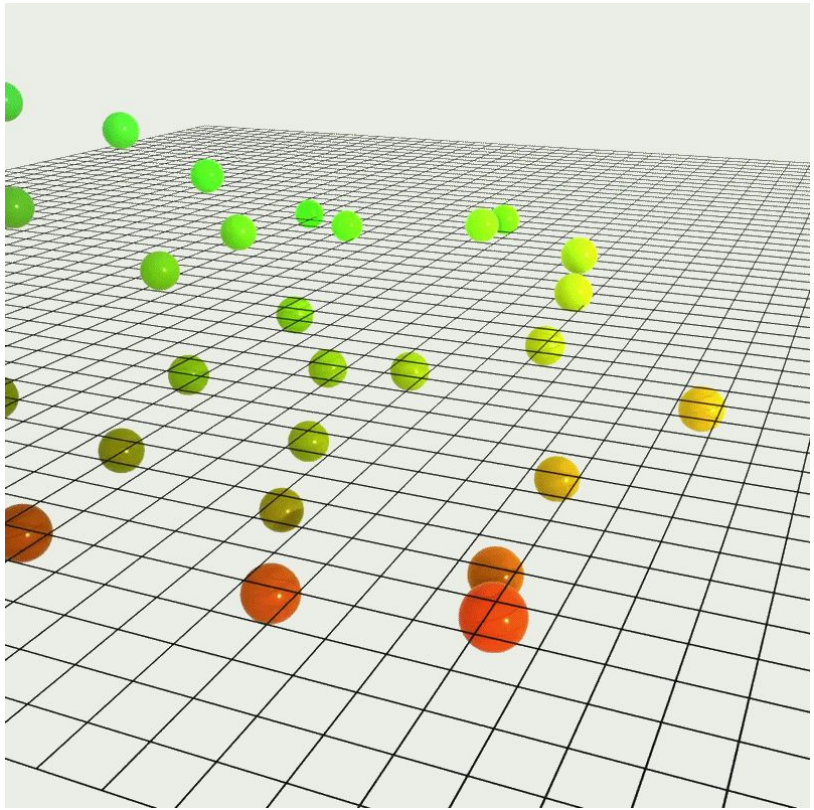
Unifusion 2.0

Unified continuous surface & semantic mapping with LLM-based knowledge gathering in outdoor environments

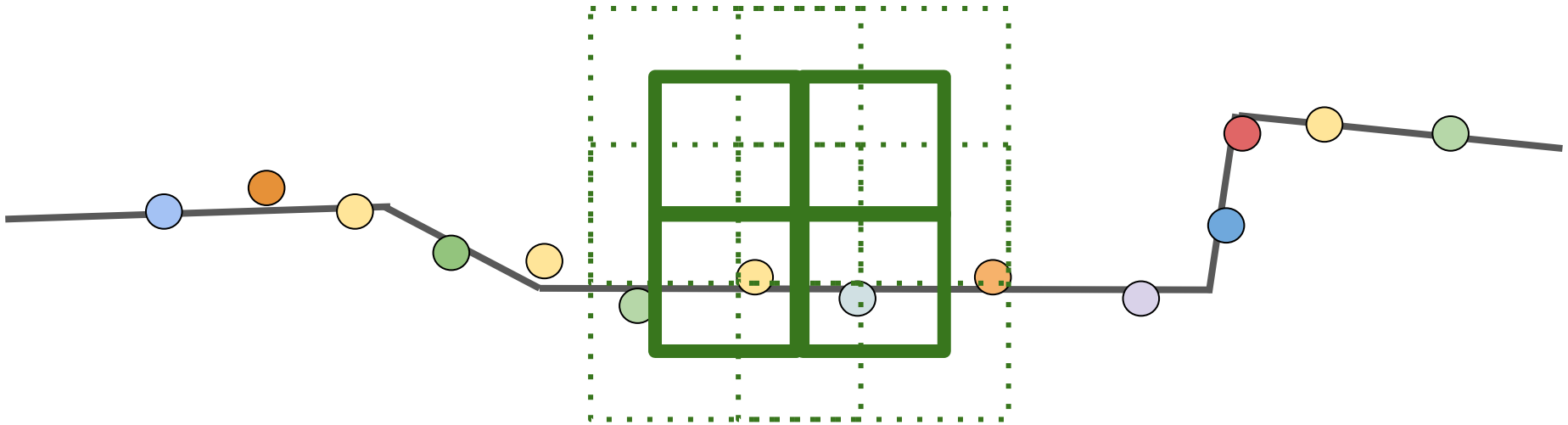
Universal Continuous mapping



Universal encoding

- Implicit representation using Nyström-approximated GPR via the Matern Kernel
- Voxel-wise calculation of feature vectors
- Continuity ensured using overlap
- Incremental refinement

