SceneScript

A **novel scene representation** using an autoregressive *structured* language model.

Ben Xia & Stone Yang

Problem Statement

[Layout Estimation]

Given **scene data** → Create a “**map**”

“map”

[Layout Estimation]

- Ge
- Lo
- Se

Useful for
AR, Robotics, BIM Applications+!

BIM: Building Information Modeling

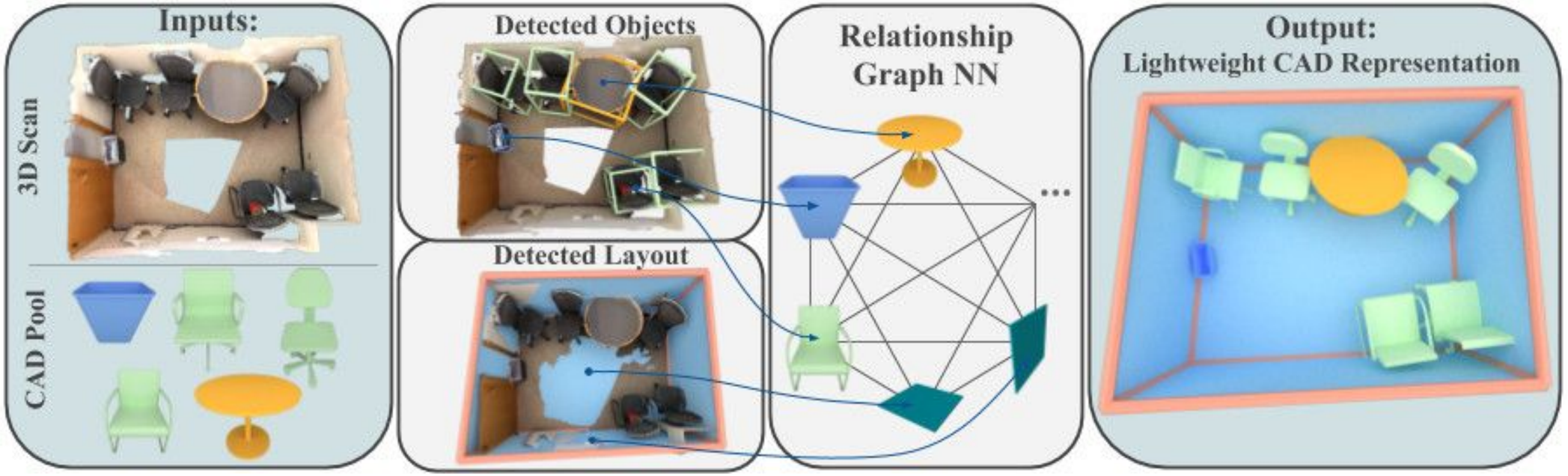
Sneak peak of SceneScript!

Prior Work: SceneCAD ('20)

[Layout Estimation]

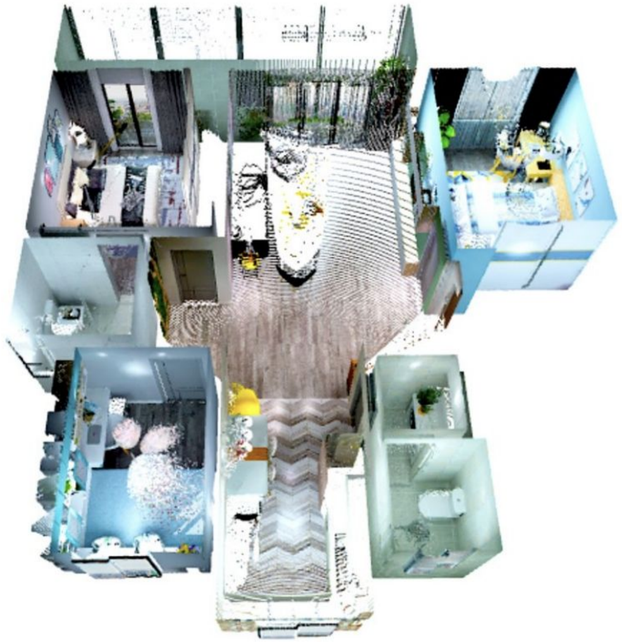
Key: GNN *jointly optimizes* CAD alignment + layout

- Handcrafted Geometric Priors (aligns *existing* CAD models)
- Non-extensible (predicts quad planes *only*)

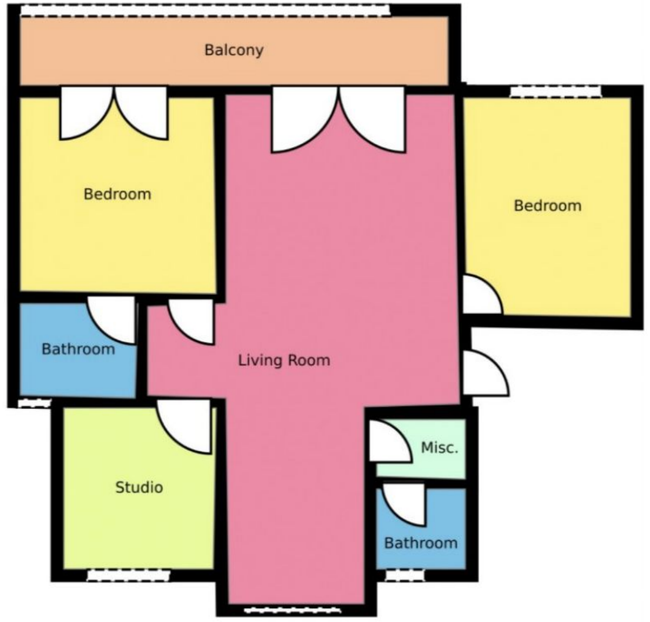


Prior Work: RoomFormer (’23)

[Layout Estimation]



RoomFormer



Input: 3D Scan

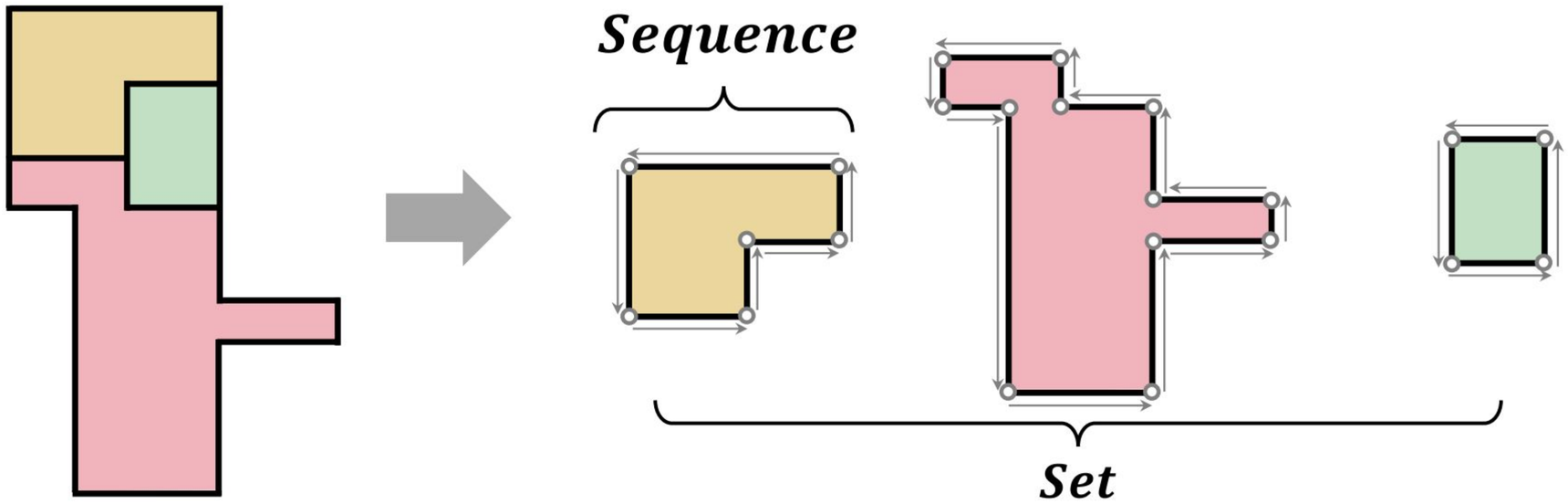
Output: Semantic Floorplan

Prior Work: RoomFormer

[Layout Estimation]

- 2D layouts ONLY
- doesn't consider objects

Key Idea: **Formulate floorplan reconstruction as a direct set prediction problem of polygons.**

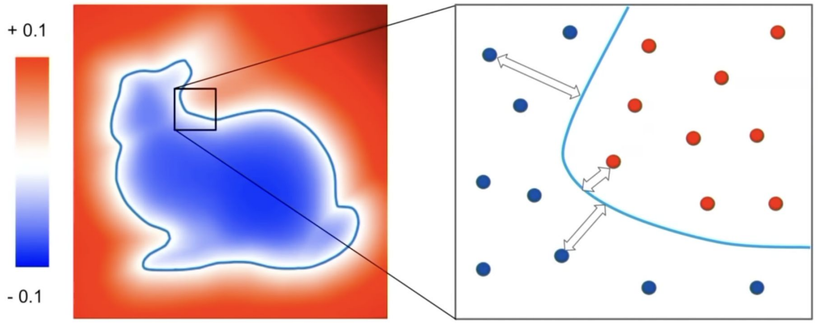


Transformer Encoder-Decoder!

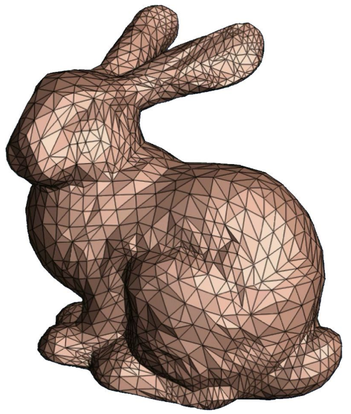
SceneScript (’24)

[Motivations]

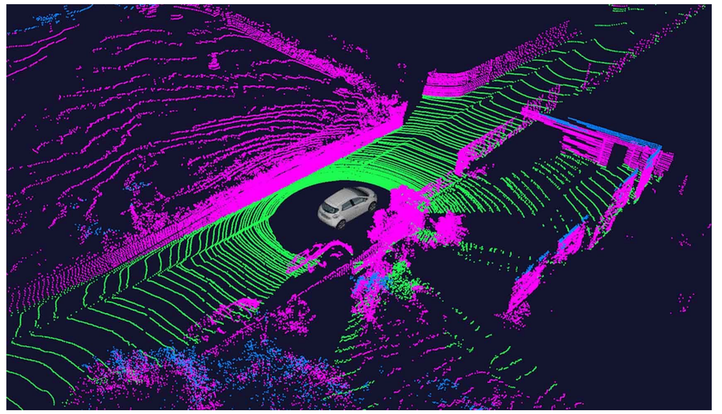
- No assumed geometric priors
- 3D-native
- Memory efficient!
- Semantically rich!
- Interpretable & Extensible!



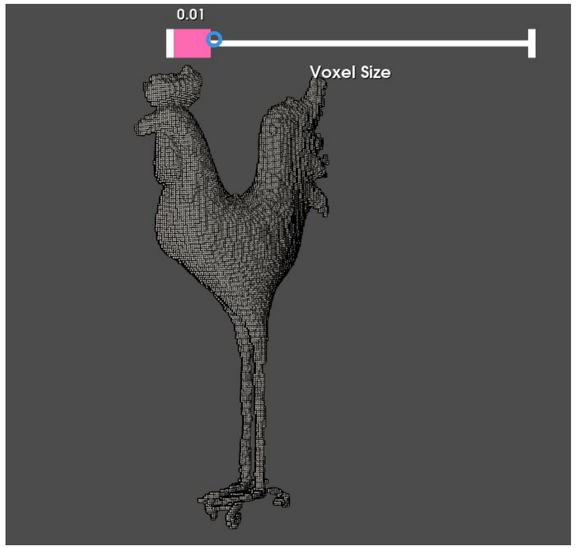
Implicit Functions



Mesh



Pointcloud



Voxel Grid

SceneScript (’24)

[Key Idea]

SceneScript

[Key Idea]

SceneScript

[Key Idea]

This is ~100 **bytes**!

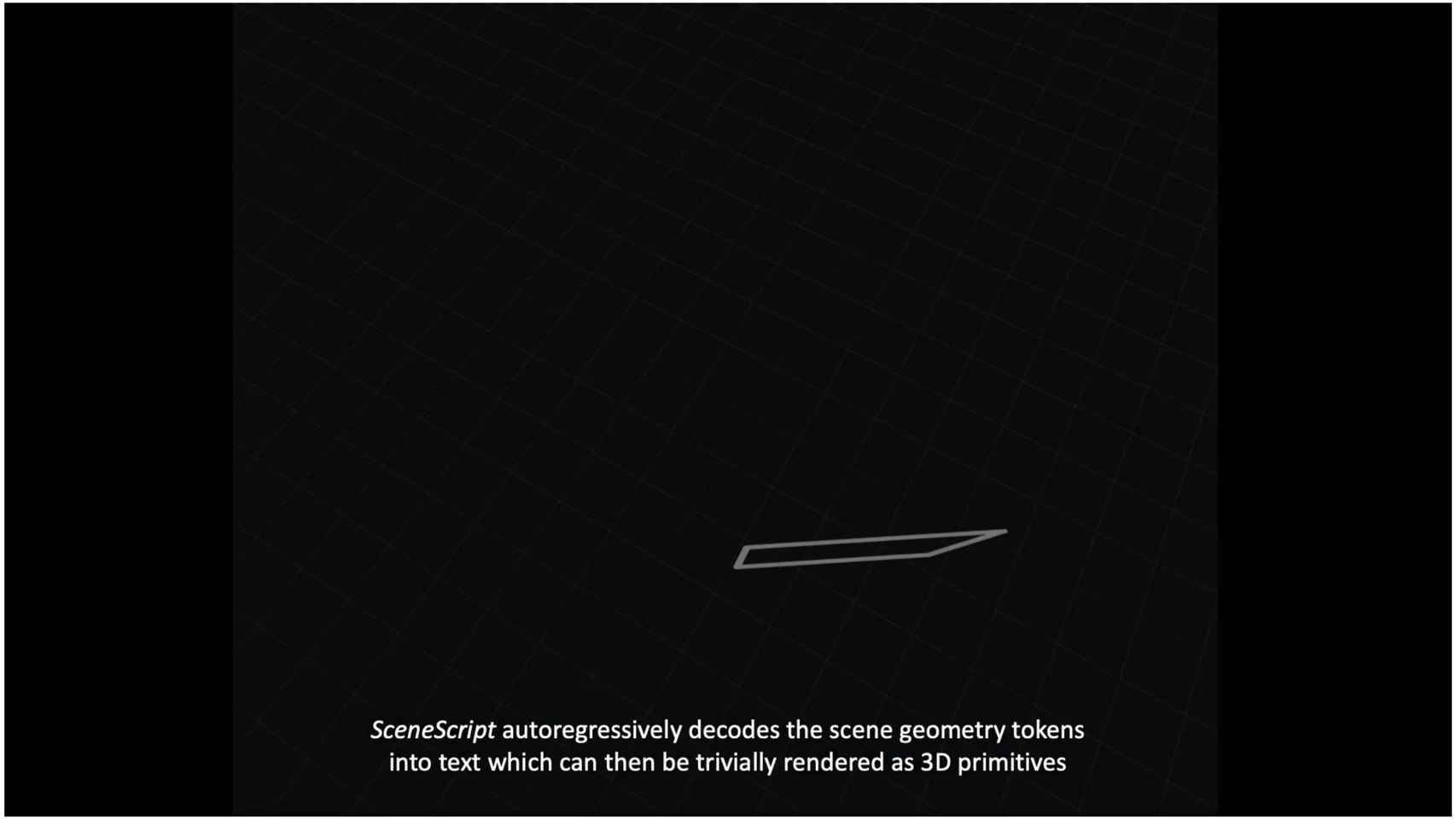
```
- make_wall:
    id: 1
    a_x: 1.0
    a_y: -2.83
    a_z: -1.62
    b_x: 1.0
    b_y: 1.0
    b_z: 1.0
```

SceneScript

[Key Idea]

