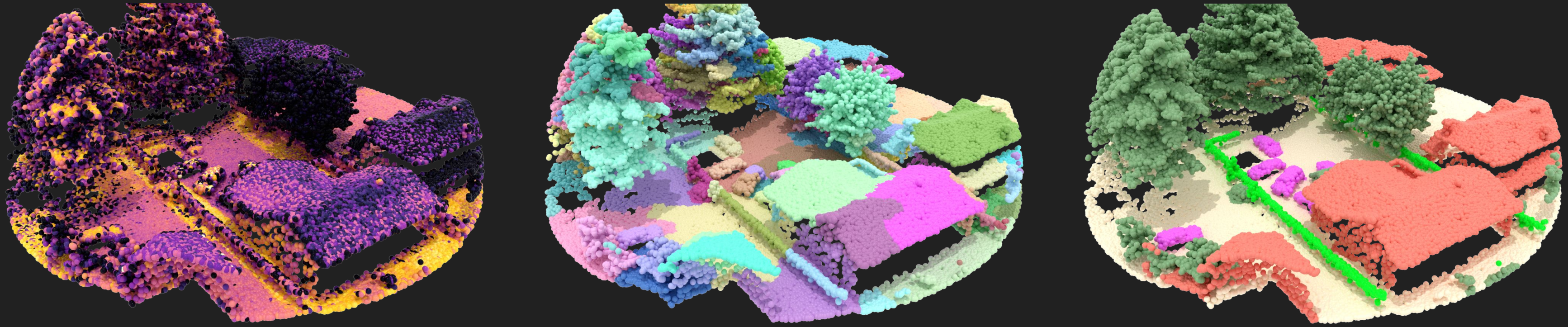
Superpoint Transformer

Efficient learning on large scale 3D point clouds



Damien Robert, University of Zurich, DM3L, EcoVision lab

About me

Damien Robert

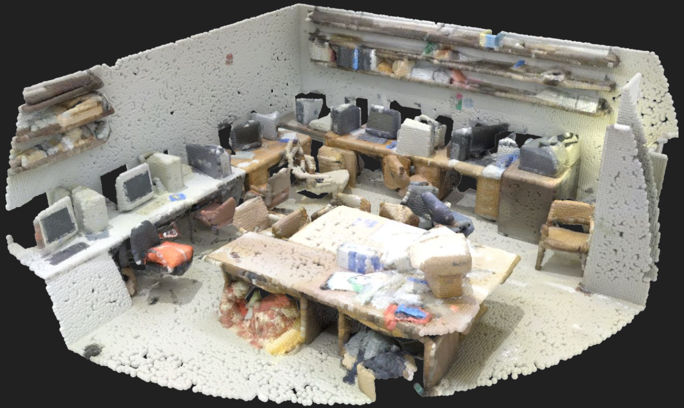
University of Zurich, DM3L, EcoVision lab

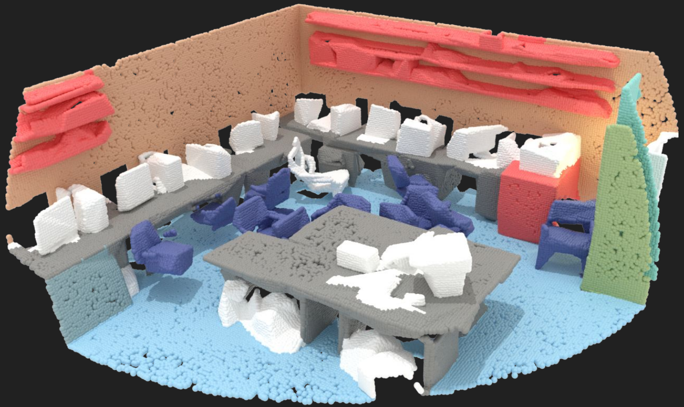
/drprojects • drprojects.github.io • damien.robert@uzh.ch