Universal encoding



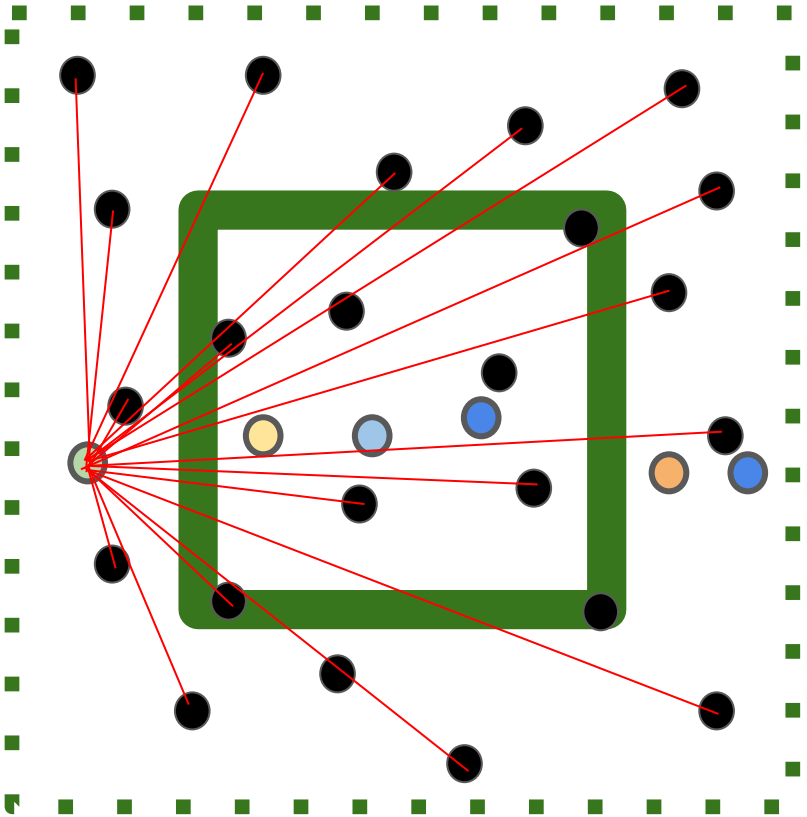
Universal Encoding

- 256 randomly distributed anchor points

- N data points

position encoding: Matern distance relationship between each data-point and each anchor point:

$$\phi(r) = \left(1 + r + \frac{2r^2}{5} + \frac{r^3}{15}\right)e^{-r}$$



Universal encoding

This yields a 256D Vector $\phi(x_i)$ for each point i that can be used to approximate the Kernel-function:

$$k(x_i, x_j) \approx \phi(x_i)^T \Lambda \phi(x_j)$$

where Λ is a one-time learned Matrix to down-project to 20D.

We will use $K = k(x_i, x_x)$ as the Kernel Matrix -> geometric similarity between observed points

Universal Encoding

GPR tells us that the best prediction μ at position x^* can be found like this:

$$\mu\left(x^*\right) = \phi(x^*)^\top \underbrace{\mathbf{\Lambda\Phi} \cdot \left(K + \sigma^2\mathbf{I}\right)^{-1}\mathbf{y}}$$