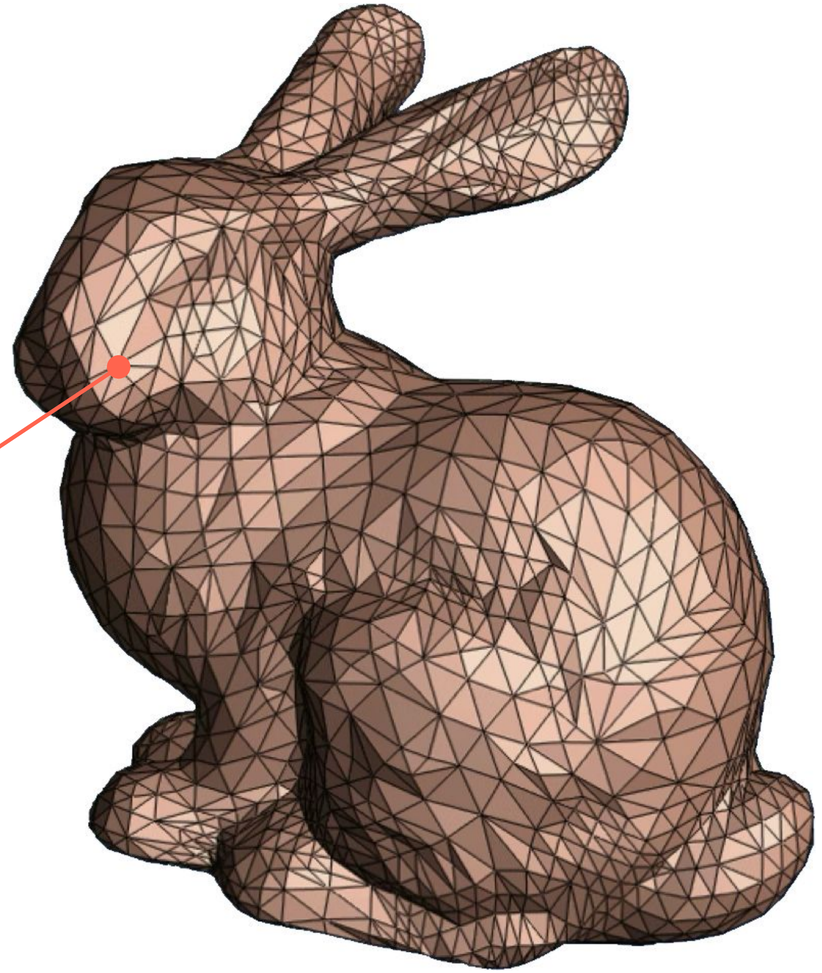
```
- make_wall:
    id: 1
    a_x: 1.0
    a_y: -2.83
    a_z: -1.62
    b_x: 1.0
    b_y: 1.0
    b_z: 1.0

- make_sofa:
    id: 2
    pos_x:
    pos_y: 0.0
    pos_z: 0.0
    rot_x: 0.0
    ...
```



Meshes

+ detailed geometry



[x, y, z]

[u, v]

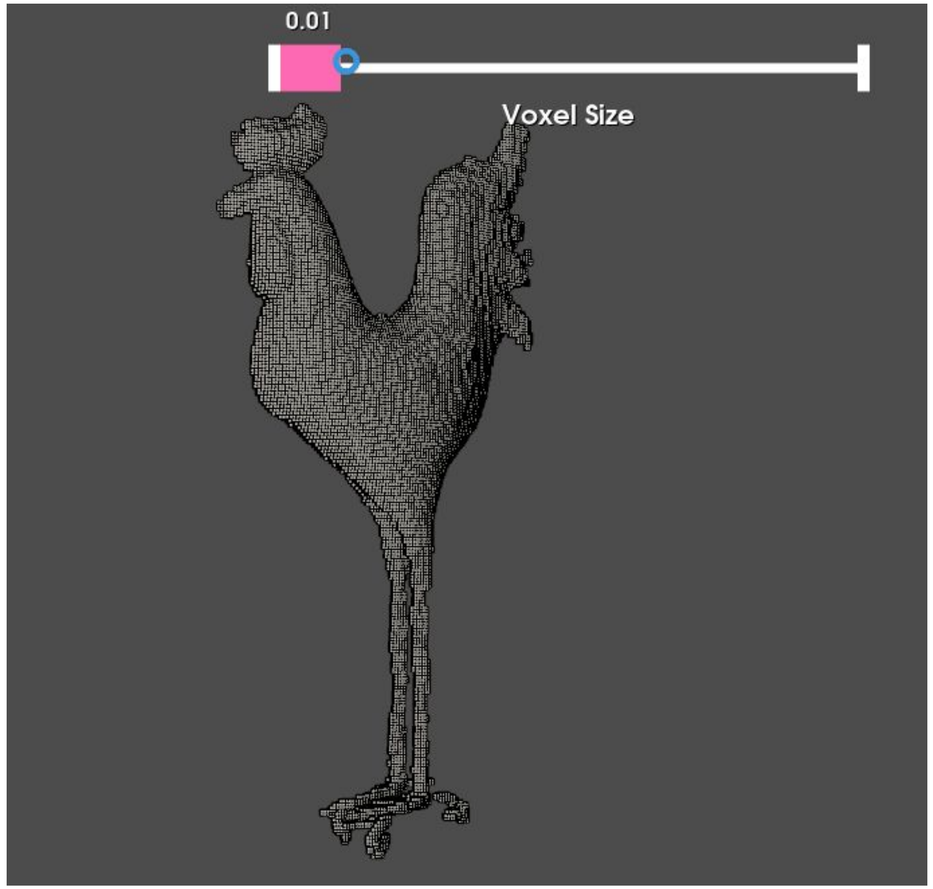
[n_x, n_y, n_z]

- storage*

- computation

Voxel Grids

+ volumetric



- discrete

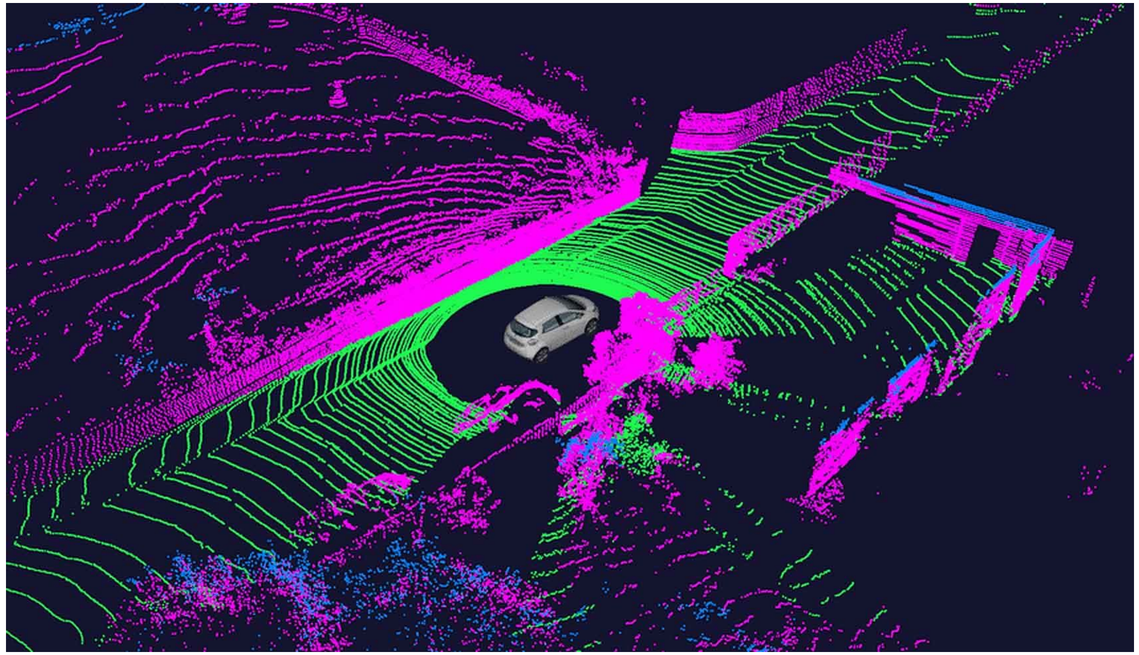
- ***storage***!

GIF from Ivan Nikolov

https://medium.com/data-science/how-to-voxelize-meshes-and-point-clouds-in-python-ca94d403f81d

Pointclouds

\+ efficient for sparse scenes



\- no connectivity semantics

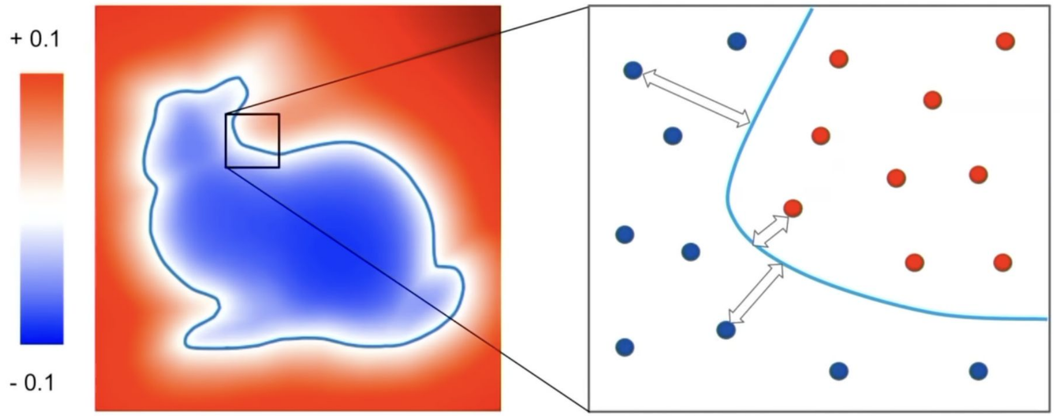
Image from Michael Abramov

https://keymakr.com/blog/how-3d-point-cloud-segmentation-will-make-the-future-hands-free/

Implicit Functions

+ Continuous

+ Compact



DeepSDF

- Lacks interpretability and edibility

- Compute

NeRF

SceneScript

Encoder-Decoder
Architecture


```
- make_wall:
    id: 1
    a_x: 1.0
    a_y: -2.83
    a_z: -1.62
    b_x: 1.0
    b_y: 1.0
    b_z: 1.0

- make_sofa:
    id: 2
    pos_x:
    pos_y: 0.0
    pos_z: 0.0
    rot_x: 0.0
    ...
```

This is ~100 **bytes**!

+ Extremely compact

+ Semantically-rich

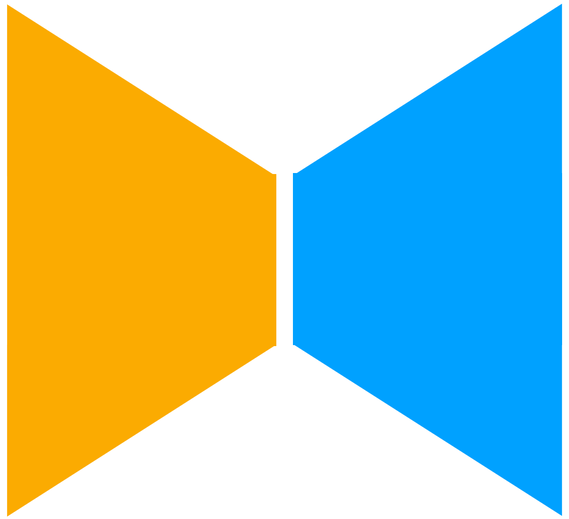
+ Interpretable & Extendible

+ sharp geometry

- Coarse

SceneScript Architecture

Encoder-Decoder Architecture

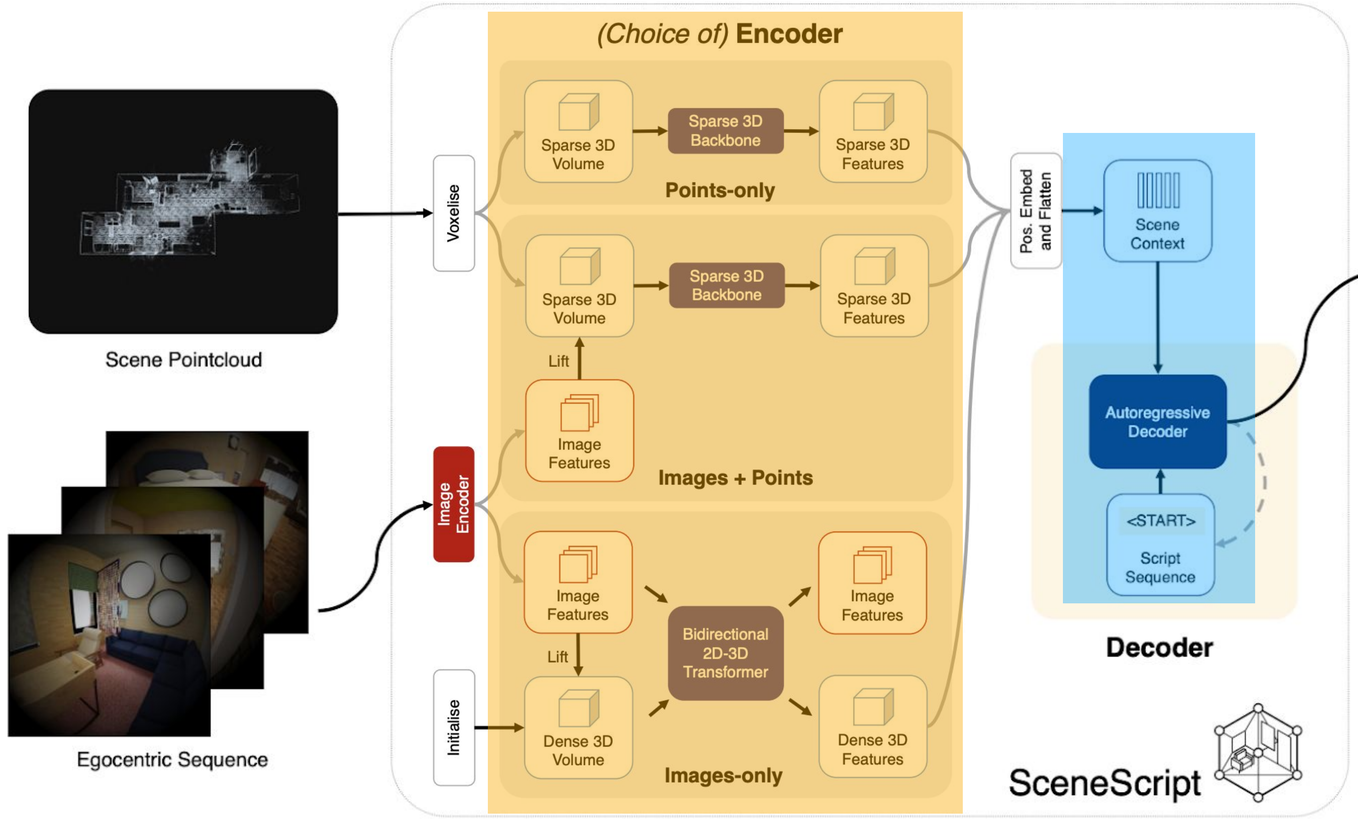


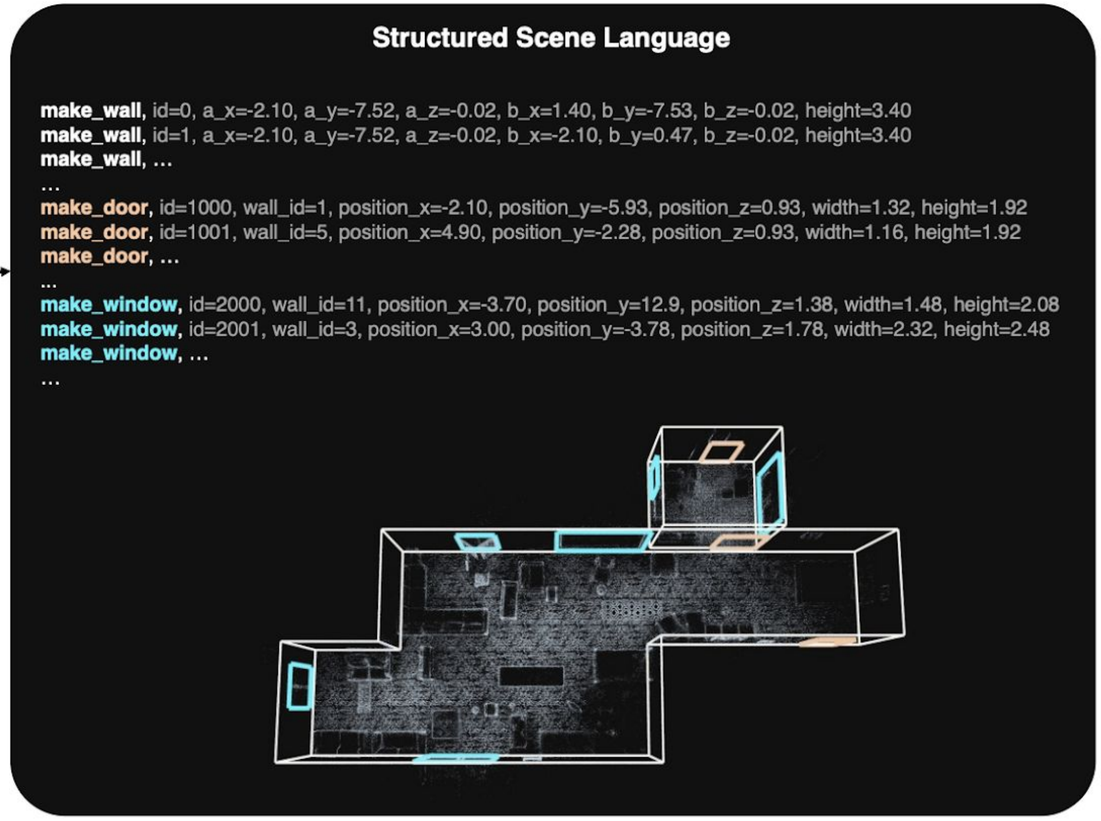
```
- make_wall:
    id: 1
    a_x: 1.0
    a_y: -2.83
    a_z: -1.62
    b_x: 1.0
    b_y: 1.0
    b_z: 1.0

- make_sofa:
    id: 2
    pos_x:
    pos_y: 0.0
    pos_z: 0.0
    rot_x: 0.0
    ...
```

