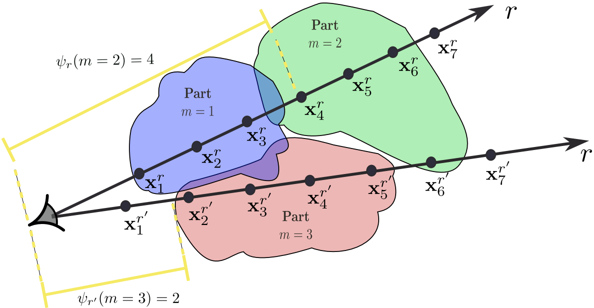


We define the set of rays $\mathcal{R}_m$ associated with the m-th part, as **the set of rays that first intersect with it**, namely:

$$\underbrace{\mathcal{R}_m}_{\substack{\text{Set of rays}\\ \text{assigned to part} m}} = \left\{ r \in \mathcal{R} \,:\, m = \underset{k \in \{0 \ldots M\}}{\operatorname{argmin}} \,\psi_r(k) \right\}.$$

Hard Assignment between Rays and Parts

Given the ordered set of points $\mathcal{X}_r$ sampled along ray r, we define a hard assignment between rays and parts, by **associating a ray with the first part it intersects**.

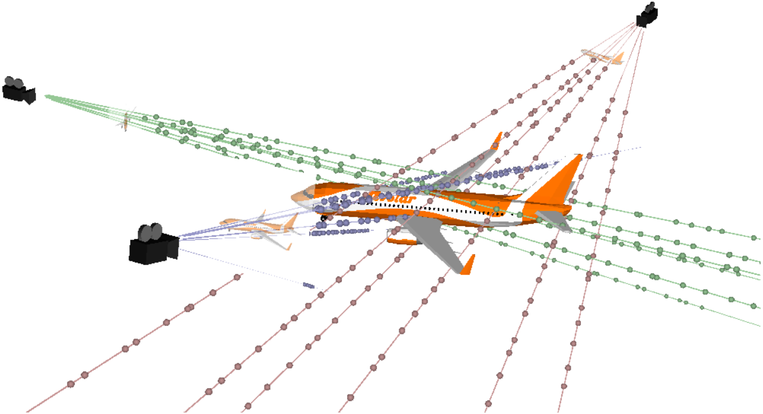


The **rendering equation for the entire object using M NeRFs** becomes

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

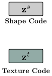
Object Generation

We are given a **collection of posed 2D images of objects** in a semantic class, **each accompanied by an object mask**. The latter is a binary image indicating whether each pixel is inside the object or not.



Object Generation

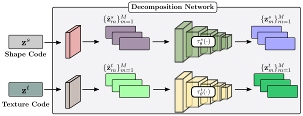
We are given a **collection of posed 2D images of objects** in a semantic class, **each accompanied by an object mask**. The latter is a binary image indicating whether each pixel is inside the object or not.



We implement our generative model as an **auto-decoder** that consists of:

Object Generation

We are given a **collection of posed 2D images of objects** in a semantic class, **each accompanied by an object mask**. The latter is a binary image indicating whether each pixel is inside the object or not.

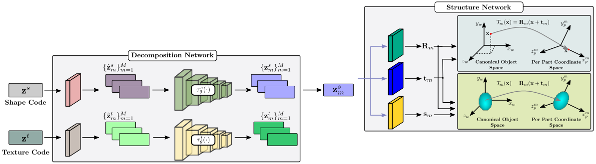


We implement our generative model as an **auto-decoder** that consists of:

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

Object Generation

We are given a **collection of posed 2D images of objects** in a semantic class, **each accompanied by an object mask**. The latter is a binary image indicating whether each pixel is inside the object or not.

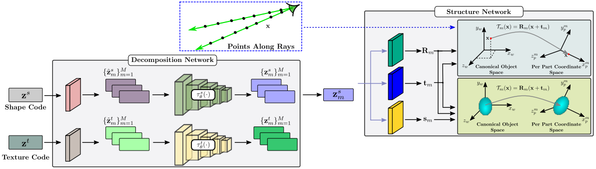


We implement our generative model as an **auto-decoder** that consists of:

- **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 the scale for each part m.

Object Generation

We are given a **collection of posed 2D images of objects** in a semantic class, **each accompanied by an object mask**. The latter is a binary image indicating whether each pixel is inside the object or not.

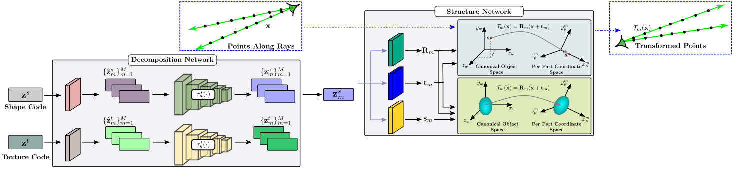


We implement our generative model as an **auto-decoder** that consists of:

- **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 the scale for each part m.

Object Generation

We are given a **collection of posed 2D images of objects** in a semantic class, **each accompanied by an object mask**. The latter is a binary image indicating whether each pixel is inside the object or not.