constant during inference => latent vector of this voxel

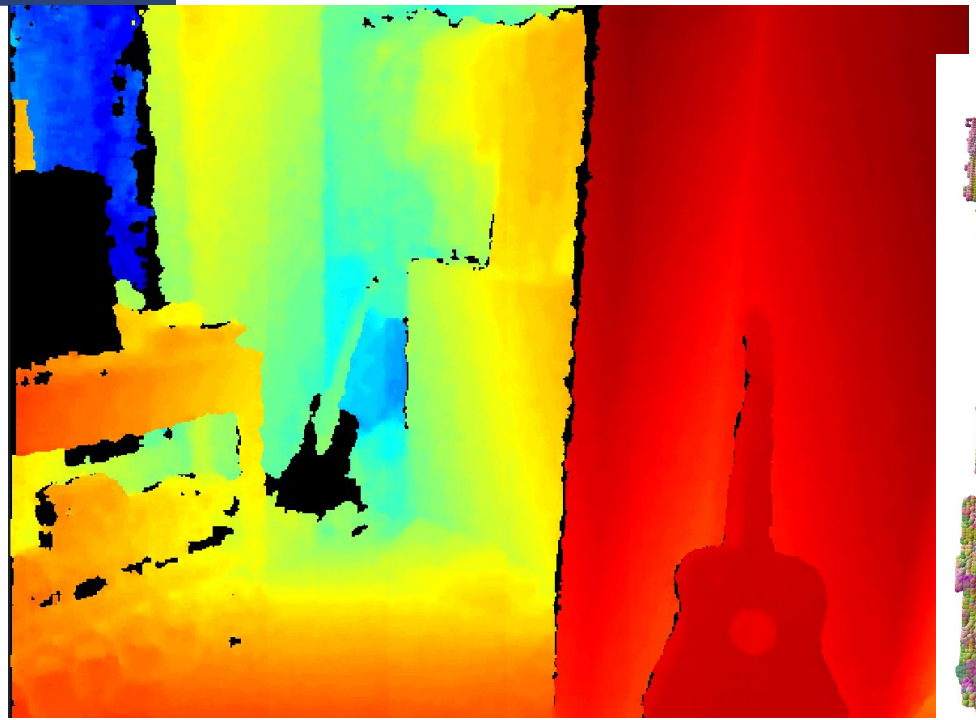
incremental refinement by continuously averaging this latent

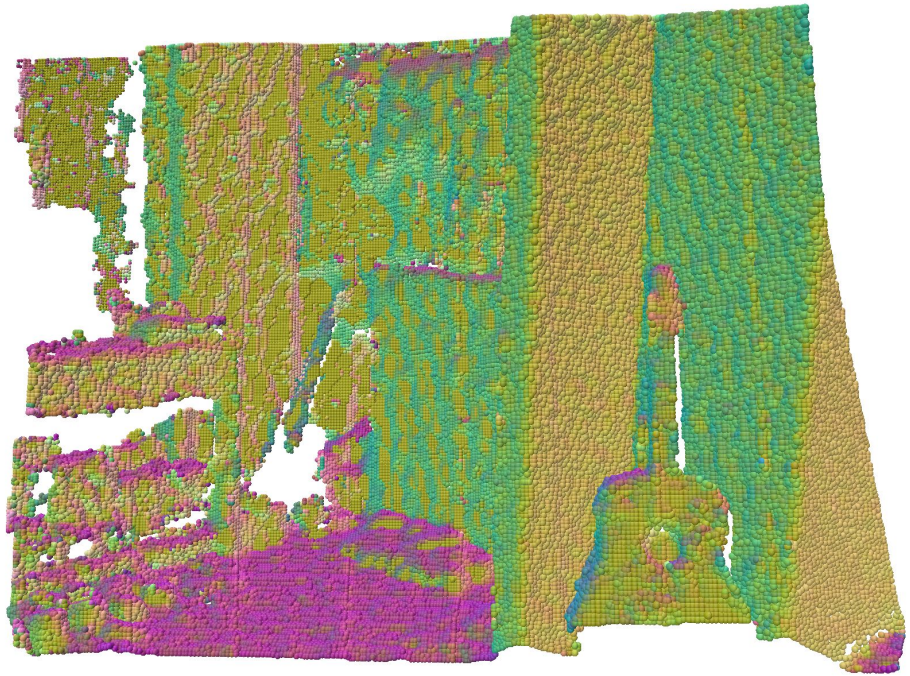
$\mathbf{y}$ being the matrix of data that is associated with each observed point