Encoder-Decoder Architecture

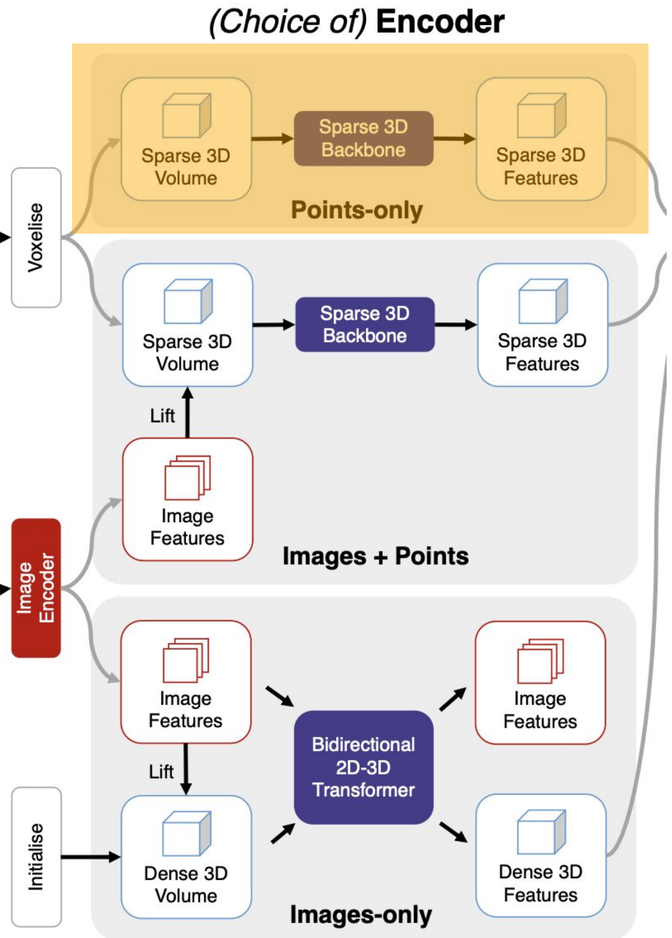
Architecture



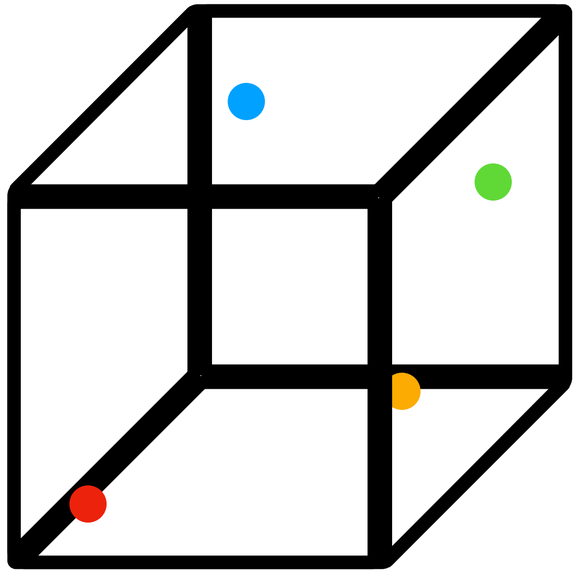


Architecture

Points-Only Encoder



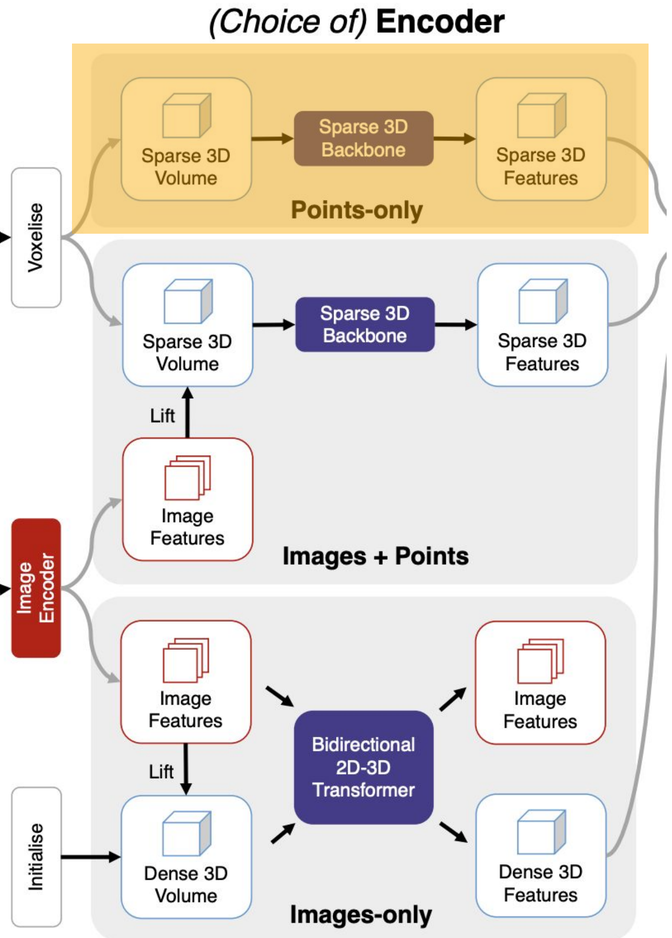
- Pointcloud is voxelized



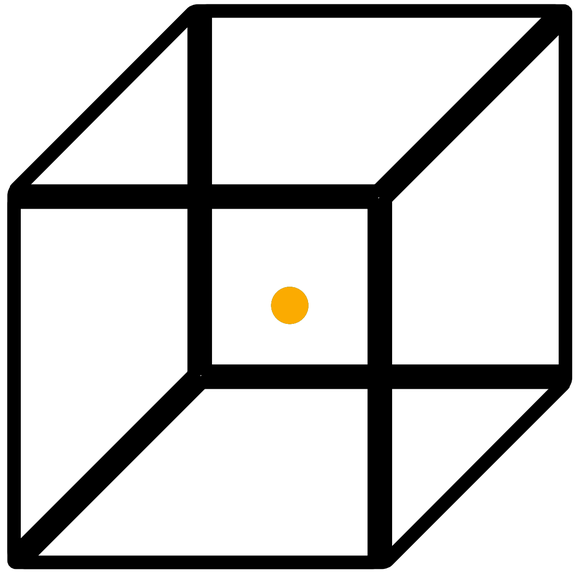
1 grid cell

Architecture

Points-Only Encoder



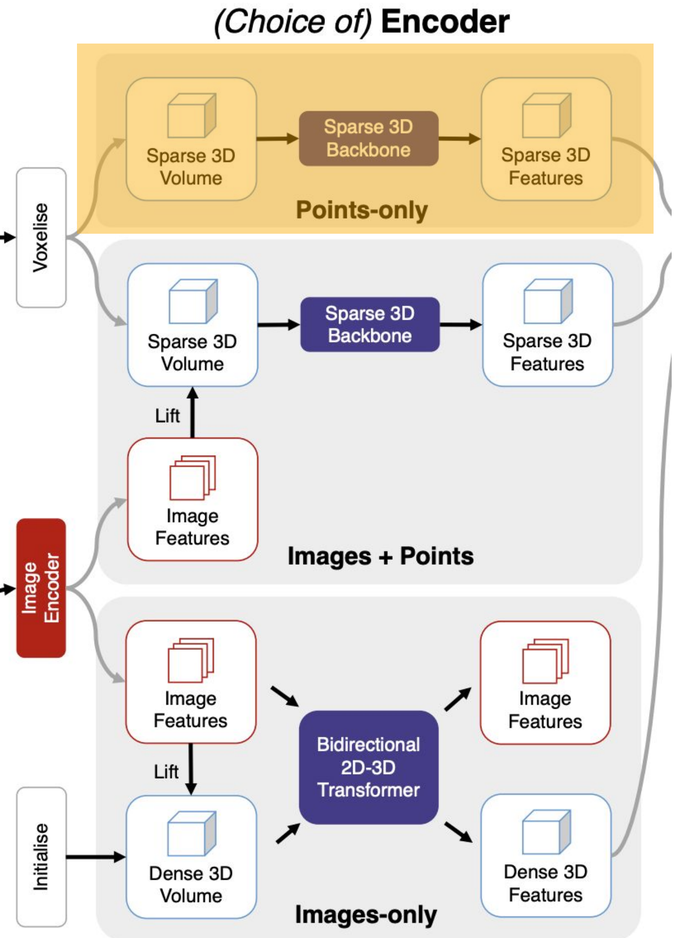
- Pointcloud is voxelized



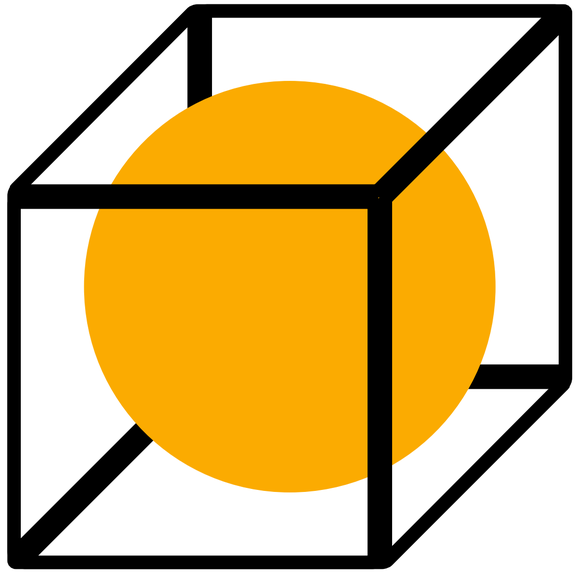
1 grid cell

Architecture

Points-Only Encoder



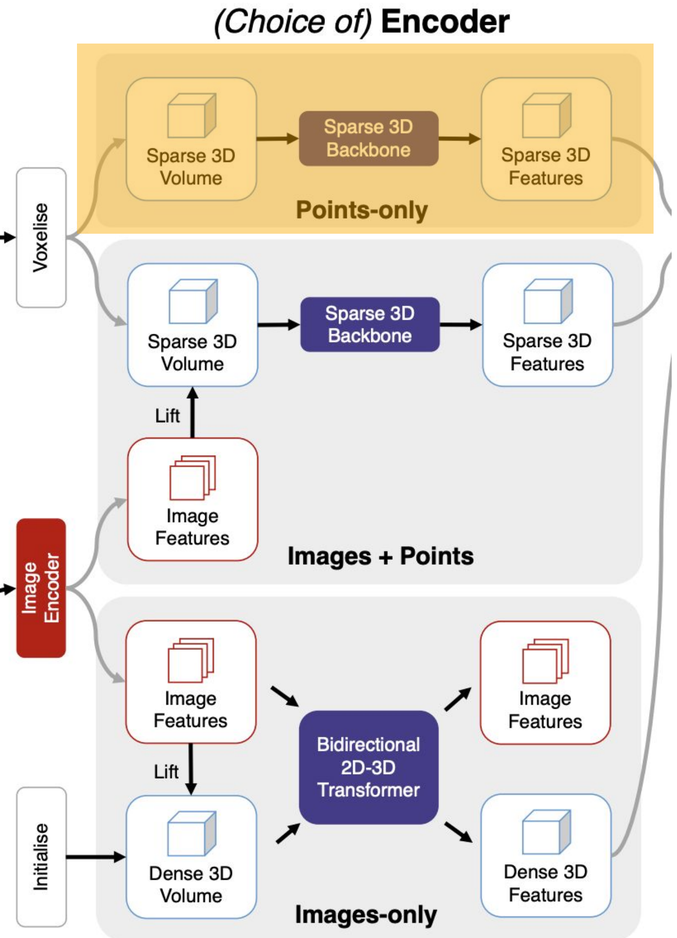
- Pointcloud is voxelized



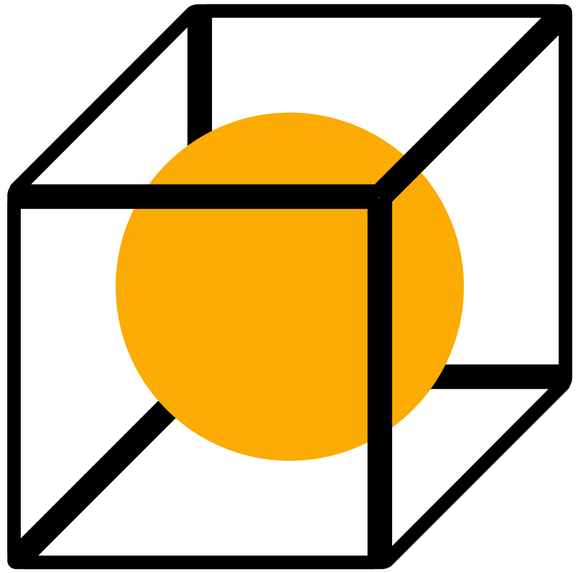
1 grid cell

Architecture

Points-Only Encoder



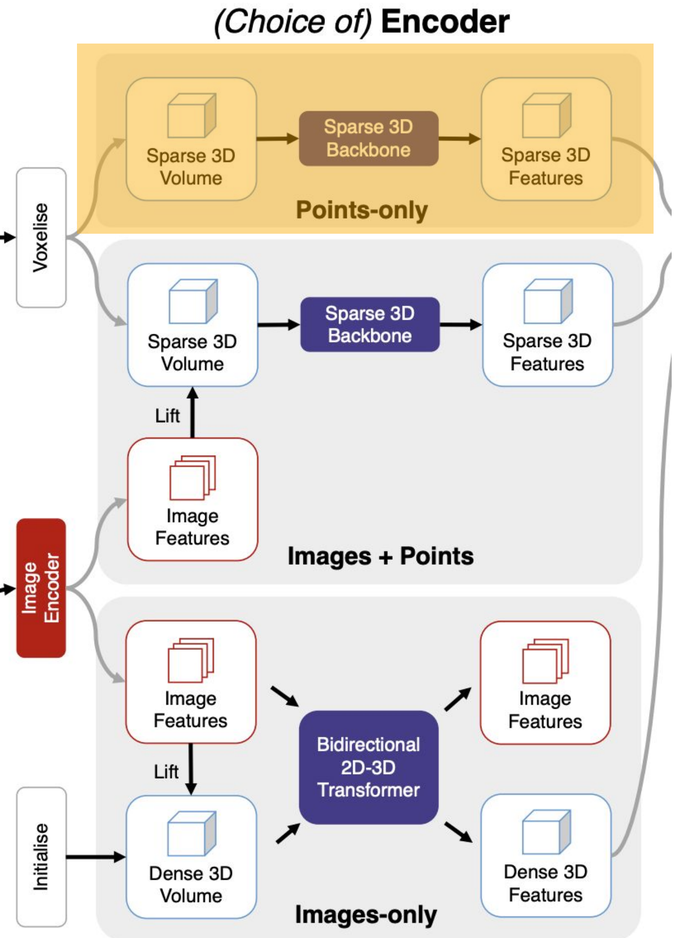
- Pointcloud is voxelized



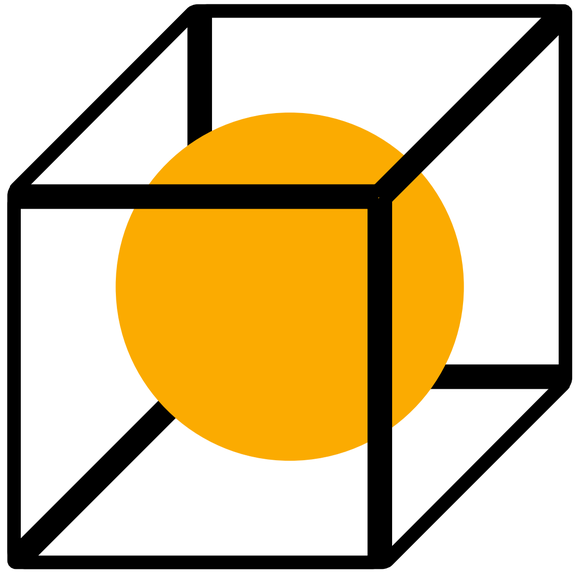
1 grid cell

Architecture

Points-Only Encoder



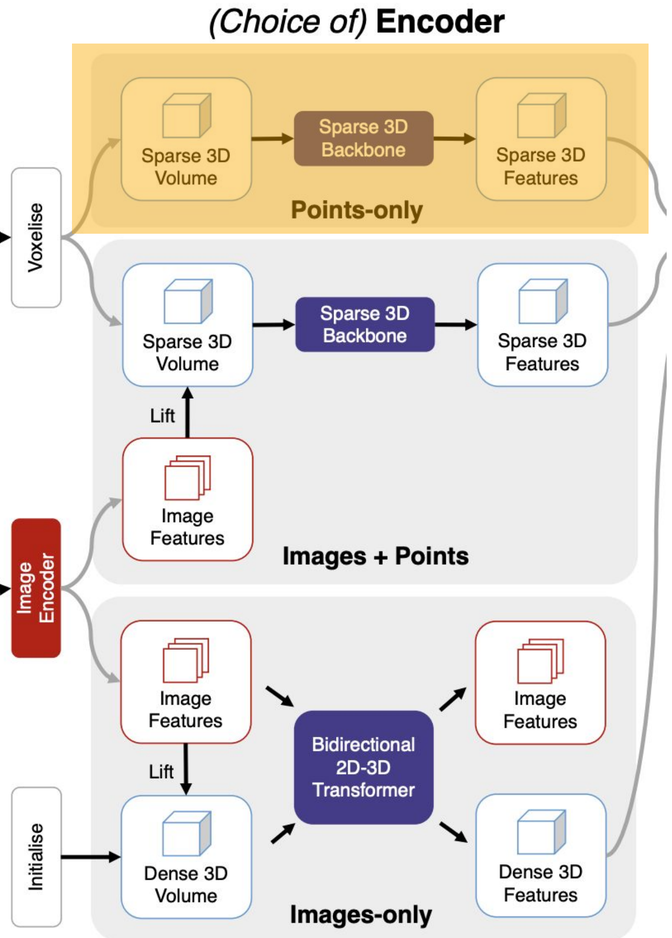
- Pointcloud is voxelized



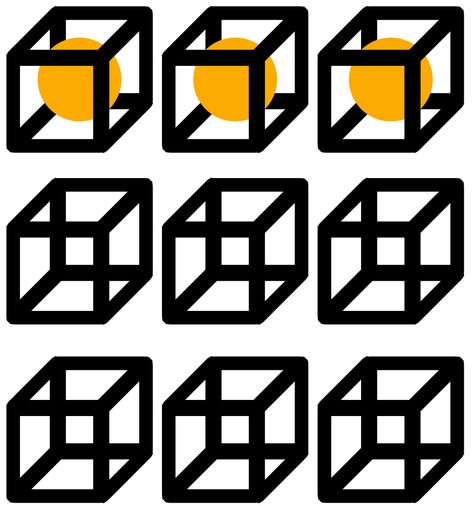