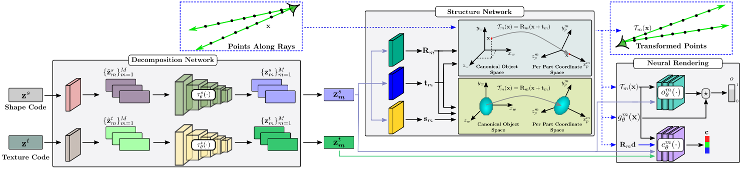


We implement our generative model as an **auto-decoder** that consists of:

- **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 the scale for each part m.

Object Generation

We are given a **collection of posed 2D images of objects** in a semantic class, **each accompanied by an object mask**. The latter is a binary image indicating whether each pixel is inside the object or not.

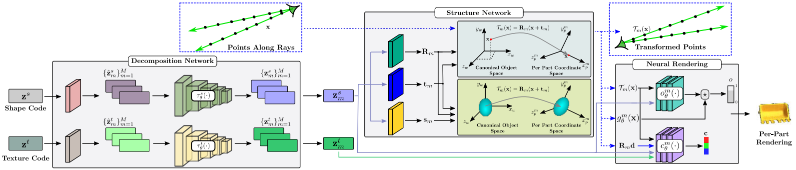


We implement our generative model as an **auto-decoder** that consists of:

- **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 the scale for each part m.
- **Neural Rendering Network:** Renders a 2D image using M locally defined NeRFs.

Object Generation

We are given a **collection of posed 2D images of objects** in a semantic class, **each accompanied by an object mask**. The latter is a binary image indicating whether each pixel is inside the object or not.

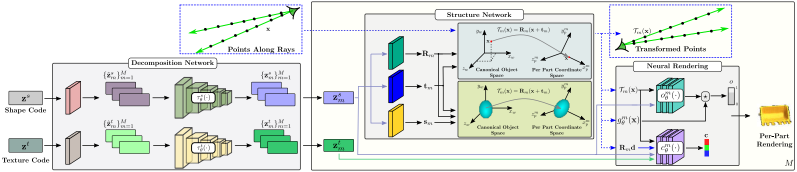


We implement our generative model as an **auto-decoder** that consists of:

- **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 the scale for each part m.
- **Neural Rendering Network:** Renders a 2D image using M locally defined NeRFs.

Object Generation

We are given a **collection of posed 2D images of objects** in a semantic class, **each accompanied by an object mask**. The latter is a binary image indicating whether each pixel is inside the object or not.

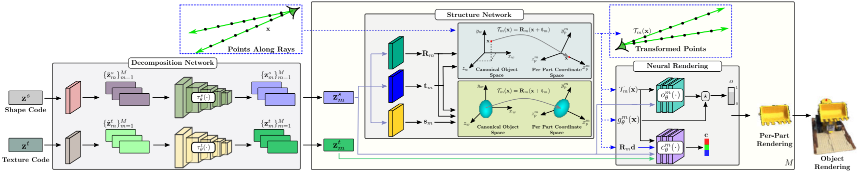


We implement our generative model as an **auto-decoder** that consists of:

- **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 the scale for each part m.
- **Neural Rendering Network:** Renders a 2D image using M locally defined NeRFs.

Object Generation

We are given a **collection of posed 2D images of objects** in a semantic class, **each accompanied by an object mask**. The latter is a binary image indicating whether each pixel is inside the object or not.



We implement our generative model as an **auto-decoder** that consists of:

- **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 the scale for each part m.
- **Neural Rendering Network:** Renders a 2D image using M locally defined NeRFs.

Optimization Objective

Our optimization objective $\mathcal{L}$ is the sum over six terms combined with two regularizers on the shape and texture embeddings $\mathbf{z}^s, \mathbf{z}^t$, namely

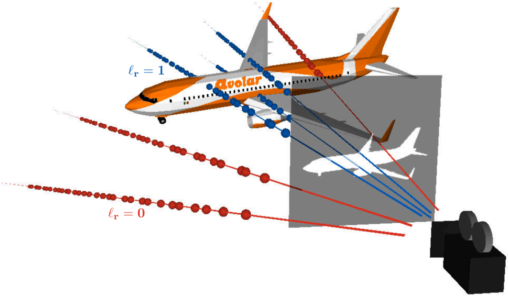
$$\mathcal{L} = \mathcal{L}_{rgb}(\mathcal{R}) + \mathcal{L}_{mask}(\mathcal{R}) + \mathcal{L}_{occ}(\mathcal{R}) + \mathcal{L}_{cov}(\mathcal{R}) \ + \mathcal{L}_{overlap}(\mathcal{R}) + \mathcal{L}_{control} + \left\|\mathbf{z}^s\right\|_2 + \left\|\mathbf{z}^t\right\|_2 .$$

Optimization Objective

Our optimization objective $\mathcal{L}$ is the sum over six terms combined with two regularizers on the shape and texture embeddings $\mathbf{z}^s, \mathbf{z}^t$, namely

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

As supervision, we use the **observed RGB color** $C(r) \in \mathbb{R}^3$ and the **object mask** $I(r) \in \{0, 1\}$ for each ray $r \in \mathcal{R}$. We also associate r with a binary label $\ell_r = I(r)$, indicating whether a ray r is *inside*, $(\ell_r = 1)$ or *outside* $(\ell_r = 0)$.