Efficient learning on large scale 3D point clouds

PhD at IGN LASTIG & ENGIE CRIGEN labs

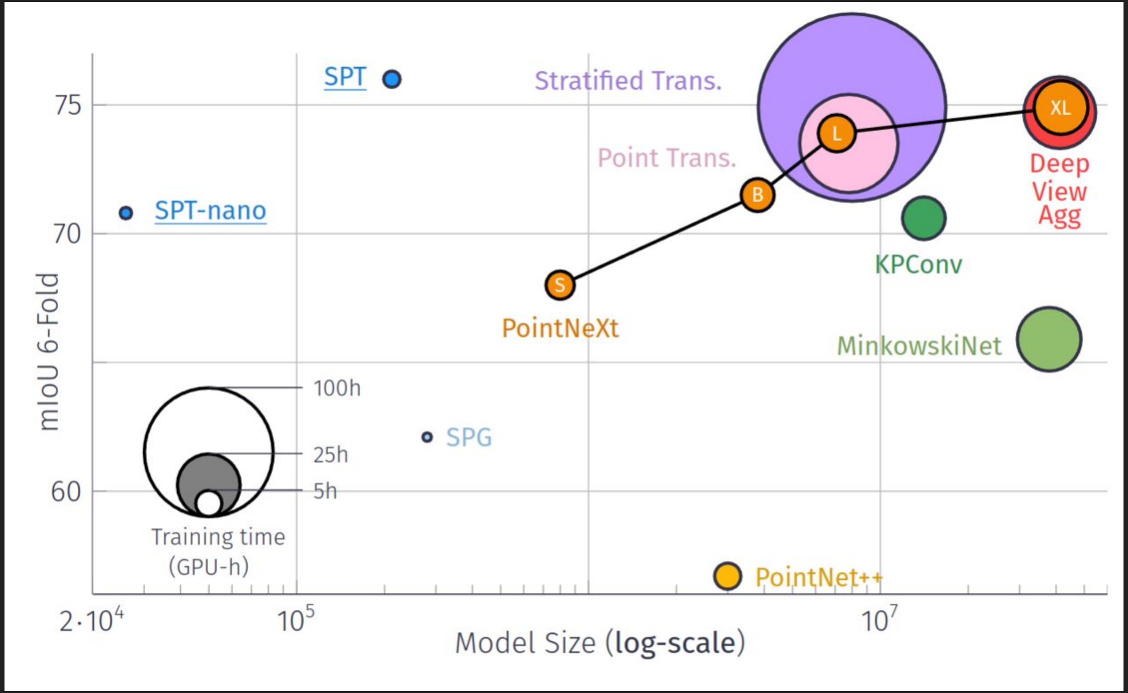
3D scene semantic understanding


Input


Target

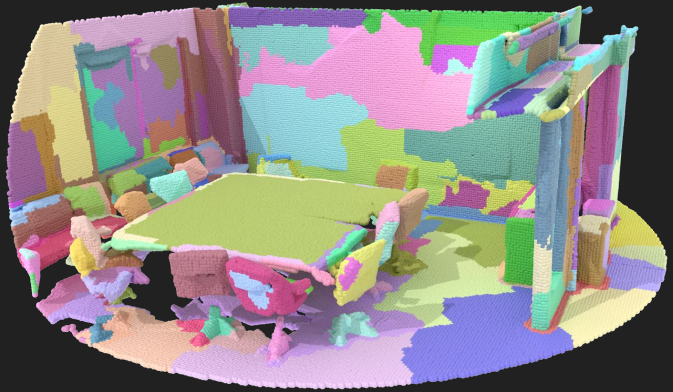
Large-scale 3D point clouds of **+10M points**

✨ Spoiler



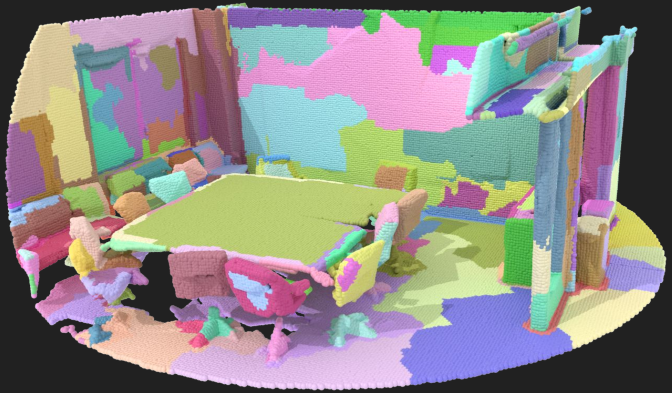
Model Size vs. Performance on S3DIS 6-Fold