$\mathbf{\Phi}$ being the encoded position of all observed points

Surface Reconstruction

1. Normal reconstruction from depth



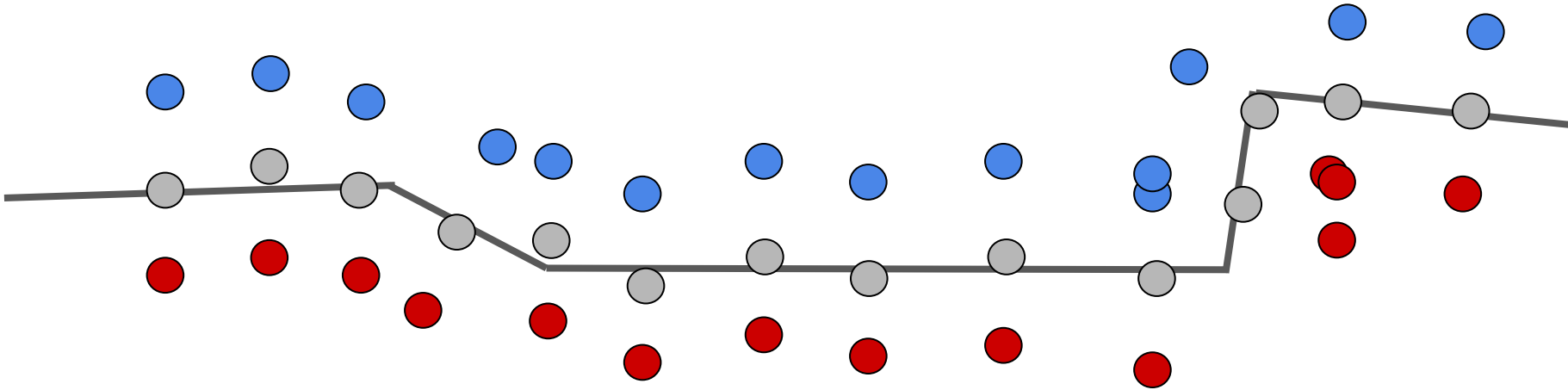


Surface reconstruction on a ScanNet depth image, using simple KNN normal calculation

Surface Reconstruction

2. Extrusion, building a SDF

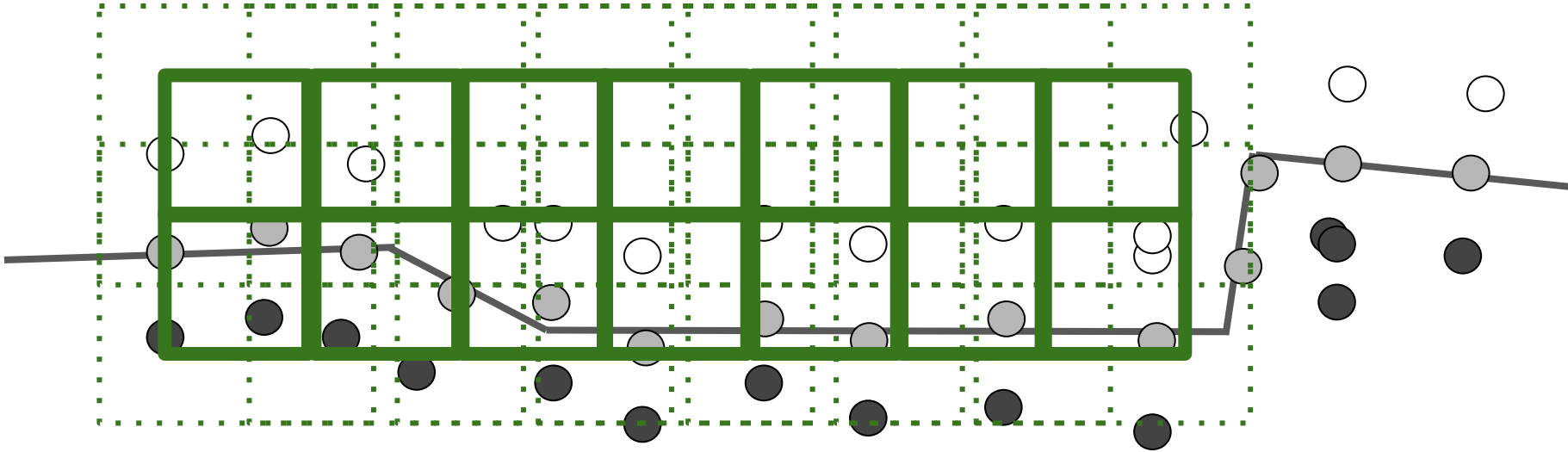
Create points extruded into the (inverse) normal direction, their values are +margin, 0, -margin