Optimization Objective

Our optimization objective $\mathcal{L}$ is the sum over six terms combined with two regularizers on the shape and texture embeddings $\mathbf{z}^s, \mathbf{z}^t$, namely

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

As supervision, we use the **observed RGB color** $C(r) \in \mathbb{R}^3$ and the **object mask** $I(r) \in \{0, 1\}$ for each ray $r \in \mathcal{R}$. We also associate r with a binary label $\ell_r = I(r)$, indicating whether a ray r is *inside*, $(\ell_r = 1)$ or *outside* $(\ell_r = 0)$.

- **Reconstruction Loss**: The rendered and the observed images should match.

Optimization Objective

Our optimization objective $\mathcal{L}$ is the sum over six terms combined with two regularizers on the shape and texture embeddings $\mathbf{z}^s, \mathbf{z}^t$, namely

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

As supervision, we use the **observed RGB color** $C(r) \in \mathbb{R}^3$ and the **object mask** $I(r) \in \{0, 1\}$ for each ray $r \in \mathcal{R}$. We also associate r with a binary label $\ell_r = I(r)$, indicating whether a ray r is *inside*, $(\ell_r = 1)$ or *outside* $(\ell_r = 0)$.

- **Reconstruction Loss**: The rendered and the observed images should match.
- **Mask Loss**: The rendered and the observed object masks should match.

Optimization Objective

Our optimization objective $\mathcal{L}$ is the sum over six terms combined with two regularizers on the shape and texture embeddings $\mathbf{z}^s, \mathbf{z}^t$, namely

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

As supervision, we use the **observed RGB color** $C(r) \in \mathbb{R}^3$ and the **object mask** $I(r) \in \{0,1\}$ for each ray $r \in \mathcal{R}$. We also associate r with a binary label $\ell_r = I(r)$, indicating whether a ray r is *inside*, $(\ell_r = 1)$ or *outside* $(\ell_r = 0)$.

- **Reconstruction Loss**: The rendered and the observed images should match.
- **Mask Loss**: The rendered and the observed object masks should match.
- **Occupancy Loss**: The generated shape should not occupy empty space.

Optimization Objective

Our optimization objective $\mathcal{L}$ is the sum over six terms combined with two regularizers on the shape and texture embeddings $\mathbf{z}^s, \mathbf{z}^t$, namely

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

As supervision, we use the **observed RGB color** $C(r) \in \mathbb{R}^3$ and the **object mask** $I(r) \in \{0, 1\}$ for each ray $r \in \mathcal{R}$. We also associate r with a binary label $\ell_r = I(r)$, indicating whether a ray r is *inside*, $(\ell_r = 1)$ or *outside* $(\ell_r = 0)$.

- **Reconstruction Loss**: The rendered and the observed images should match.
- **Mask Loss**: The rendered and the observed object masks should match.
- **Occupancy Loss**: The generated shape should not occupy empty space.
- **Coverage Loss**: Prevent degenerate part arrangements.

Optimization Objective

Our optimization objective $\mathcal{L}$ is the sum over six terms combined with two regularizers on the shape and texture embeddings $\mathbf{z}^s, \mathbf{z}^t$, namely

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

As supervision, we use the **observed RGB color** $C(r) \in \mathbb{R}^3$ and the **object mask** $I(r) \in \{0, 1\}$ for each ray $r \in \mathcal{R}$. We also associate r with a binary label $\ell_r = I(r)$, indicating whether a ray r is *inside*, $(\ell_r = 1)$ or *outside* $(\ell_r = 0)$.

- **Reconstruction Loss**: The rendered and the observed images should match.
- **Mask Loss**: The rendered and the observed object masks should match.
- **Occupancy Loss**: The generated shape should not occupy empty space.
- **Coverage Loss**: Prevent degenerate part arrangements.
- **Overlapping Loss**: Prevent overlapping parts.

Optimization Objective

Our optimization objective $\mathcal{L}$ is the sum over six terms combined with two regularizers on the shape and texture embeddings $\mathbf{z}^s, \mathbf{z}^t$, namely

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

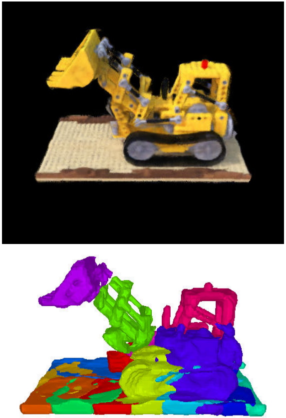
As supervision, we use the **observed RGB color** $C(r) \in \mathbb{R}^3$ and the **object mask** $I(r) \in \{0, 1\}$ for each ray $r \in \mathcal{R}$. We also associate r with a binary label $\ell_r = I(r)$, indicating whether a ray r is *inside*, $(\ell_r = 1)$ or *outside* $(\ell_r = 0)$.