💡 Motivation



🧩 **Partition** point cloud into semantically-homogeneous **superpoints**

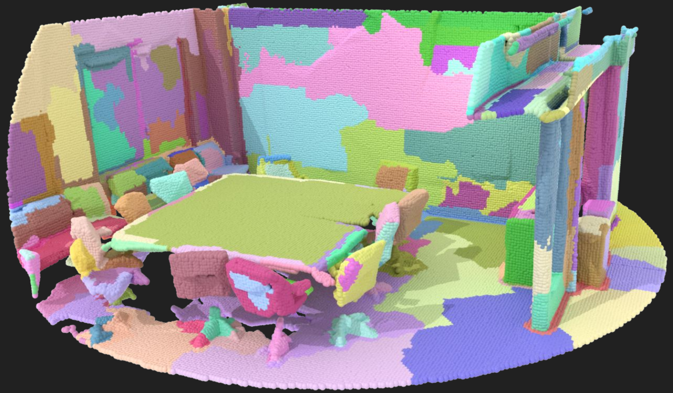
💡 Motivation



🧩 **Partition** point cloud into semantically-homogeneous **superpoints**

Learn to **classify** the superpoints

💡 Motivation



🧩 **Partition** point cloud into semantically-homogeneous **superpoints**

Learn to **classify** the superpoints

🧩 **Geometry-guided** compute effort allocation

🔩 Pipeline - **Preprocessing**

✨***RAW***

🔩 Pipeline - **Preprocessing**

✨***RAW***

✂️ **tiling**

🔩 Pipeline - Preprocessing

P_0

✂️ tiling
➕
🧊 voxelization ➡️ P_0

🔩 Pipeline - Preprocessing

P_0

✂️ tiling

➕

🧊 voxelization ➡️ P_0

➕

📏 neighbor search

🔩 Pipeline - Preprocessing

P_0

✂️ tiling
➕
🧊 voxelization ➡️ P_0
➕
📏 neighbor search
➕
📊 local features

🔩 Pipeline - Preprocessing

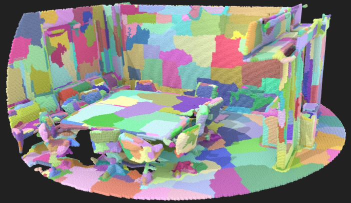
P_0

✂️ tiling
➕
🧊 voxelization ➡️ P_0
➕
📏 neighbor search
➕
📊 local features
➕
🕸 point adjacency graph

🔩 Pipeline - Preprocessing

P_0

P_1

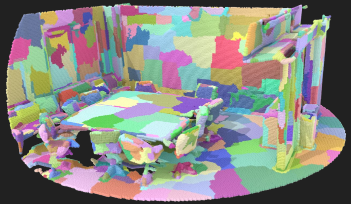


✂️ tiling
➕
🧊 voxelization ➡️ P_0
➕
📏 neighbor search
➕
📊 local features
➕
🕸 point adjacency graph
➕
🧩 hierarchical superpoint partition ➡️ $P_{i>0}$

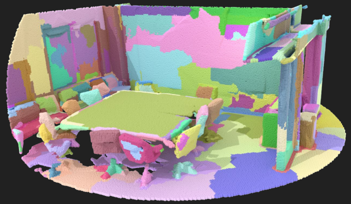
🔩 Pipeline - Preprocessing

P_0

P_1



P_2



✂️ tiling

➕

🧊 voxelization ➡️ P_0

➕

📏 neighbor search

➕

📊 local features

➕

🕸 point adjacency graph

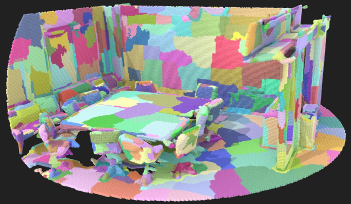
➕

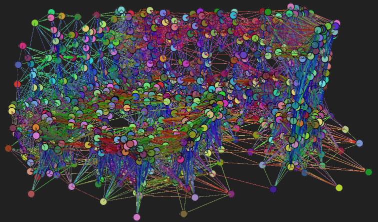
🧩 hierarchical superpoint partition ➡️ $P_{i>0}$