1 grid cell

Architecture

Points-Only Encoder

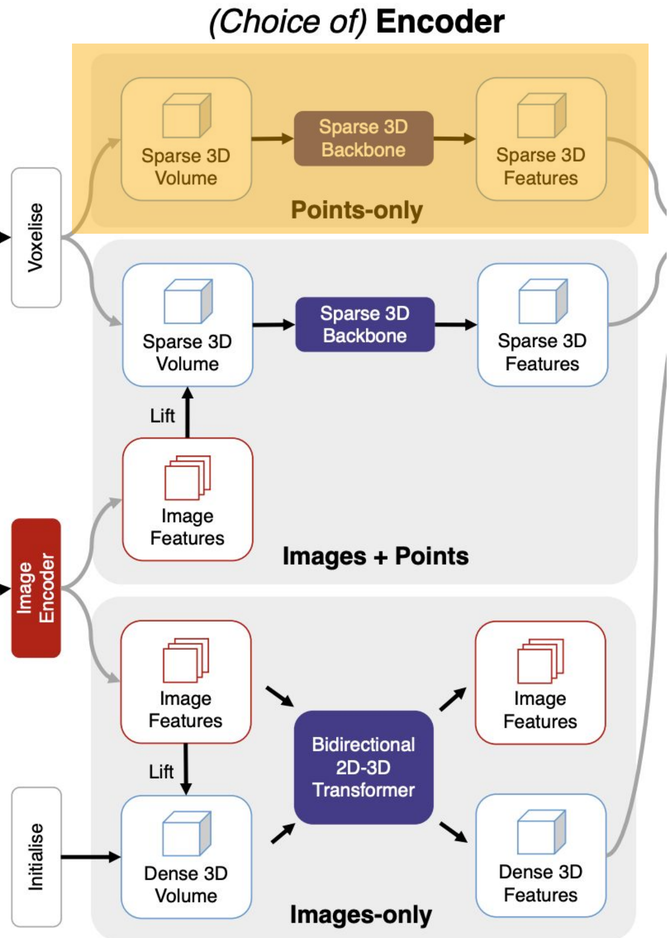


- Apply **sparse 3d convolution** to voxelized points

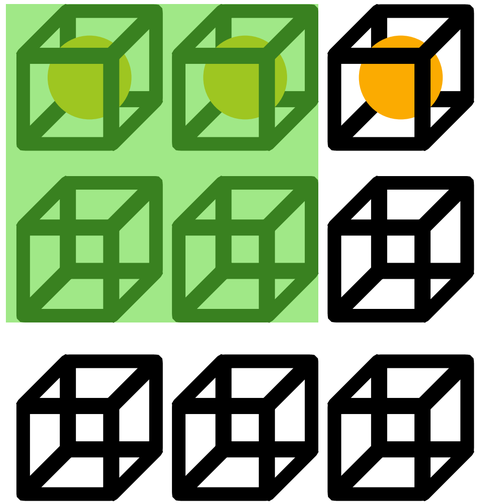


Architecture

Points-Only Encoder

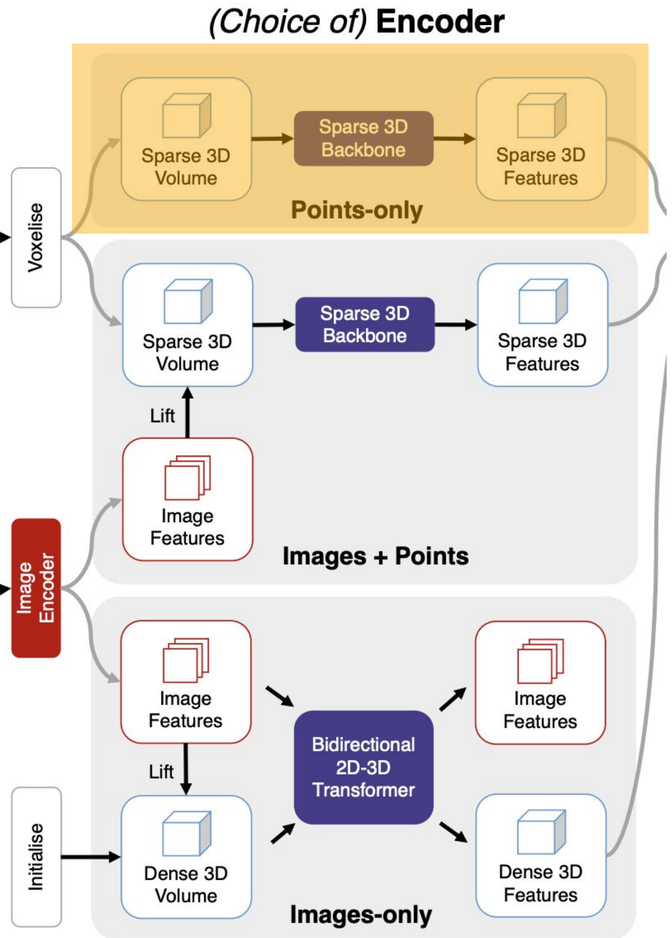


- Apply **sparse 3d convolution** to voxelized points

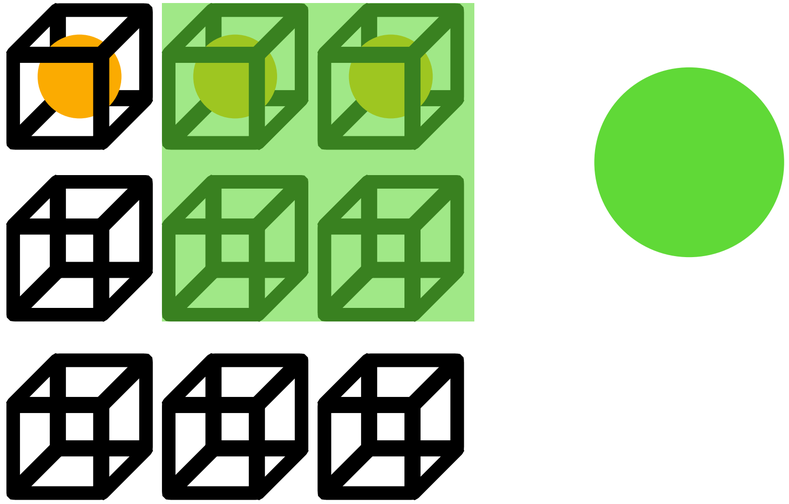


Architecture

Points-Only Encoder

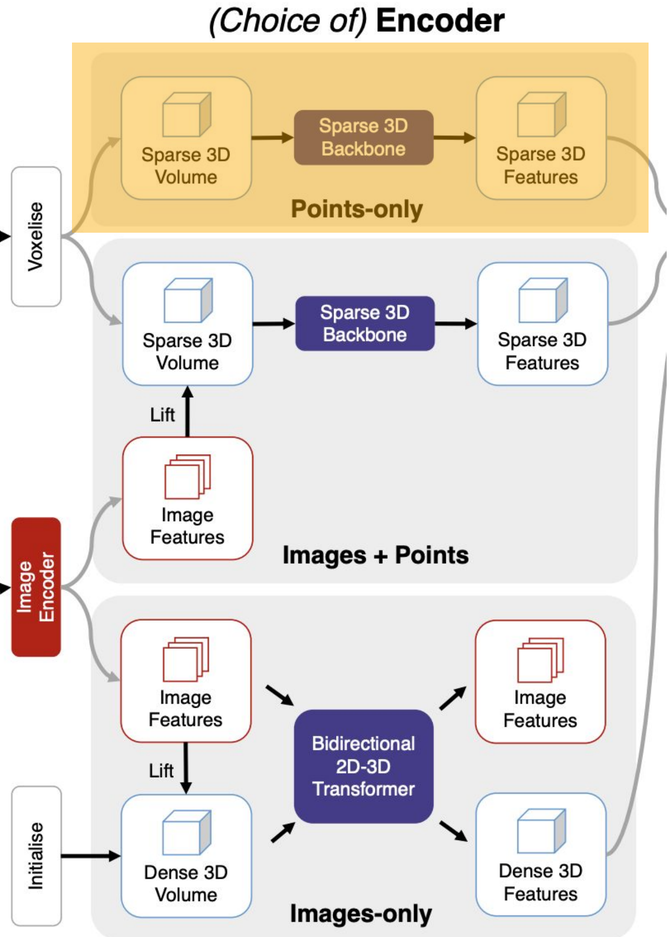


- Apply **sparse 3d convolution** to voxelized points



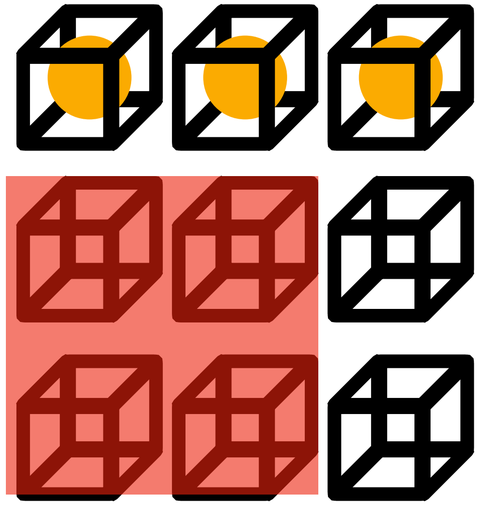
Architecture

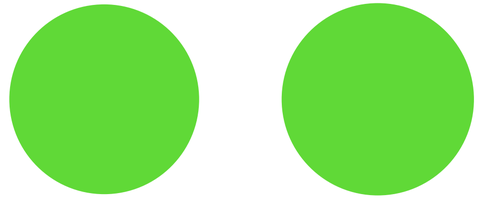
Points-Only Encoder



- Apply **sparse 3d convolution** to our points

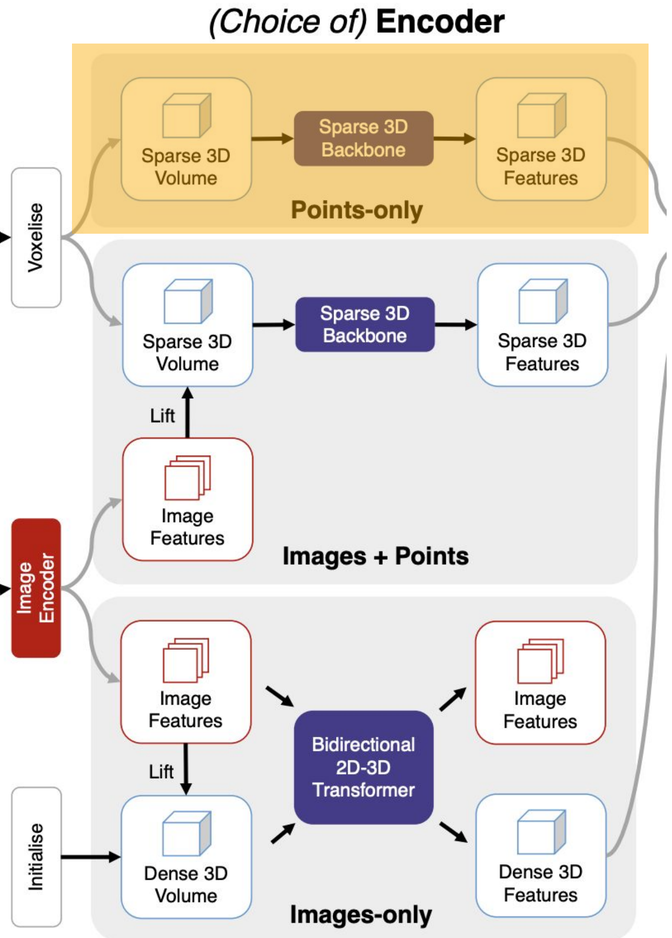
Red parts are skipped!





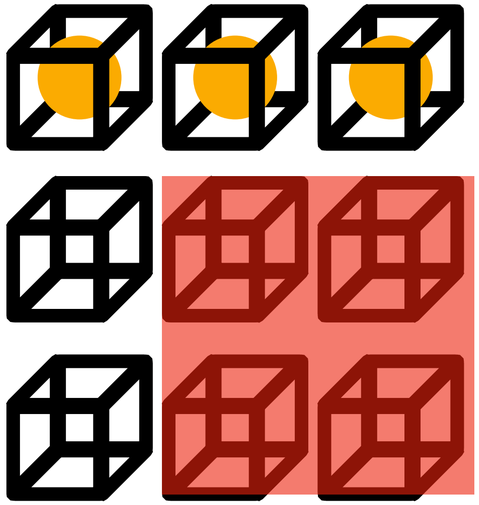
Architecture

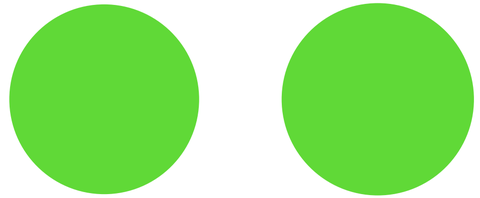
Points-Only Encoder



- Apply **sparse 3d convolution** to our points

Red parts are skipped!