- **Reconstruction Loss**: The rendered and the observed images should match.
- **Mask Loss**: The rendered and the observed object masks should match.
- **Occupancy Loss**: The generated shape should not occupy empty space.
- **Coverage Loss**: Prevent degenerate part arrangements.
- **Overlapping Loss**: Prevent overlapping parts.
- **Control Loss**: Ensure uniform control across the shape.

How well does it work?

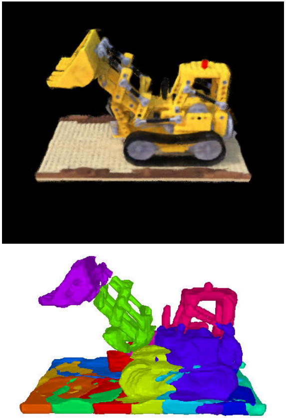
Scene-Specific Editing

No Editing

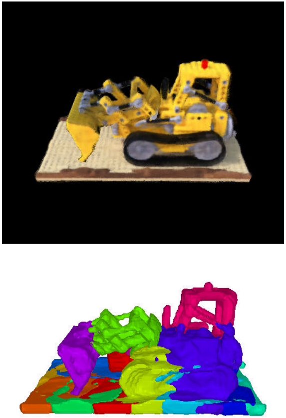


Scene-Specific Editing

No Editing

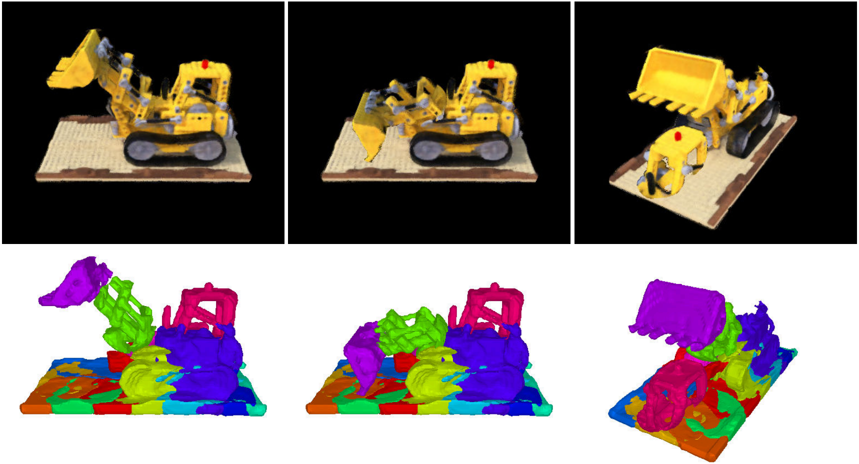


Rotation



Scene-Specific Editing

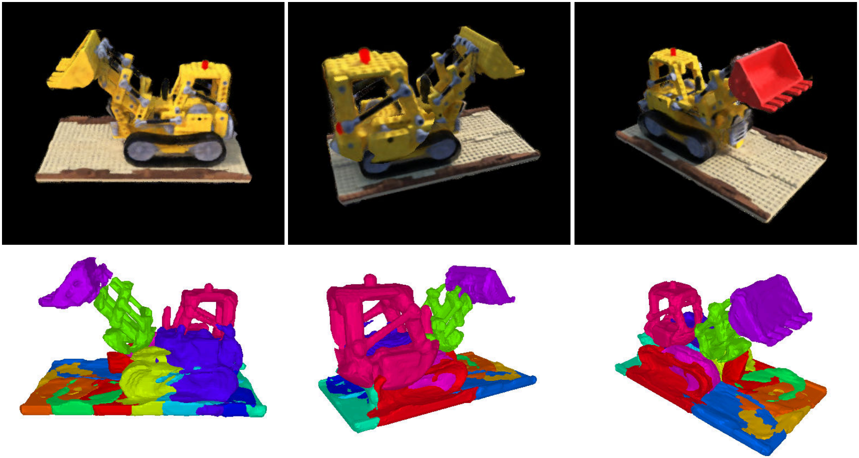
No Editing | Rotation | Translation



During all editing operations, **only a specific parts of the object changes**, while the rest do not change.