Surface Reconstruction

3. Assign to voxels

Each voxel has half its length overlap with its neighbours for continuity



Surface Reconstruction

4. Run the Universal Encoder storing this part per voxel

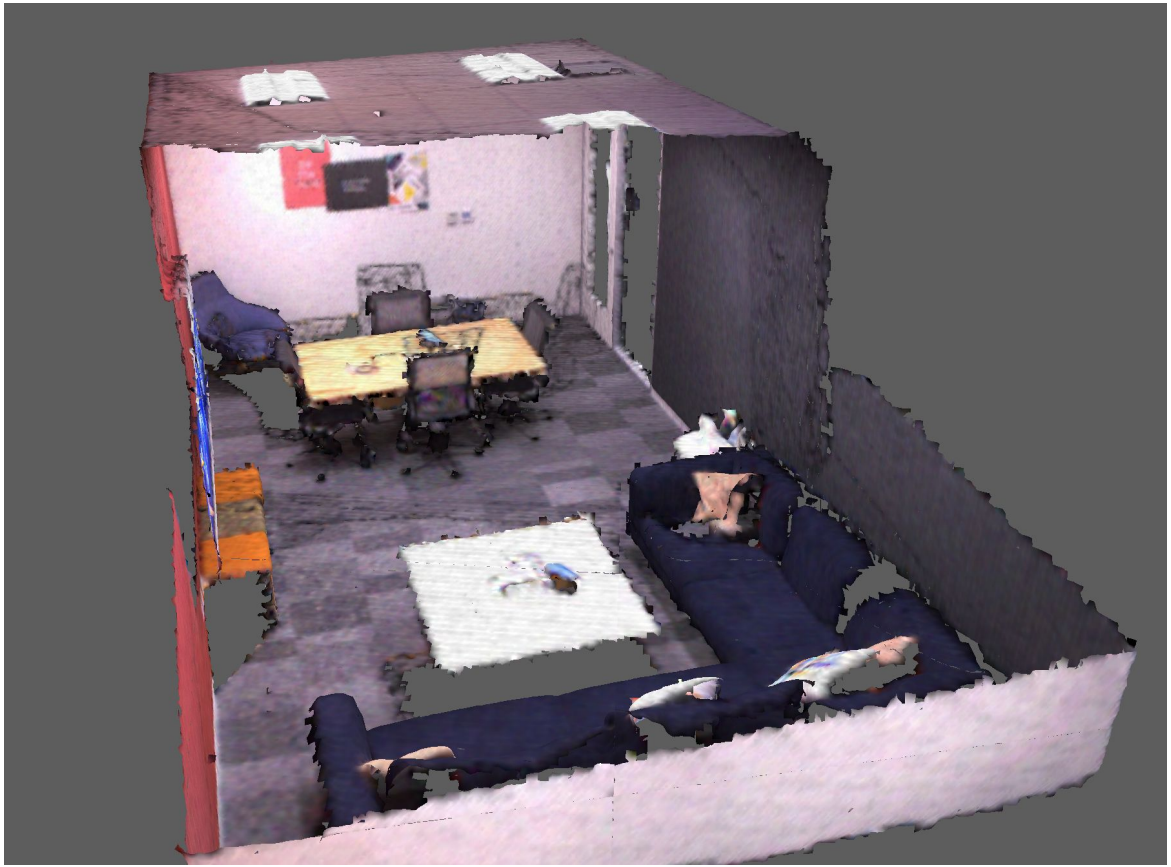
$$\mu\left(x^*\right) = \phi(x^*)^\top \overbrace{\mathbf{\Lambda\Phi} \cdot \left(K + \sigma^2\mathbf{I}\right)^{-1}\mathbf{y}}$$