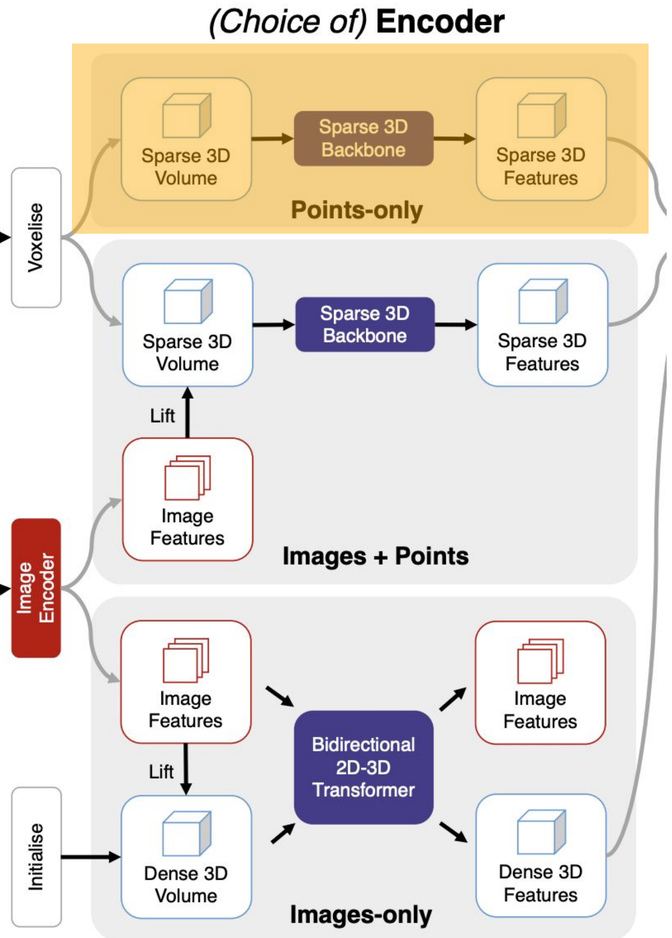
TorchSparse: Efficient Point Cloud Inference Engine

Haotian Tang [*1] Zhijian Liu [*1] Xiuyu Li [*2] Yujun Lin [1] Song Han [1]

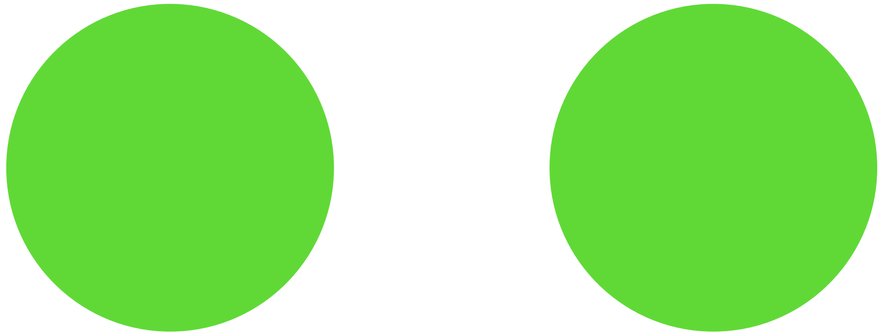
https://torchsparse.mit.edu

Architecture

Points-Only Encoder



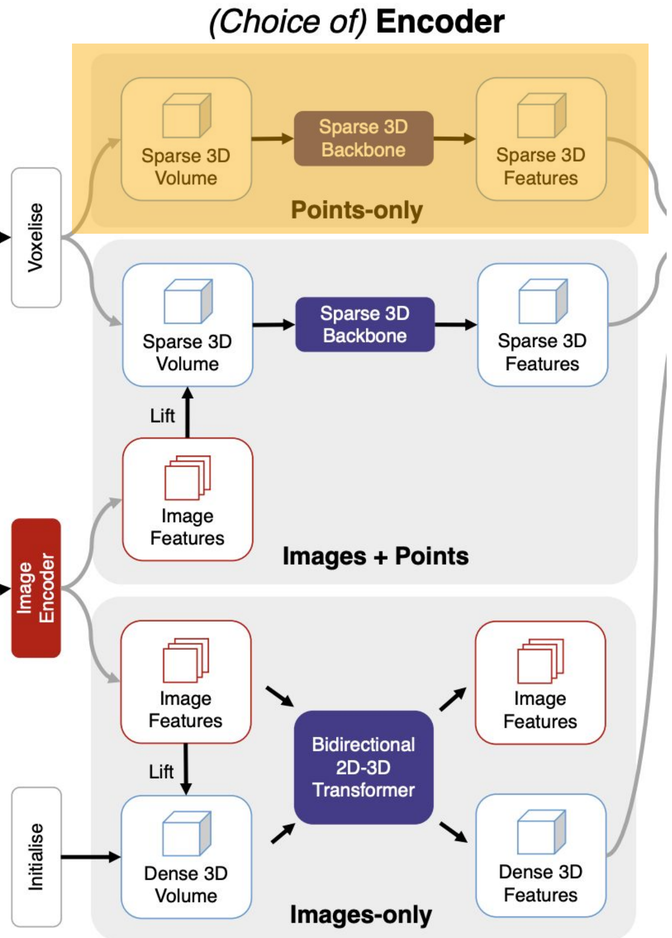
- Apply **sparse 3d convolution** to our points



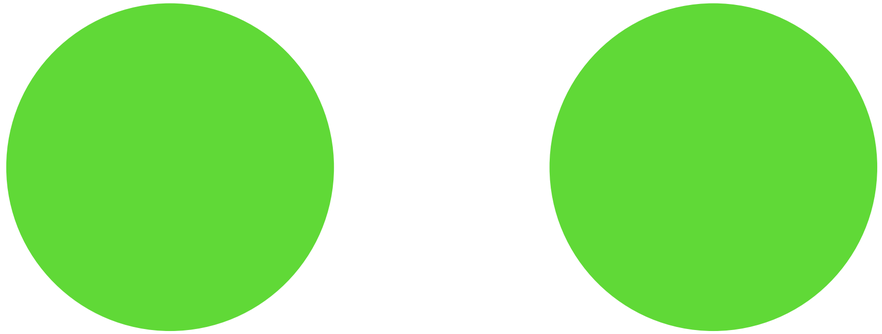
“active sites”

Architecture

Points-Only Encoder



- Apply **sparse 3d convolution** to our points

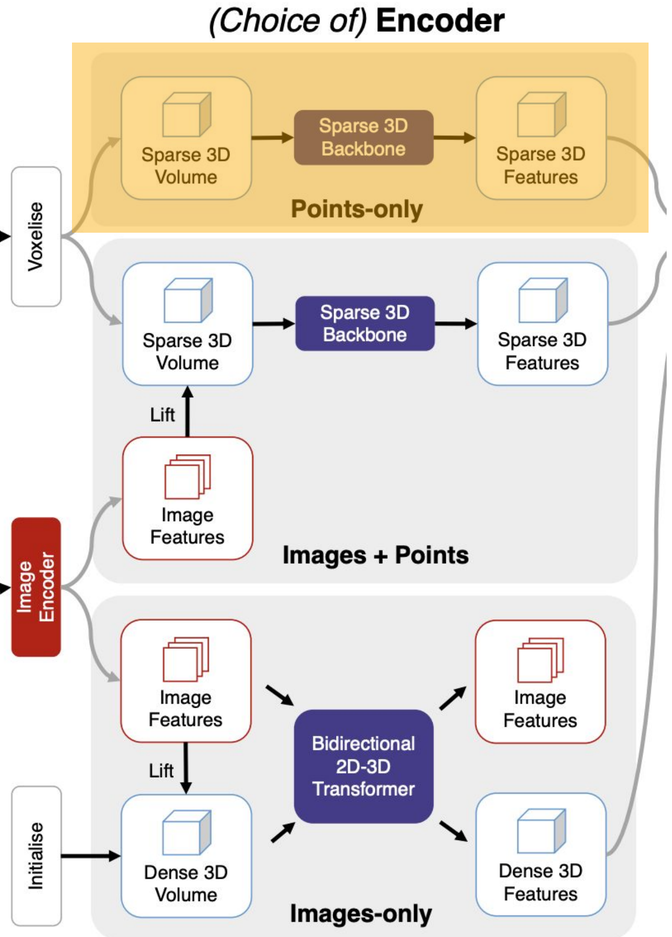


“active sites”

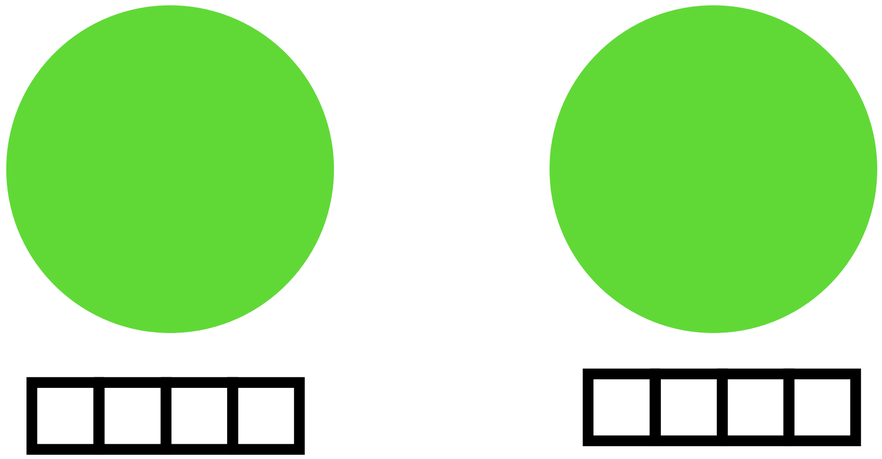
Each site contains **FEATURES**!

Architecture

Points-Only Encoder



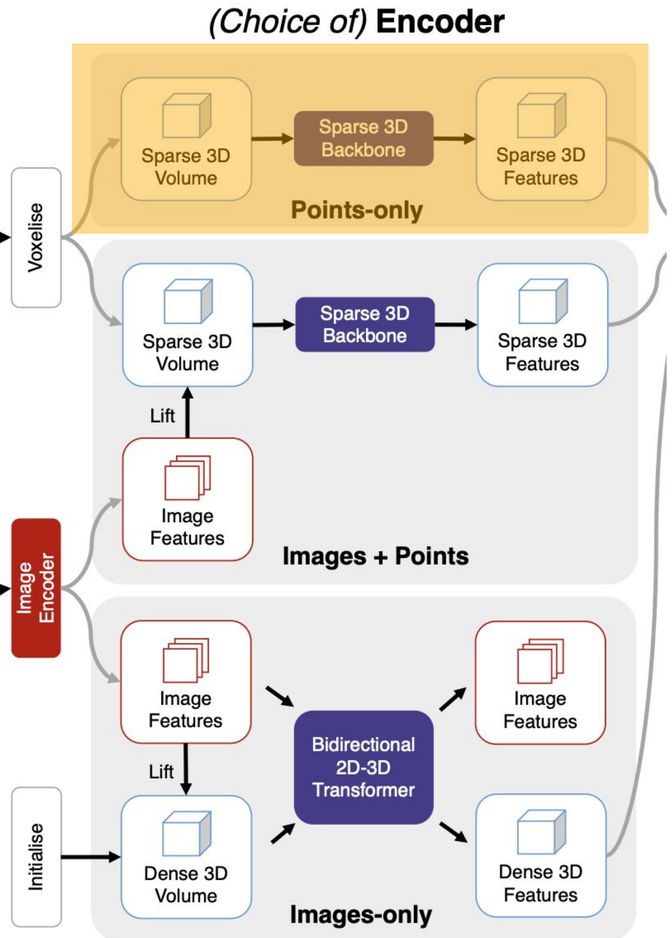
- Apply **sparse 3d convolution** to our points



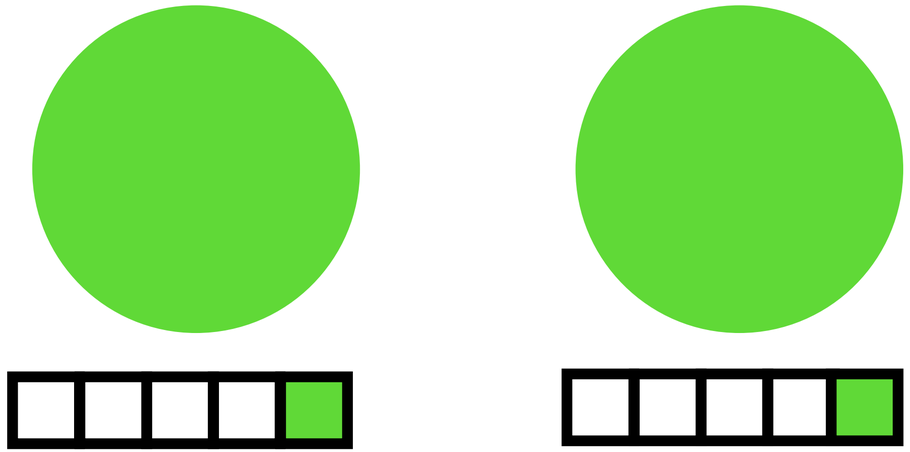
Features associated with each *active site* !

Architecture

Points-Only Encoder



- Apply **sparse 3d convolution** to our points



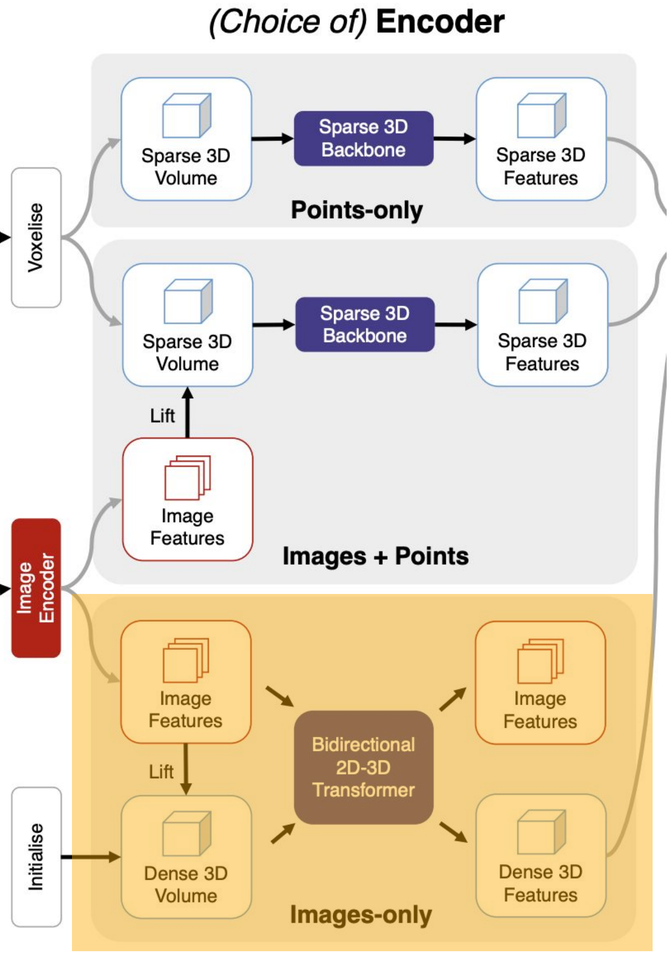
Features associated with each *active site* !

+ Positional Encodings!

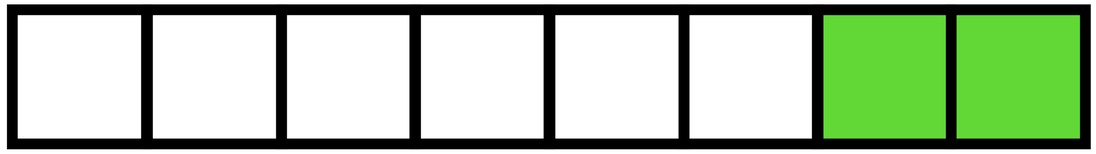
Architecture

Posed-View Encoder

Adopted from "RayTran"



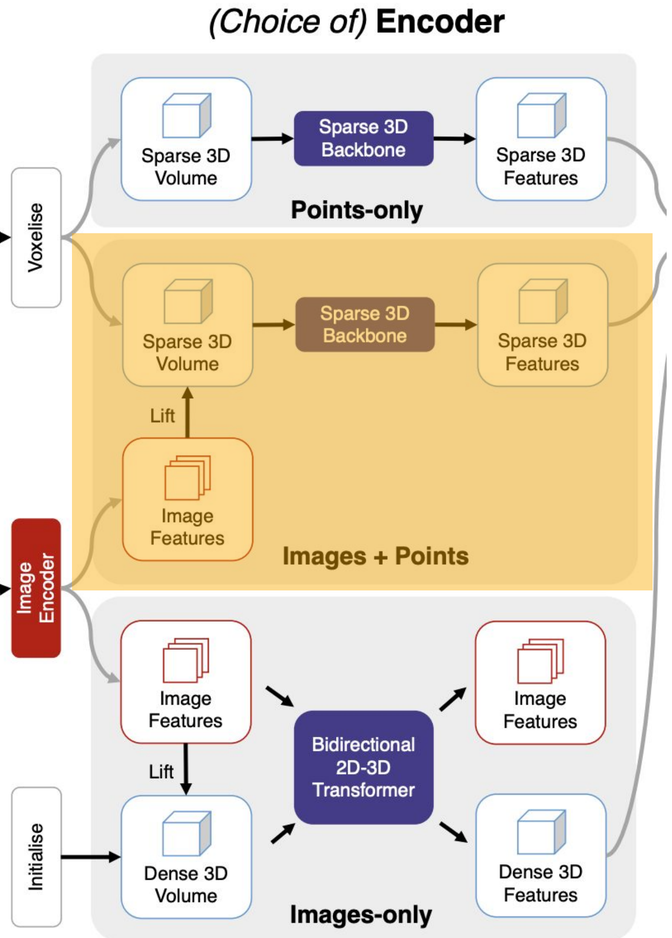
- 2D to 3D bidirectional encoder
- Initialize voxel grid of features
- Sample keyframes from images
- For each sample, get image features from CNN
- Iterative refinement of image and voxel grid features (through **viewpoint** + **global scene** info)