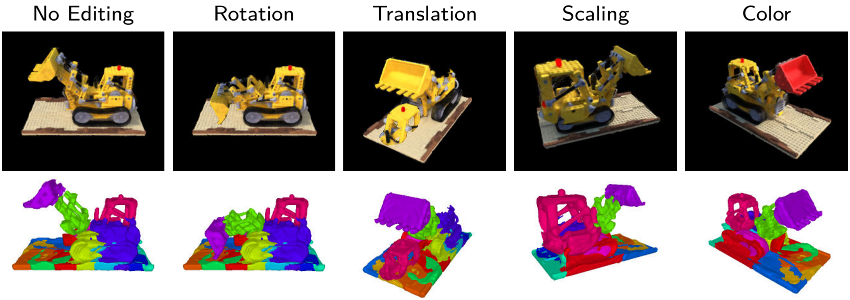
Scene-Specific Editing

No Editing | Scaling | Color

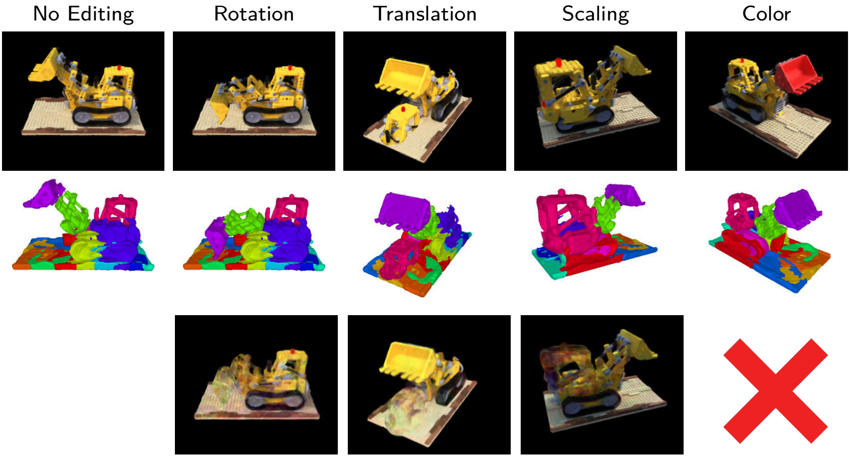


During all editing operations, **only a specific parts of the object changes**, while the rest do not change.

Impact of Hard Ray-Part Assignment



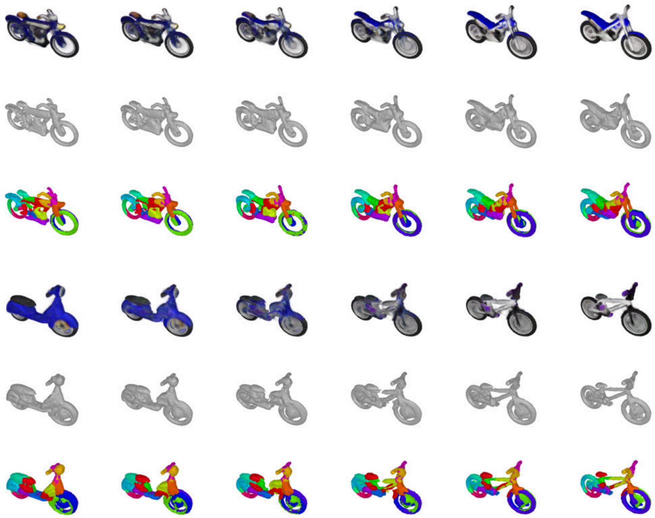
Impact of Hard Ray-Part Assignment



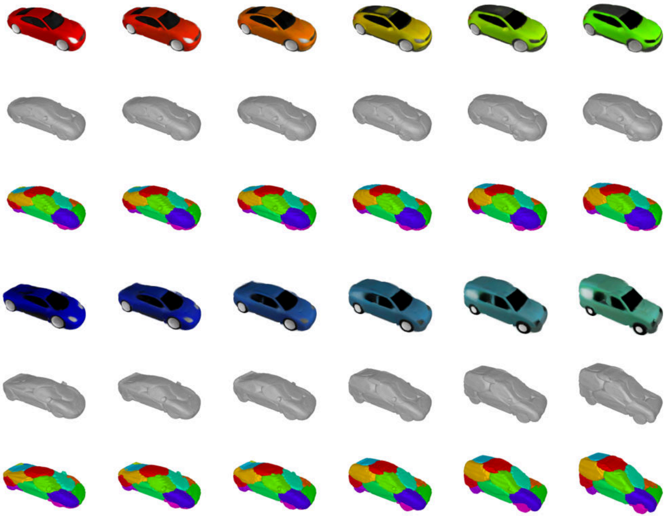
The **hard ray-part assignment** enforces that the color of a ray is determined by **a single NeRF/part**, hence transforming one part **does not alter the other parts**.