🔩 Pipeline - **Preprocessing**

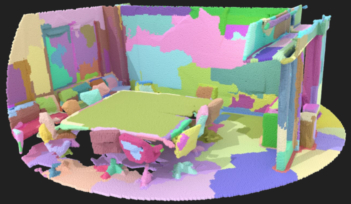
P_0

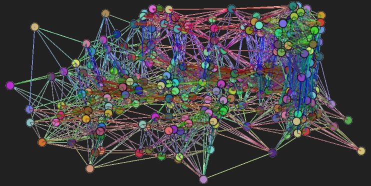
P_1





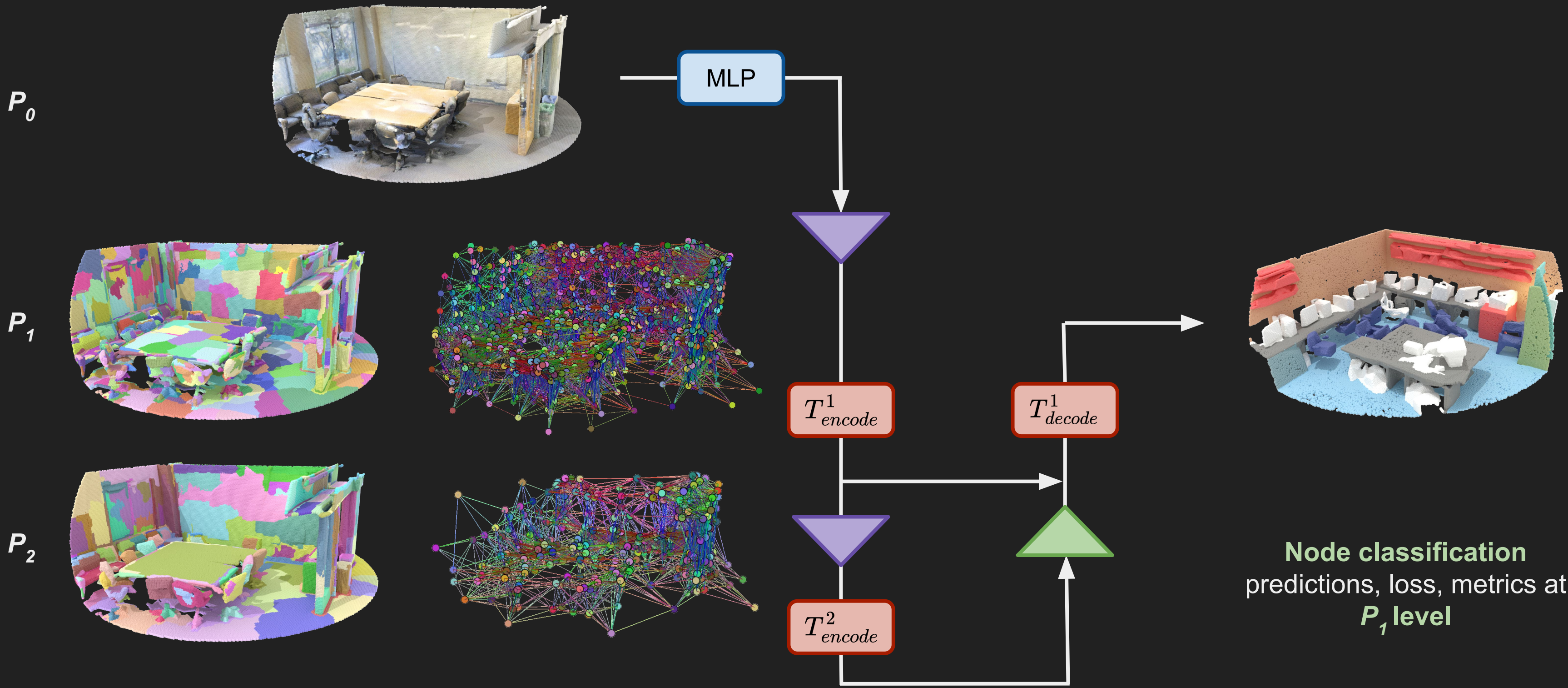
P_2





✂️ tiling
➕
🧊 voxelization ➡️ P_0
➕
📏 neighbor search
➕
📊 local features
➕
🕸 point adjacency graph
➕
🧩 hierarchical superpoint partition ➡️ $P_{i>0}$
➕
🕸 superpoint adjacency graphs