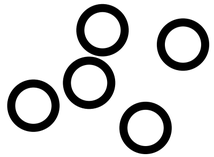
+ Positional Embedding

Architecture

Lifted-Feature Point Cloud Encoder



“camera projection function”

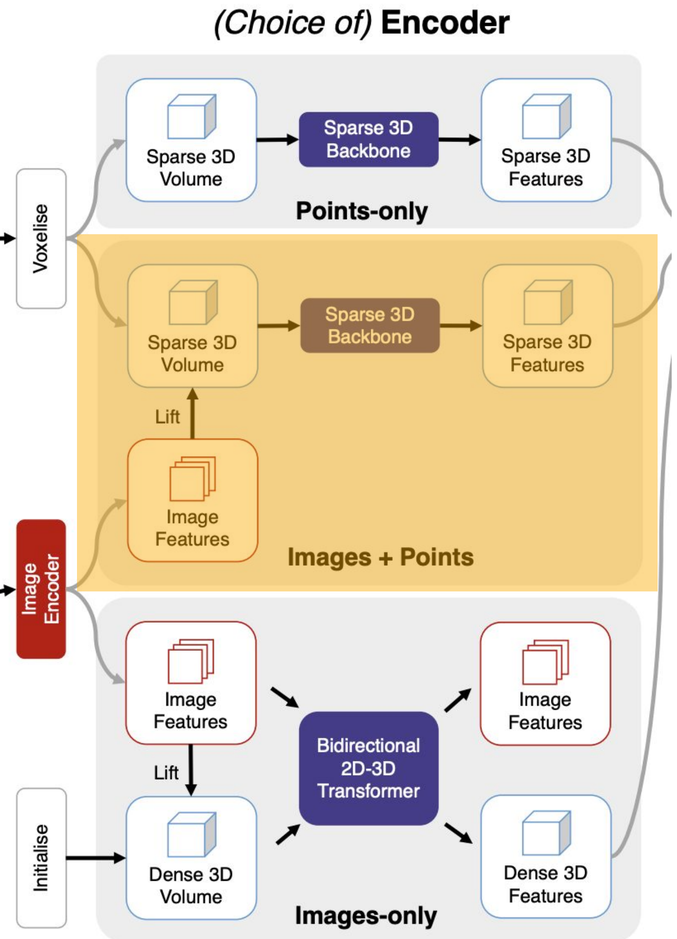


“points”

“Image features”

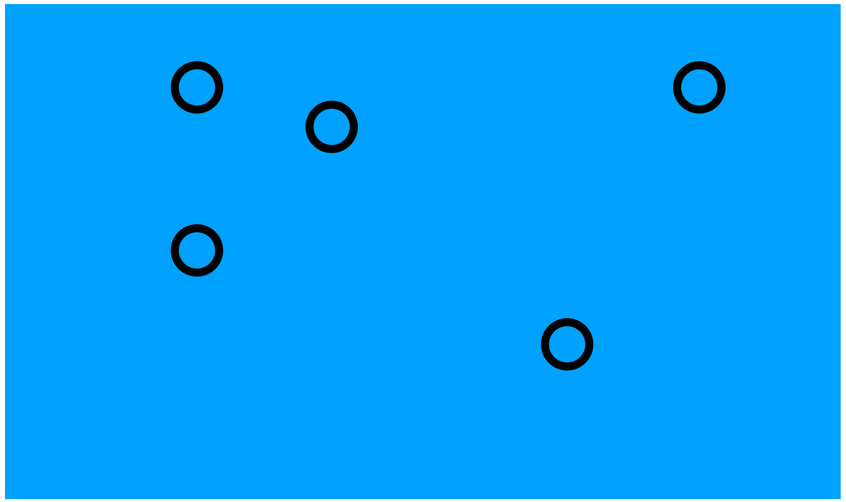
Architecture

Lifted-Feature Point Cloud Encoder



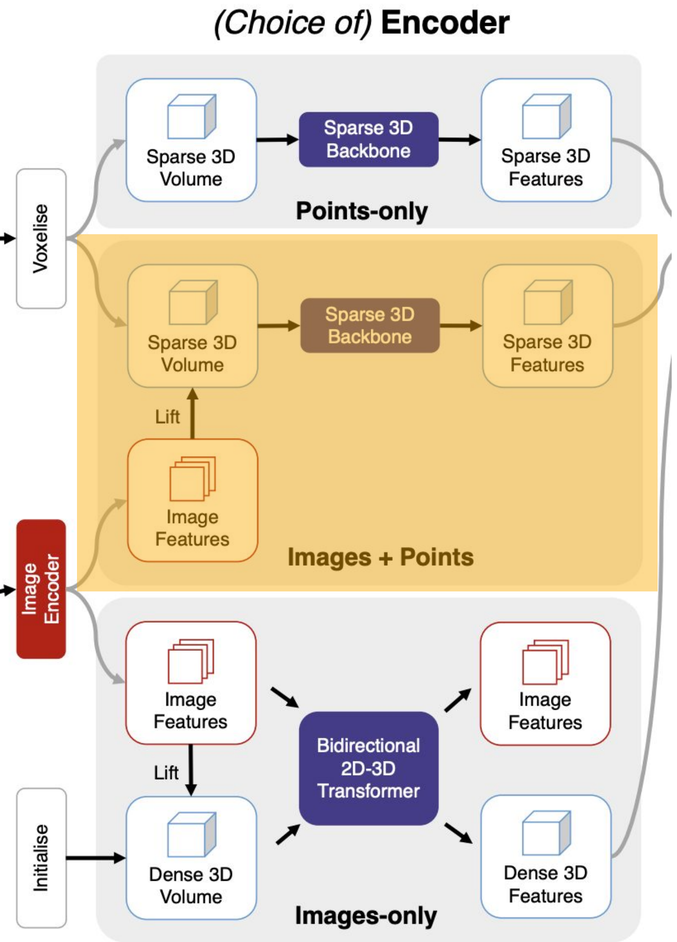
(( ))

“Feature vector @ point for keyframe ”



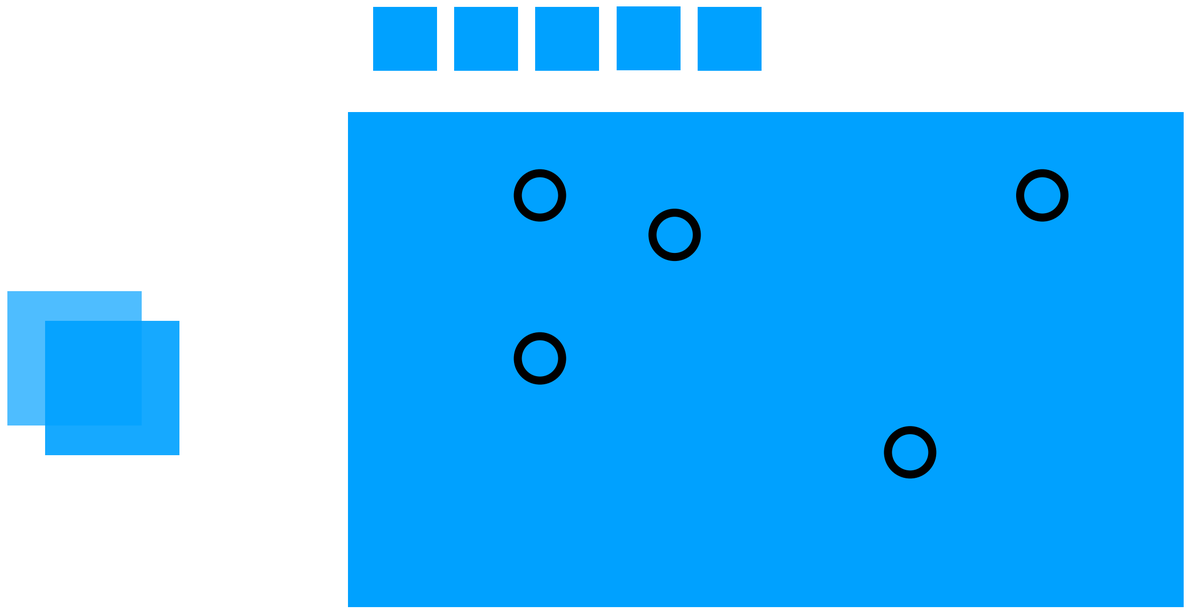
Architecture

Lifted-Feature Point Cloud Encoder



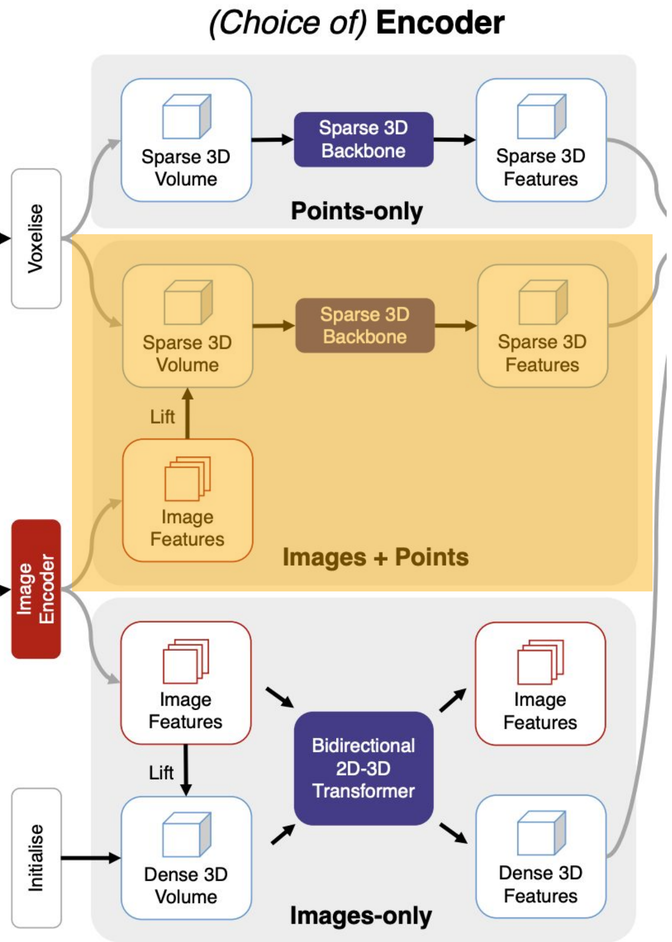
(( ))

“Feature vector @ point for keyframe ”



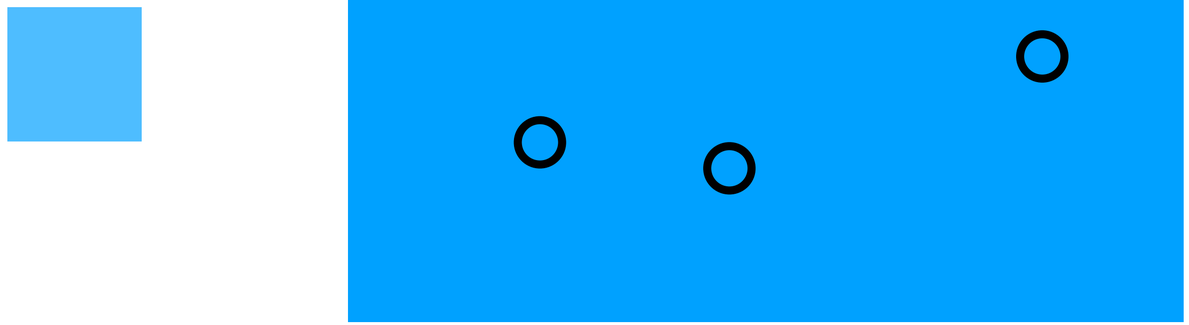
Architecture

Lifted-Feature Point Cloud Encoder



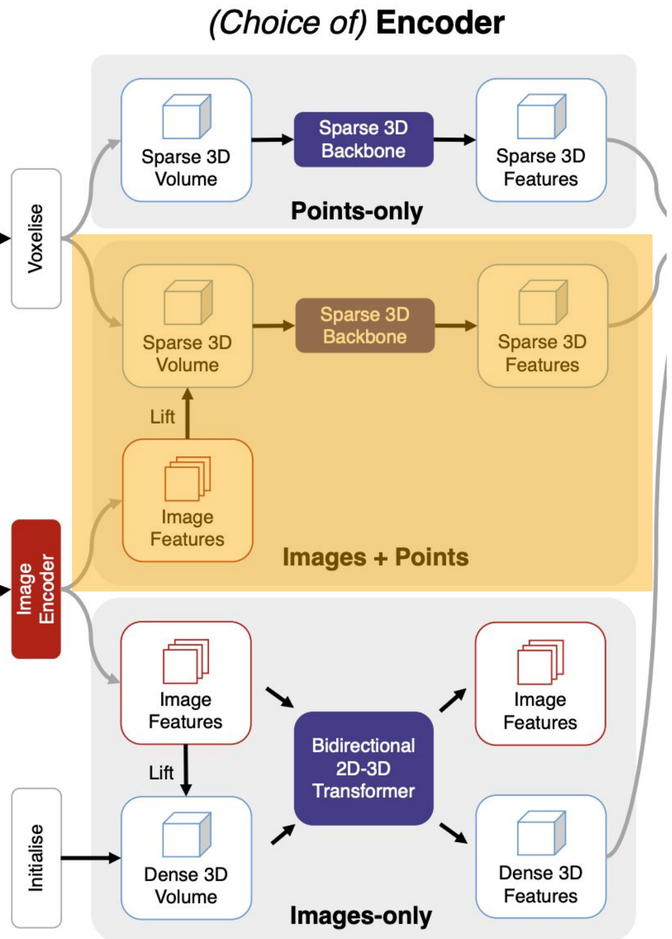
(( ))

“Feature vector @ point for keyframe ”



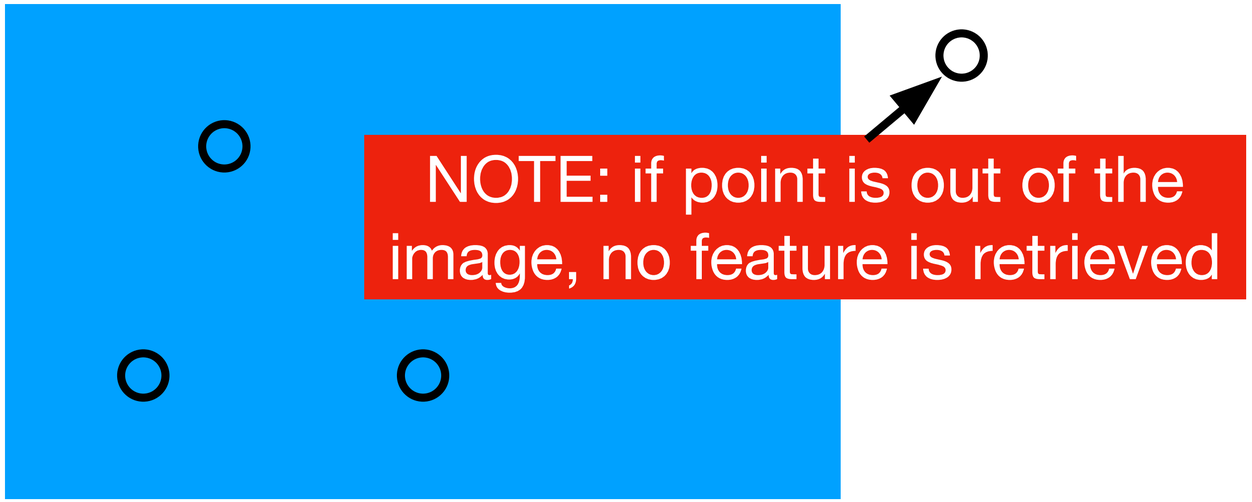
Architecture

Lifted-Feature Point Cloud Encoder



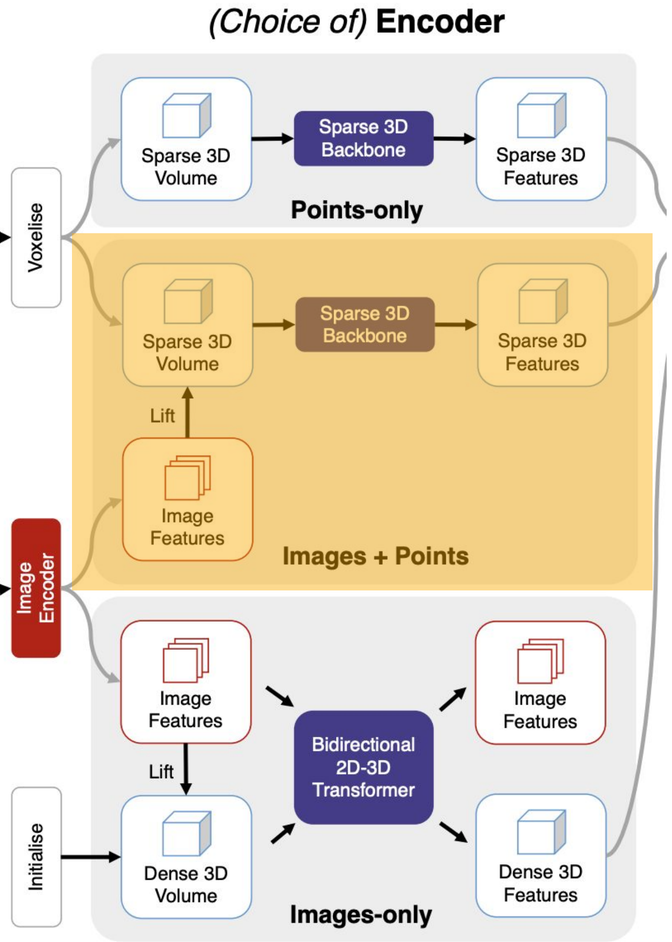
(( ))