Shape Synthesis

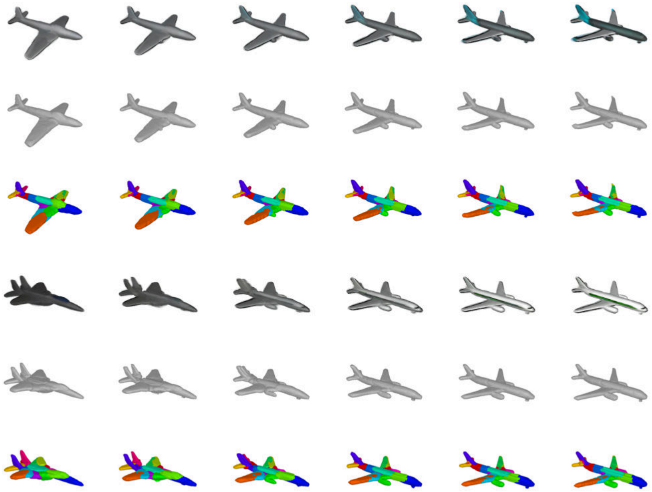
Shape Interpolations



Shape Interpolations

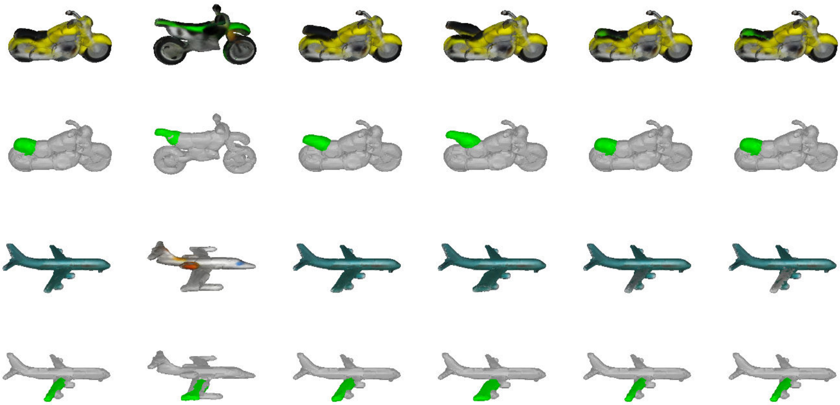


Shape Interpolations

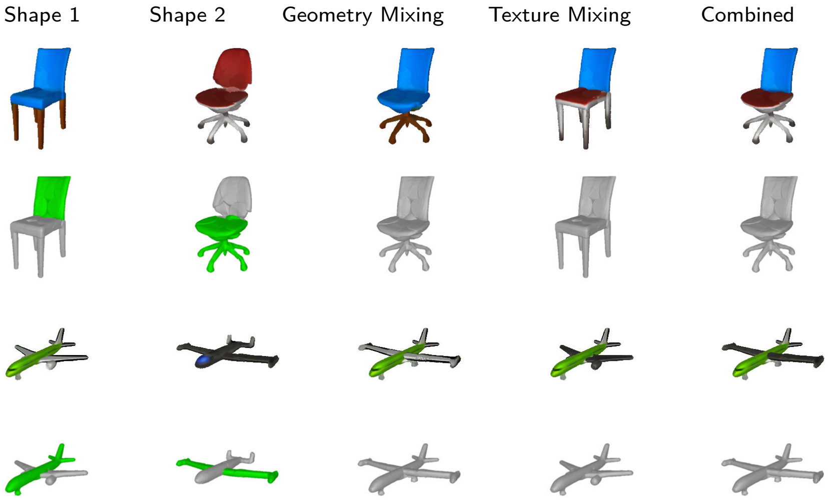


Part Interpolation

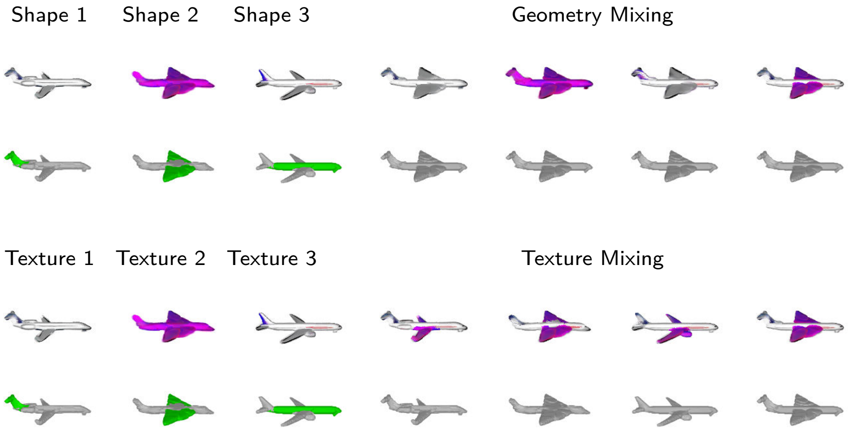
Shape 1 | Shape 2 | Shape Code Interpolation | Texture Code Interpolation