Pipeline - Training

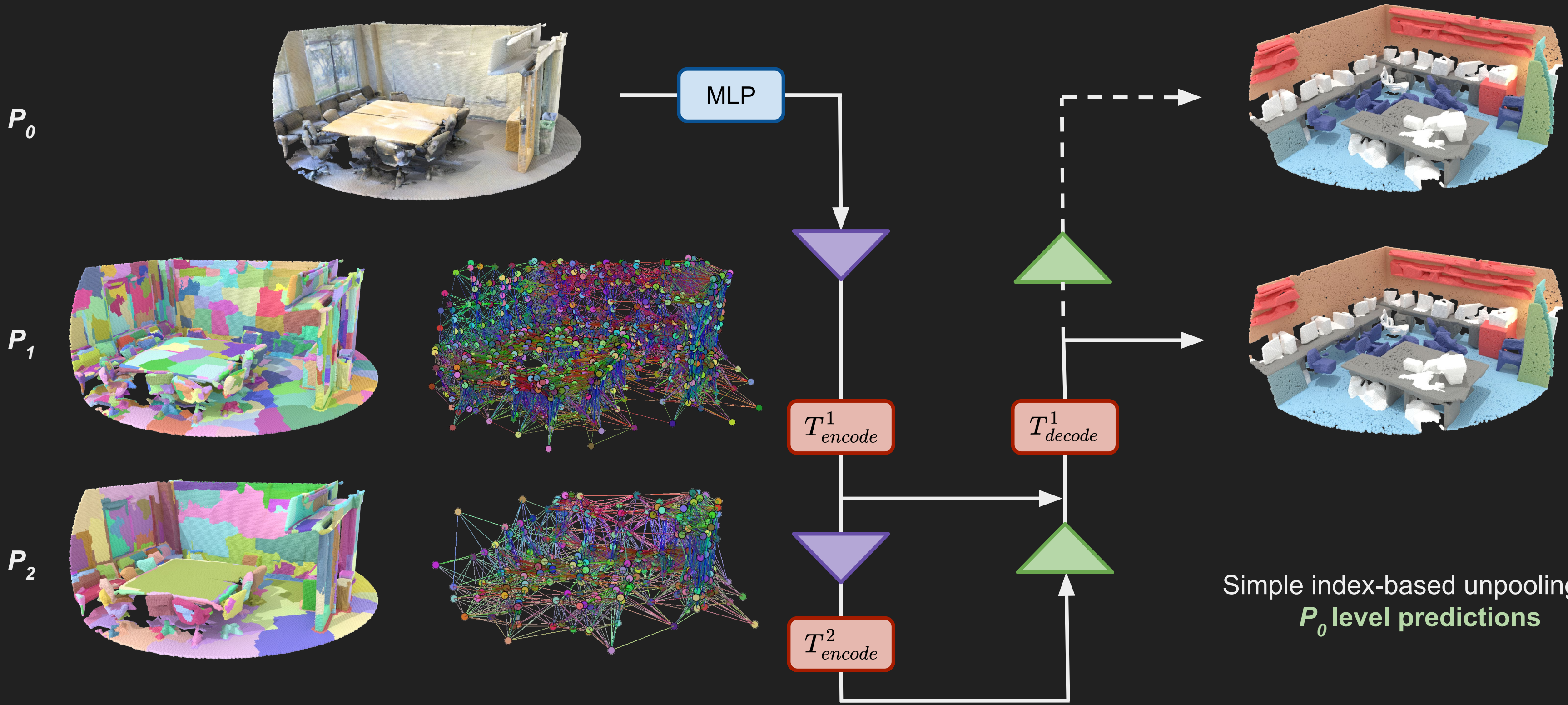


Node classification predictions, loss, metrics at ***P_1* level**

▽△ Partition-based pooling / unpooling

🟥 Graph-based transformer

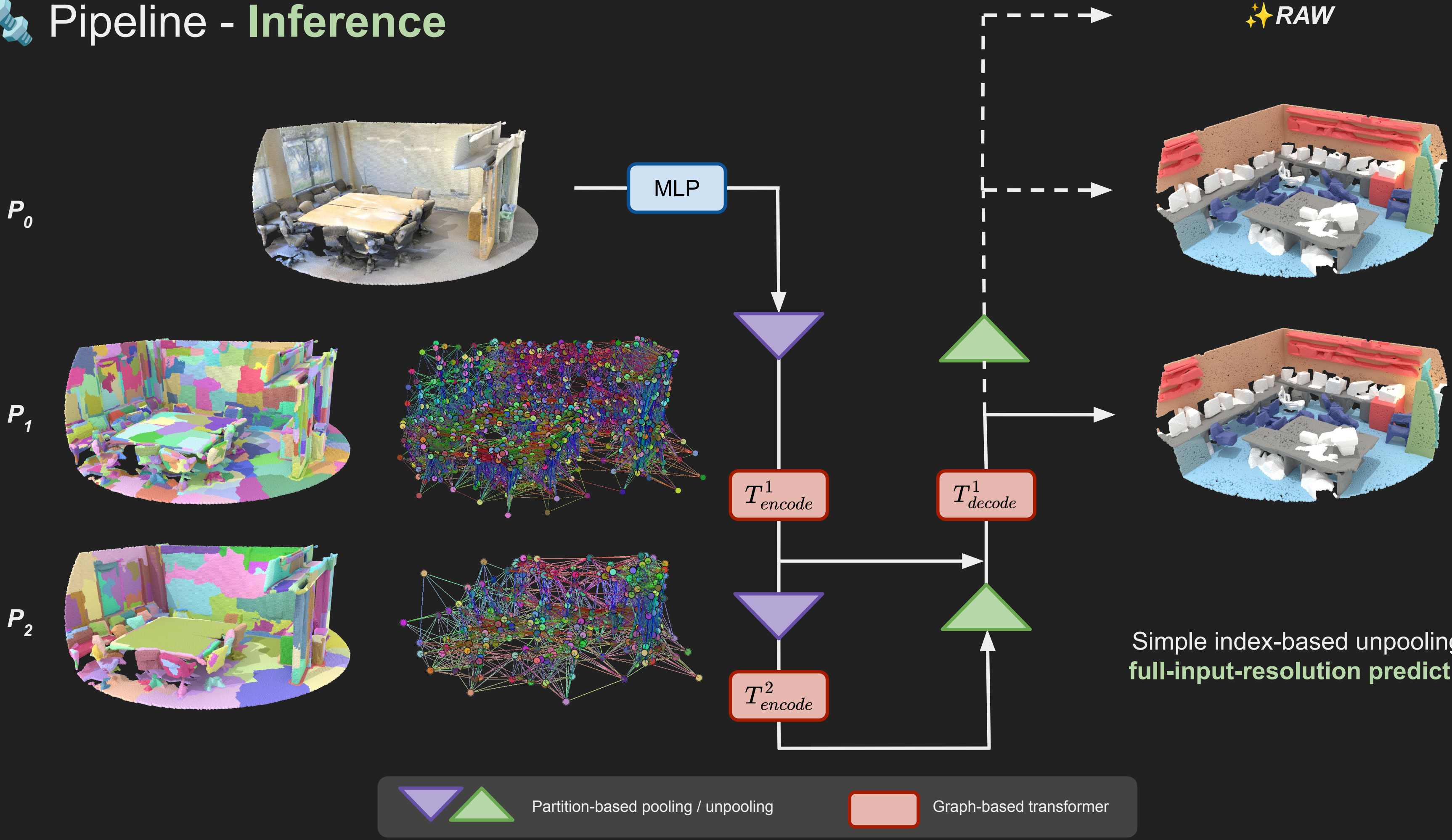
Pipeline - Inference



Simple index-based unpooling for **P_0 level predictions**

▽△ Partition-based pooling / unpooling   🟥 Graph-based transformer

Pipeline - Inference



Simple index-based unpooling for **full-input-resolution predictions**

👩‍💻 Implementation details - **Project structure & dependencies**

Code structure	`lightning-hydra-template`
Dataset structure	`PyTorch Geometric`
Data structures	`Data & NAG`