“Feature vector @ point for keyframe ”



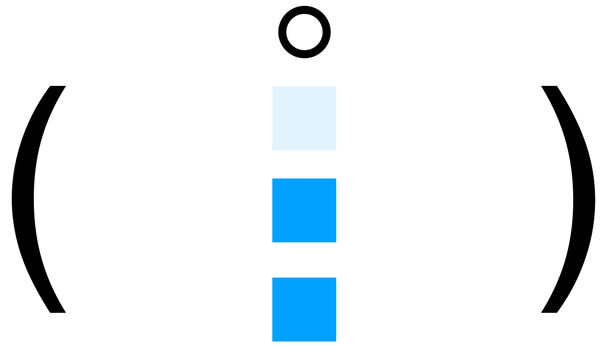
Architecture

Lifted-Feature Point Cloud Encoder



(( ))

“Feature vector @ point for keyframe ”

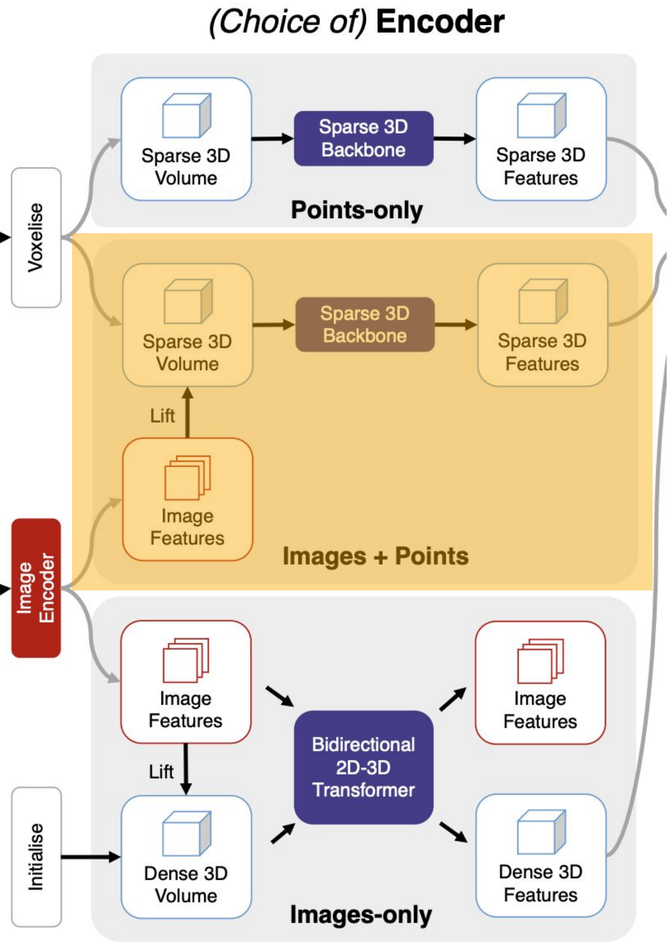


“Feature vector for is the *average* over keyframes”

+ Positional Encoding with original p_XYZ

Architecture

Lifted-Feature Point Cloud Encoder



(( ))

“Feature vector @ point for keyframe ”

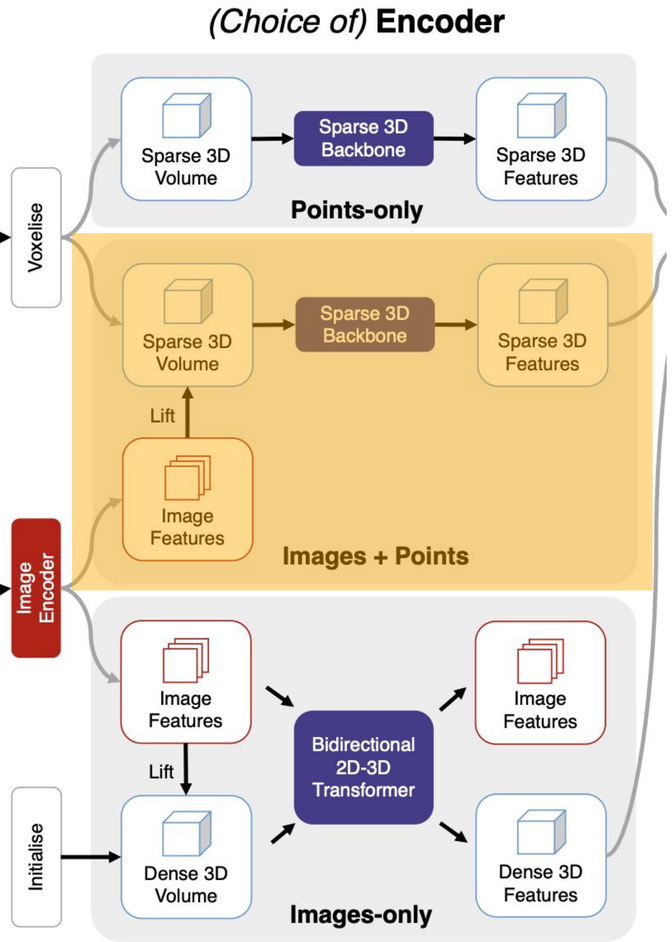
( )

“Feature vector for is the *average* over keyframes”

+ Positional Encoding with original p_XYZ

Architecture

Lifted-Feature Point Cloud Encoder



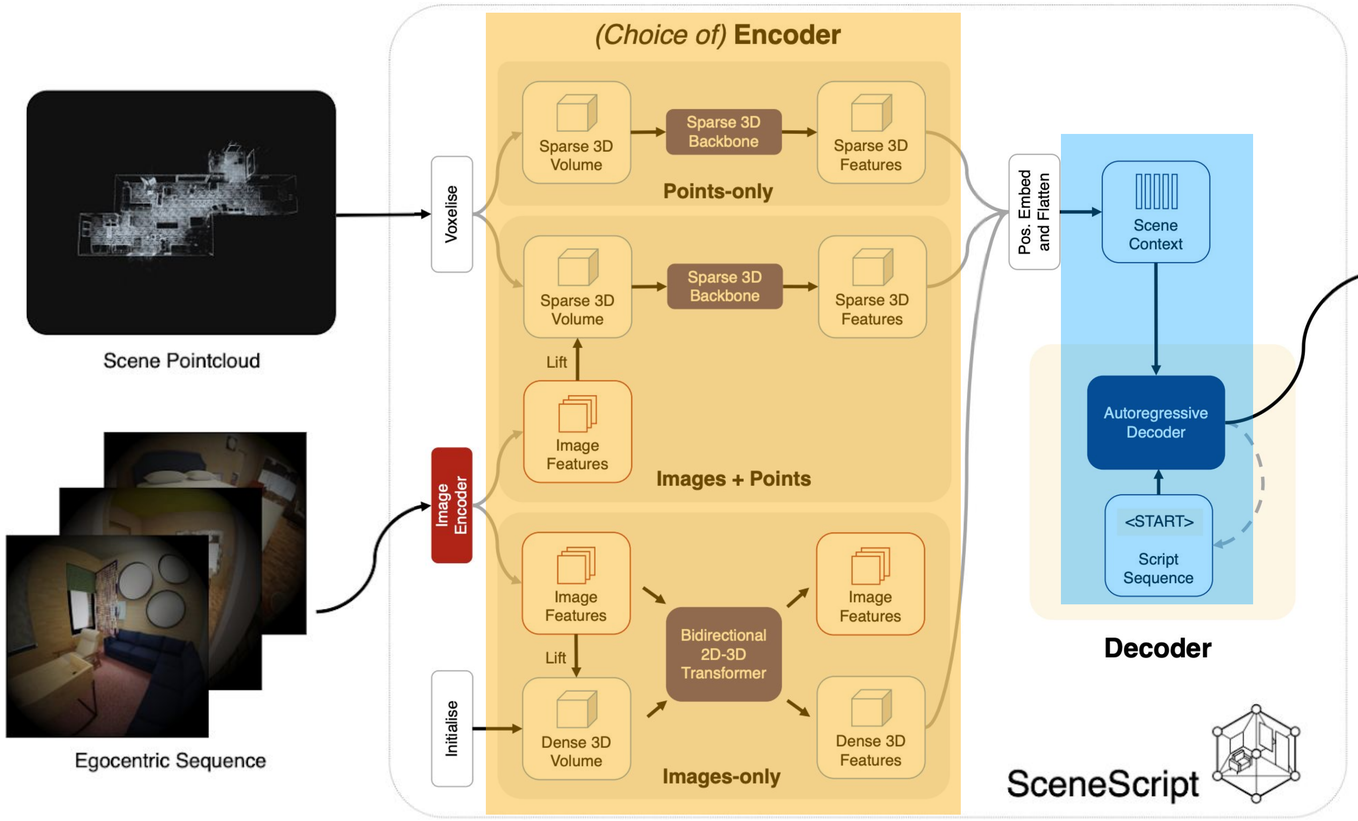
(( ))

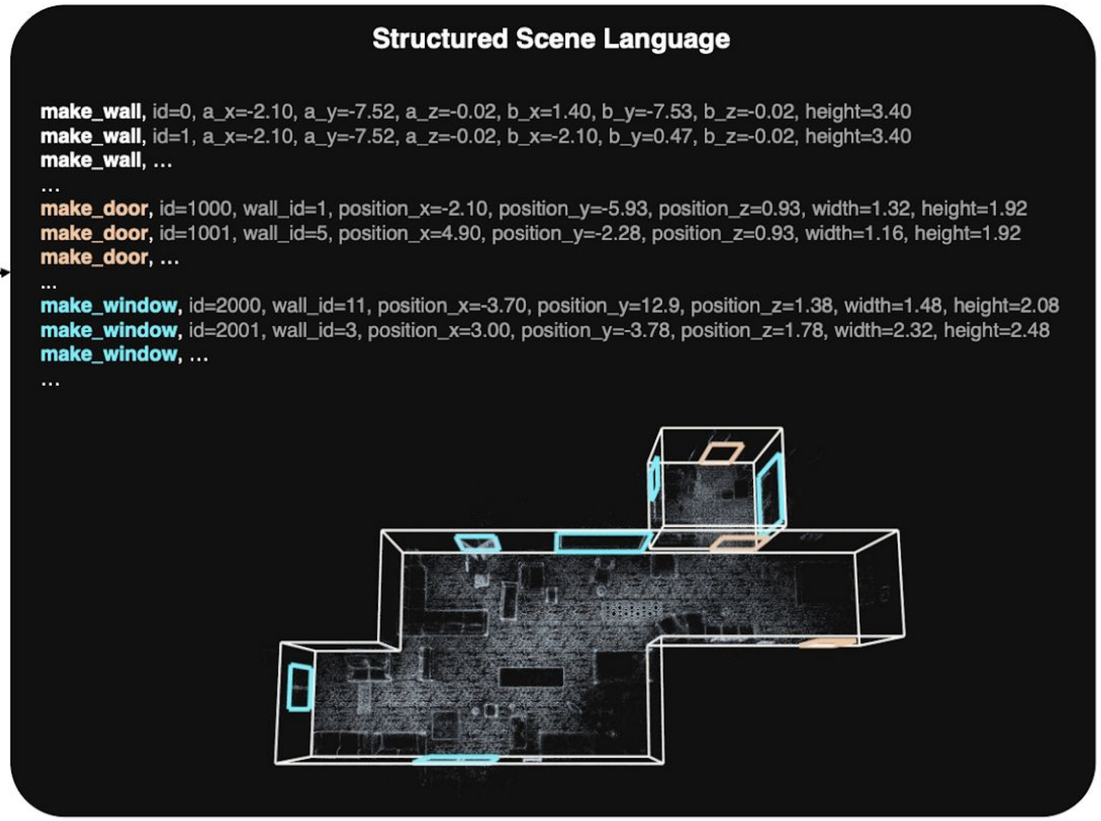
“Feature vector @ point for keyframe ”

“Feature vector for is the *average* over keyframes”

+ Positional Encoding with original p_XYZ

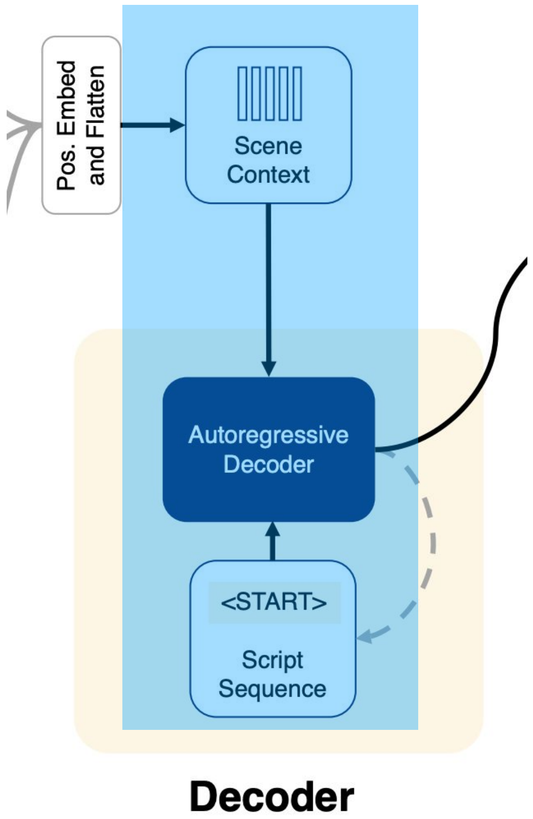
Architecture

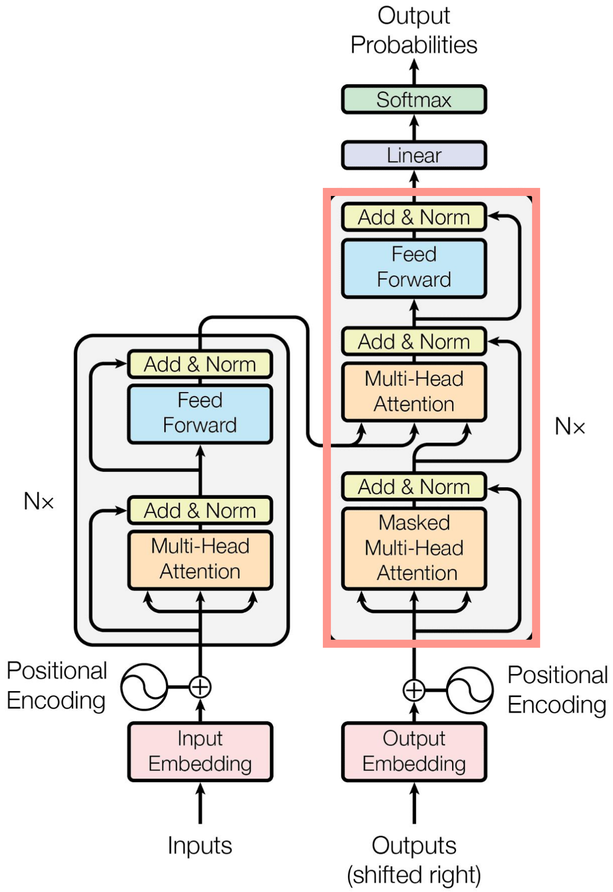




Architecture

Decoder





- Simply, the Transformer Decoder
- [!] Causal Attention layer ensures autoregressive generation

Architecture

Training?