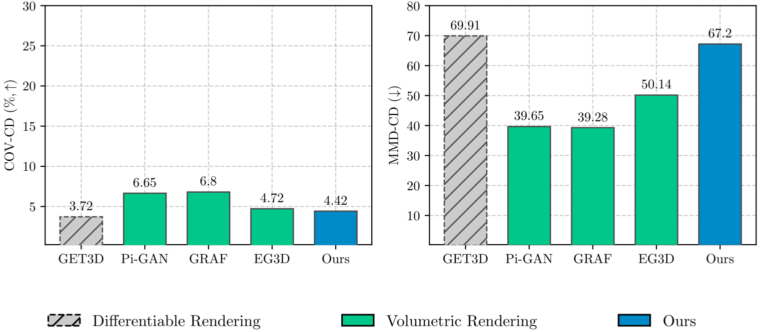
Shape Mixing



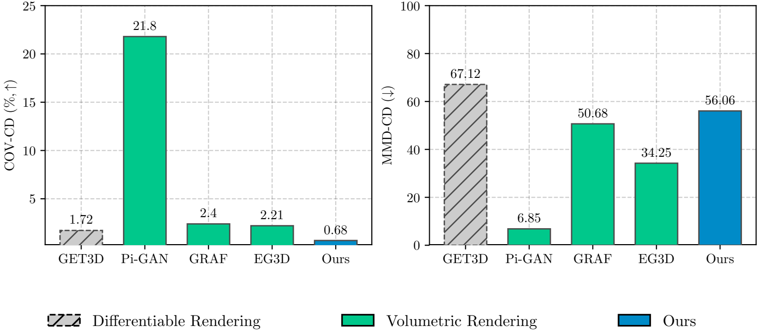
Shape Mixing



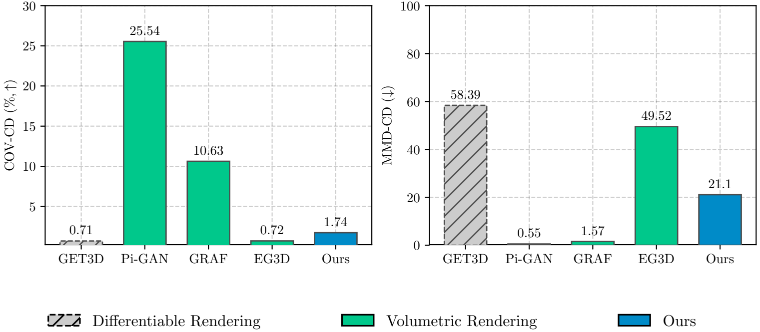
ShapeNet Comparison - Chairs



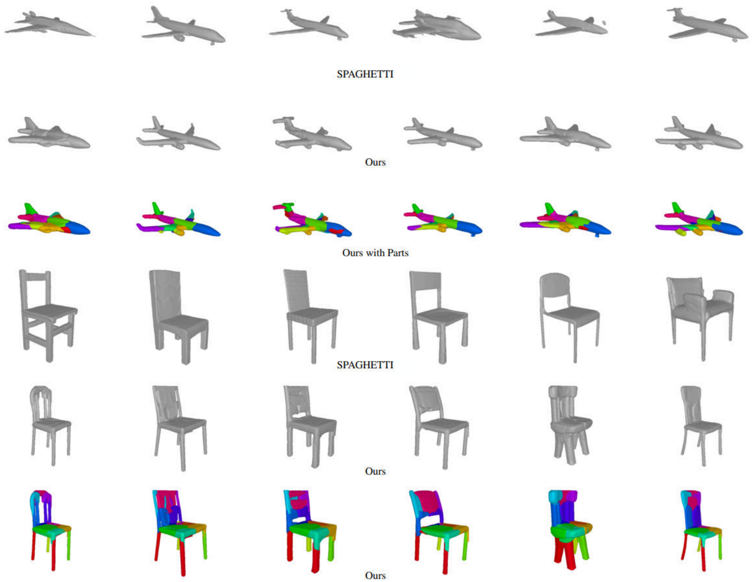
ShapeNet Comparison - Motorbikes



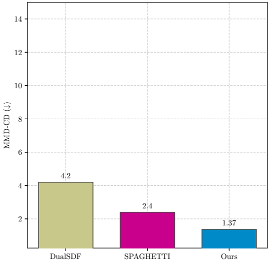
ShapeNet Comparison - Cars



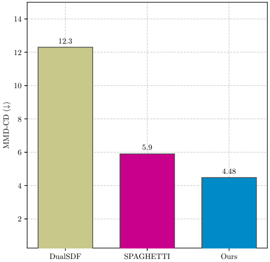
ShapeNet Comparison to Part-based Methods



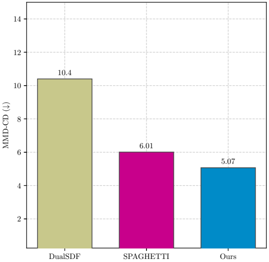
ShapeNet Comparison - Part-based Methods