Before starting

See README and docs/

👩‍💻 Implementation details - **Data transforms**

Voxelization	GridSampling3D
Neighbor search	KNN
Elevation estimation	GroundElevation
Pointwise local geometric features	PointFeatures
Adjacency graph	AdjacencyGraph
Hierarchical partition	CutPursuitPartition
Superpoint-wise handcrafted features	SegmentFeatures
Superpoint adjacency graph and features	RadiusHorizontalGraph

`pre_transform`

only once at preprocessing time

Point / superpoint / edge sampling*	SampleSubNodes SampleRadiusSubgraphs SampleSegments SampleEdges NAGRestrictSize
Superpoint graph features	OnTheFlyHorizontalEdgeFeatures
Feature augmentations	NAGJitterKey NAGDropoutColumns NAGDropoutRows

`on_device_transform`

for each **train*** / val / test batch on GPU

👩‍💻 Tutorial

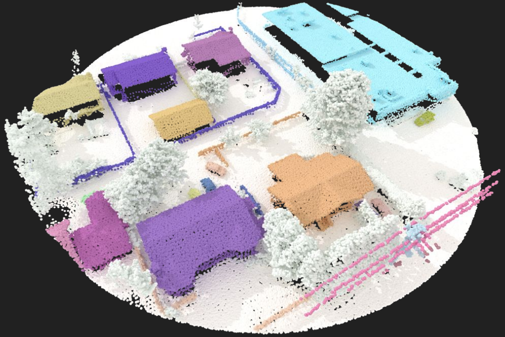
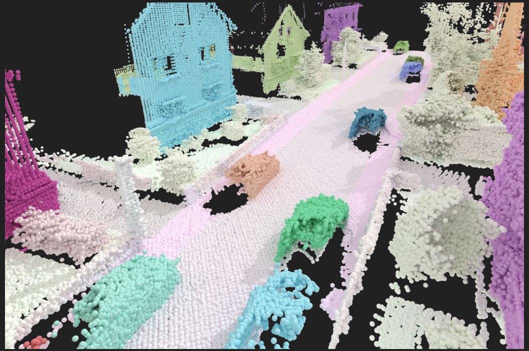
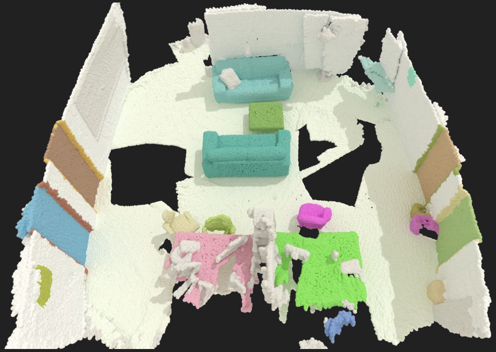
GitHub repository

drprojects/superpoint_transformer

Tutorial notebook

notebooks/superpoint_transformer_tutorial.ipynb

🤓 Going further - **Panoptic segmentation with SuperCluster**

See [paper]() (3DV'24 Oral)

👼 Be an angel - **Good practices for using SPT**

If you use or simply ❤️ `superpoint_transformer`

Leave us a ⭐on Github, it means a lot to us !

Before opening an issue

Have you thoroughly read our README, docs/, and notebooks/ ?
Have you looked at the documented code ?
Have you checked already-closed issues ?
Have you ***truly investigated*** the problem yourself beforehand ?
Have you modified the code (we only provide support for the code we released) ?
Are you using the latest version of the code ?

Writing your issue

Explain your issue in detailed and ***clear English***
Provide a ***minimum reproducible example***
Provide the ***entire error traceback***, if any
Make the most of Markdown to make your message for easily readable
Do not email me for issues, use GitHub

Writing a PR

We would gladly accept PRs for bug fixes, new functionalities, new datasets, new models

Discussion

💬