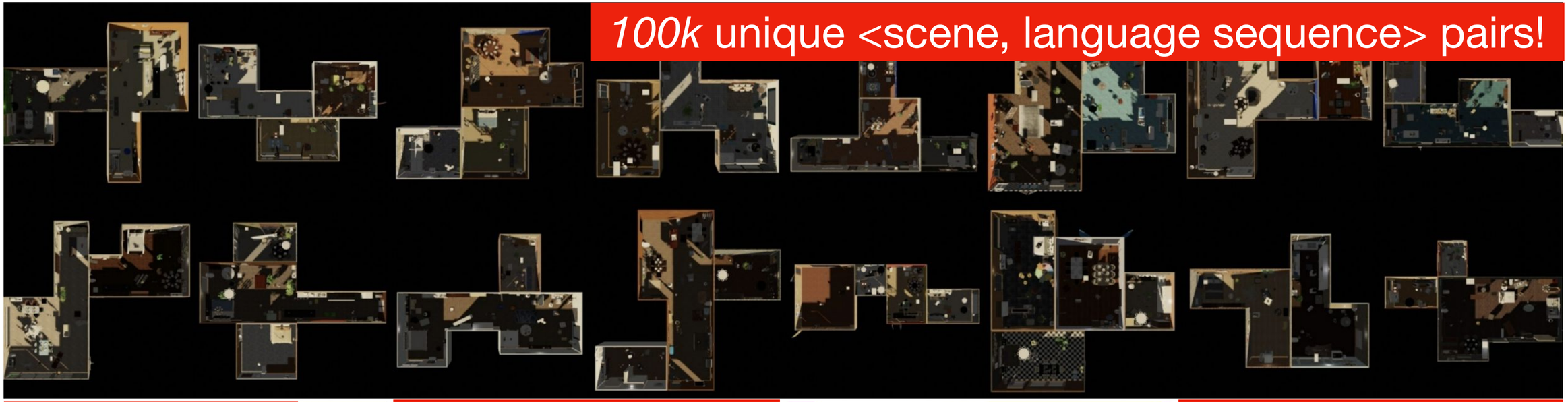
We need a dataset with…

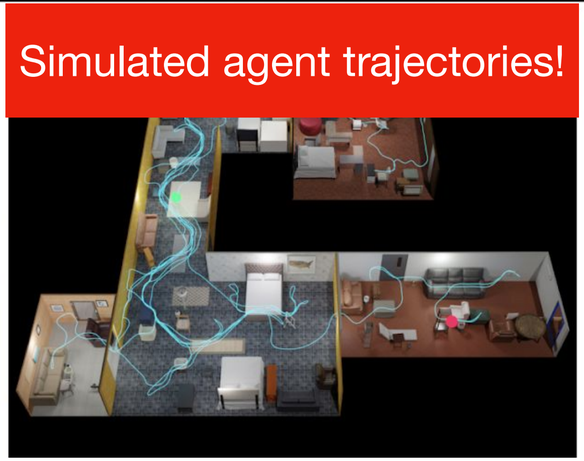
- Scene Walkthrough
- Corresponding Language Commands

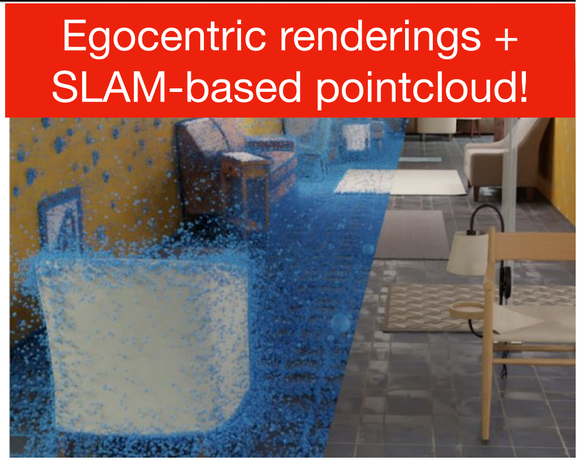
This *DOESN'T* exist!

Aria Synthetic Environments (ASE)

Large-Scale Synthetic Training Dataset

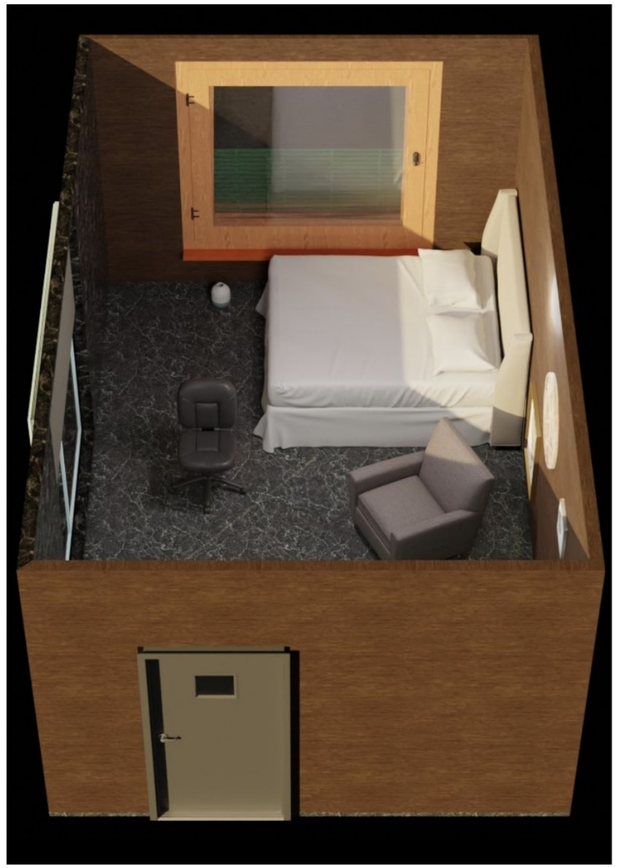






Aria Synthetic Environments (ASE)

An example of a training data pair:



- make_wall, id=0, a_x=2.1, a_y=3.9, a_z=0.0, b_x=7.8, b_y=3.9, b_z=0.0, height=2.7
- make_wall, id=1, a_x=7.8, a_y=3.9, a_z=0.0, b_x=7.8, b_y=0.3, b_z=0.0, height=2.7
- make_wall, id=2, a_x=7.8, a_y=0.3, a_z=0.0, b_x=2.1, b_y=0.3, b_z=0.0, height=2.7
- make_wall, id=3, a_x=2.1, a_y=0.3, a_z=0.0, b_x=2.1, b_y=3.9, b_z=0.0, height=2.7
- make_door, id=4, wall0_id=1, wall1_id=-1, position_x=7.8, position_y=1.5, position_z=1.0, width=1.0, height=1.9
- make_window, id=5, wall0_id=2, wall1_id=-1, position_x=5.3, position_y=0.3, position_z=1.4, width=2.3, height=2.5
- make_window, id=6, wall0_id=3, wall1_id=-1, position_x=2.1, position_y=2.1, position_z=1.4, width=2.2, height=2.1

Aria Synthetic Environments (ASE)

An example of a training data pair:

```
- make_wall, id=0, a_x=2.1, a_y=3.9, a_z=0.0,
  b_x=7.8, b_y=3.9, b_z=0.0, height=2.7
- make_wall, id=1, a_x=7.8, a_y=3.9, a_z=0.0,
  b_x=7.8, b_y=0.3, b_z=0.0, height=2.7
- make_wall, id=2, a_x=7.8, a_y=0.3, a_z=0.0,
  b_x=2.1, b_y=0.3, b_z=0.0, height=2.7
- make_wall, id=3, a_x=2.1, a_y=0.3, a_z=0.0,
  b_x=2.1, b_y=3.9, b_z=0.0, height=2.7
- make_door, id=4, wall0_id=1, wall1_id=-1,
  position_x=7.8, position_y=1.5, position_z=1.0, width=1.0, height=1.9
- make_window, id=5, wall0_id=2, wall1_id=-1,
  position_x=5.3, position_y=0.3, position_z=1.4, width=2.3, height=2.5
- make_window, id=6, wall0_id=3, wall1_id=-1,
  position_x=2.1, position_y=2.1, position_z=1.4, width=2.2, height=2.1
```

- Same training as in NLP
- Cross Entropy Loss

+ Flexible sequence length
+ No limitations on # of sub-sequences

$$[\texttt{START}, \texttt{PART}, \texttt{CMD}, \texttt{PARAM\_1}, \texttt{PARAM\_2}, \ldots, \texttt{PARAM\_N}, \texttt{PART}, \ldots, \texttt{STOP}]$$

Token Sequence Schema

Tasks

[Layout Estimation]

- Other works (Scan2BIM, RoomFormer, SceneCAD) **make assumptions** about the scene and **requires heuristics**

Tasks

[Layout Estimation]

- **SceneScript** *doesn't* require **heuristics** nor **prior knowledge** of the scene!

Table 2: Layout Estimation on *Aria Synthetic Environments* Quantitative comparison between our method [illegible] ork.

SOTA

Method	Avg F1 mean	wall	door	window	Avg F1 mean	wall	door	window
SceneCAD '20 [2]	-	0.048	-	-	-	0.275	-	-
RoomFormer '23 [52]	0.139	0.159	0.148	0.110	0.464	0.505	0.481	0.407
Ours (Point cloud)	0.848	0.930	0.922	0.692	0.784	0.816	0.811	0.724
Ours (Lifted features)	**0.903**	**0.943**	**0.959**	**0.806**	**0.801**	**0.818**	**0.822**	**0.764**
Ours (Image-only)	0.661	0.687	0.798	0.497	0.719	0.727	0.772	0.658

Tasks

[3D Object Detection]

- Compared with **Cube R-CNN 3DETR, ImVoxelNet, SoftGroup**
- **SceneScript** performs comparably to **3DETR**

NOT SOTA

(a) *Aria Synthetic Envi*

Method	Input	F1 @.25 IoU	F1 @.50 IoU
3DETR '21 [26]	Points	0.201	0.078
Cube R-CNN '23 [4]	RGB	0.394	0.228
ImVoxelNet '22 [36]	RGB	0.584	0.516
Ours	Points	**0.620**	**0.577**

(b) ScanNet [10]

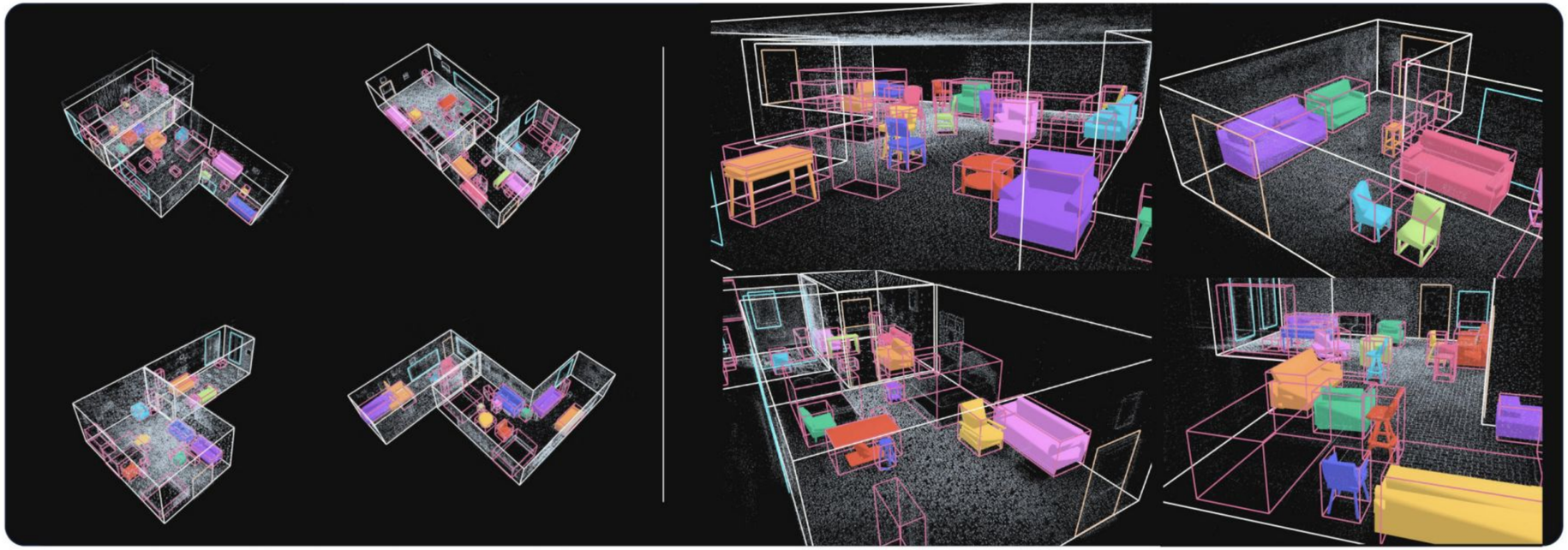
Method	Input	F1 @.25 IoU	F1 @.50 IoU
3DETR '21 [26]	Points	0.480	0.349
3DETR-m '21 [26]	Points	0.536	0.407
SoftGroup '22 [46]	RGB Points	**0.622**	**0.573**
Ours	RGB Points	0.506	0.406

Limitations

- Coarse — Challenging to capture fine-grained details (5cm voxels)
- Manually-defined structured language commands
  - Ex. Can’t automatically create “make_sink”

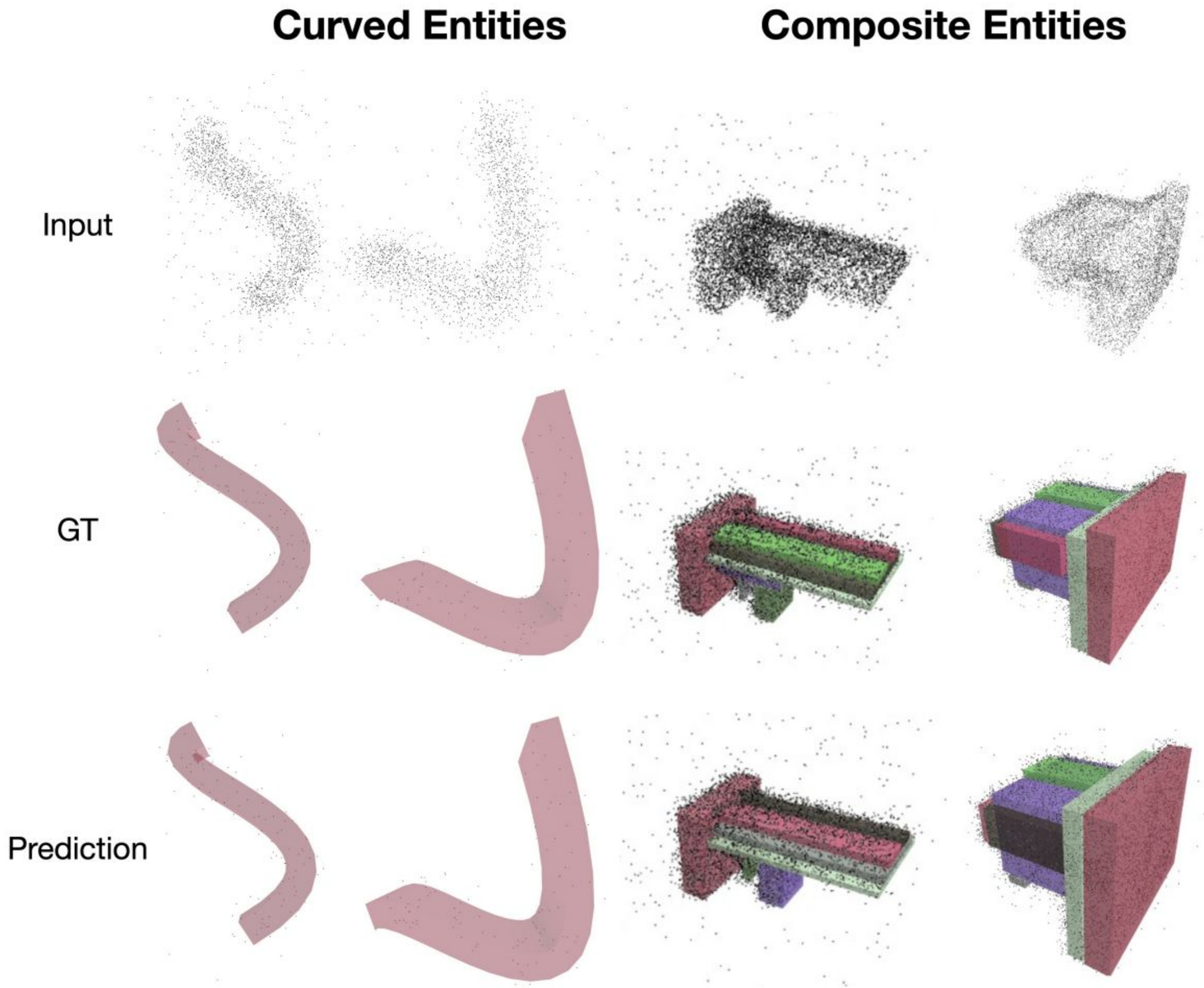
Extensions

[Corase 3D Object Reconstruction]



Extensions

[Curved Objects]



Extensions

[Object State Estimation]

- make_door:
  pos_x: 0.0
  ...
  angle_z: 1.86

Can you think of a way to extend SceneScript?

Trivia!

(If time permits)

1. What's the **high level idea** of SceneScript?

2. What are some **benefits** it provides **over** other **representations**?

3. How does the **joint encoder** work?