Airplanes

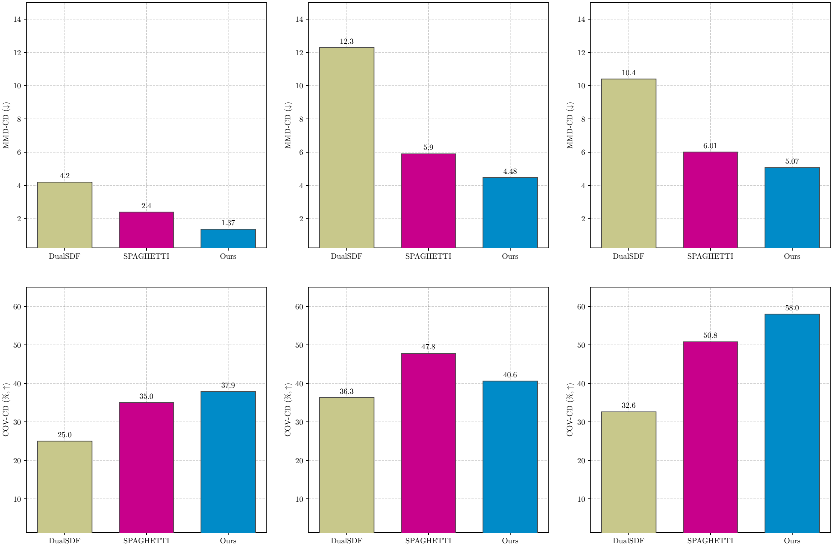


Tables



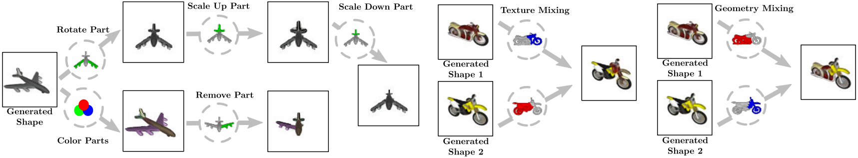
Chairs

ShapeNet Comparison - Part-based Methods



Airplanes Tables Chairs

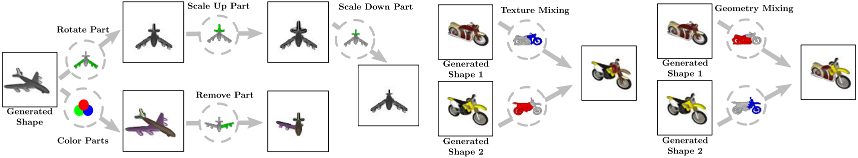
Summary and Limitations



- We introduced the **first part-aware generative model that parametrizes parts as NeRFs**.

	Representation	Supervision	Parts	Shape Editing	Texture Editing	Mixing
GET3D	Mesh	2D	✗	✗	✗	✗
GRAF	Neural Field	2D	✗	✗	✗	✗
Pi-GAN			✗	✗	✗	✗
EG3D			✗	✗	✗	✗
DualSDF	Implicit	3D	✓	✓	✗	✗
SPAGHETTI			✓	✓	✗	✓
PartNeRF	Neural Field	2D	✓	✓	✓	✓

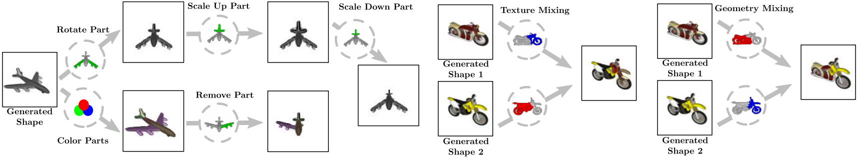
Summary and Limitations



- We introduced the **first part-aware generative model that parametrizes parts as NeRFs**.
- As our model considers the decomposition of objects into parts, it **enables intuitive part-level control** and several editing operations not previously possible.

	Representation	Supervision	Parts	Shape Editing	Texture Editing	Mixing
GET3D	Mesh	2D	✗	✗	✗	✗
GRAF	Neural Field	2D	✗	✗	✗	✗
Pi-GAN			✗	✗	✗	✗
EG3D			✗	✗	✗	✗
DualSDF	Implicit	3D	✓	✓	✗	✗
SPAGHETTI			✓	✓	✗	✓
PartNeRF	Neural Field	2D	✓	✓	✓	✓

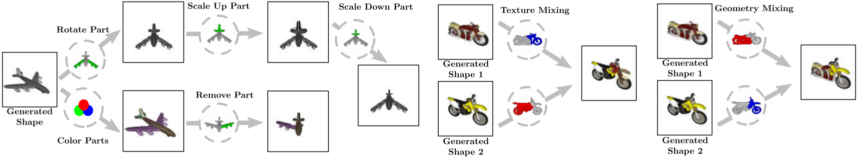
Summary and Limitations



- We introduced the **first part-aware generative model that parametrizes parts as NeRFs**.
- As our model considers the decomposition of objects into parts, it **enables intuitive part-level control** and several editing operations not previously possible.
- Our model is **trained without explicit 3D supervision**, using only posed images and object masks.

	Representation	Supervision	Parts	Shape Editing	Texture Editing	Mixing
GET3D	Mesh	2D	✗	✗	✗	✗
GRAF Pi-GAN EG3D	Neural Field	2D	✗ ✗ ✗	✗ ✗ ✗	✗ ✗ ✗	✗ ✗ ✗
DualSDF SPAGHETTI	Implicit	3D	✓ ✓	✓ ✓	✗ ✗	✗ ✓
PartNeRF	Neural Field	2D	✓	✓	✓	✓

Summary and Limitations

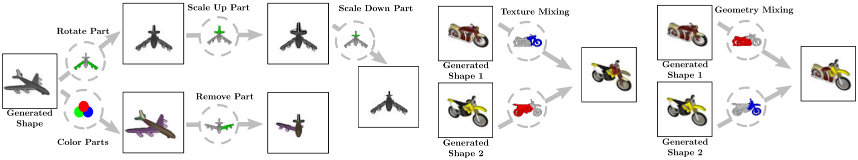


- We introduced the **first part-aware generative model that parametrizes parts as NeRFs**.
- As our model considers the decomposition of objects into parts, it **enables intuitive part-level control** and several editing operations not previously possible.
- Our model is **trained without explicit 3D supervision**, using only posed images and object masks.

Limitations:

- Considering GAN losses or triplane representations could further improve the quality of our generated textures.

Summary and Limitations



- We introduced the **first part-aware generative model that parametrizes parts as NeRFs**.
- As our model considers the decomposition of objects into parts, it **enables intuitive part-level control** and several editing operations not previously possible.
- Our model is **trained without explicit 3D supervision**, using only posed images and object masks.

Limitations:

- Considering GAN losses or triplane representations could further improve the quality of our generated textures.
- The generated parts are not necessarily interpretable.

Thank you for your attention!