5. Infer points in a semi-regular grid in all voxels that have data

=> yields sparse TSDF

6. Run Marching cubes on this TSDF

Surface Reconstruction



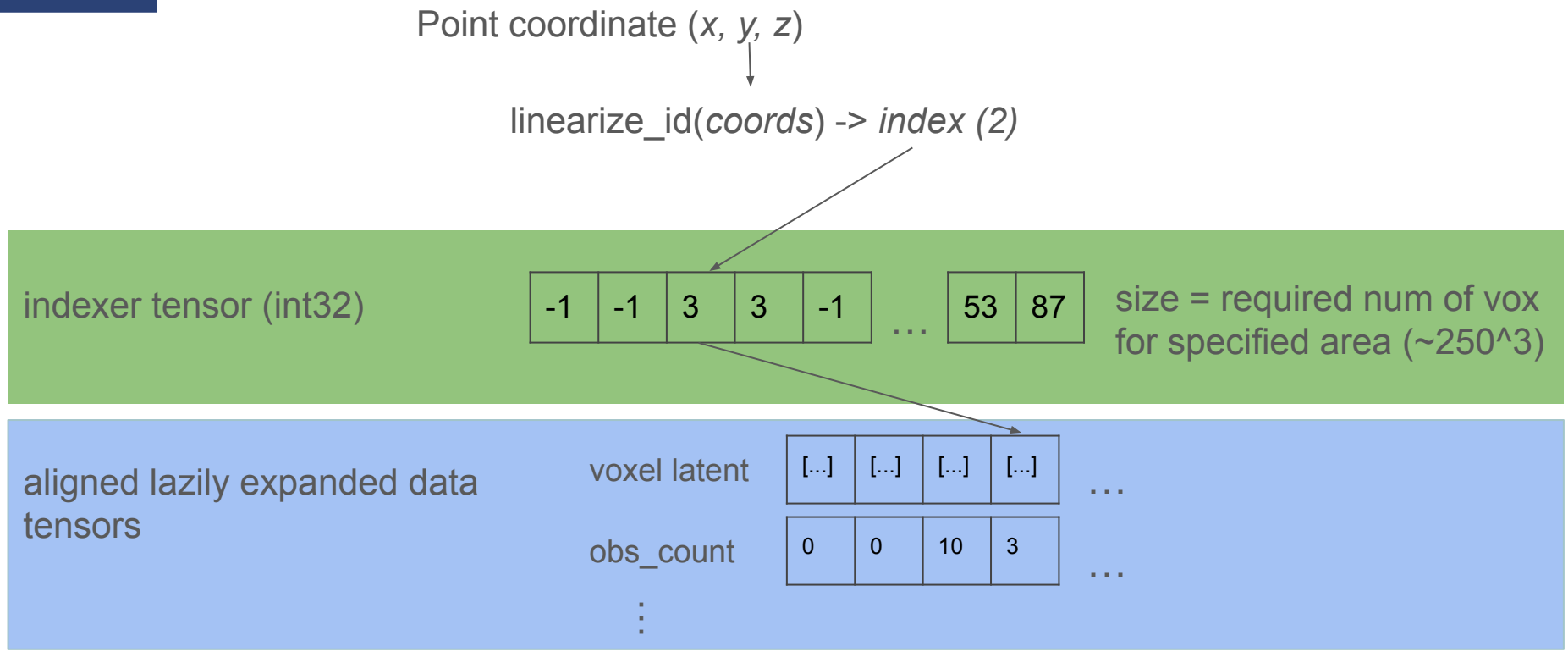
Replica-Dataset
GT or ORBSLAM-3 poses

~28 IPS on 5090
~14 IPS on 2080S

=> bottleneck somewhere

Data Structure

Voxels span limited volume => dense indexer



Data Structure

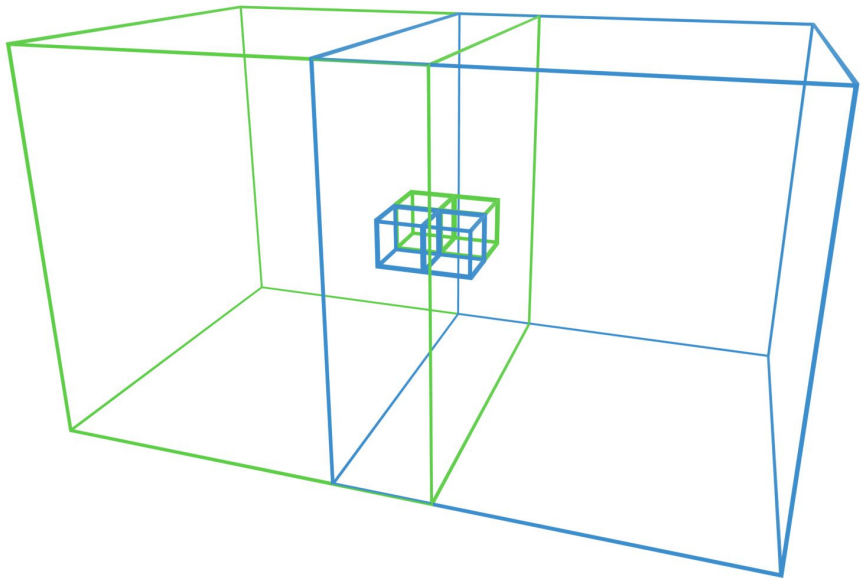
Voxels span limited area => dense indexer

+ simple, performant and memory-efficient

- requires limited and know spanning volume

- voxels can not be unloaded from VRAM

=> overlapping chunks of dense indexers



Data Structure

Chunk management

- 2 voxel overlap required for continuity => point & data duplication

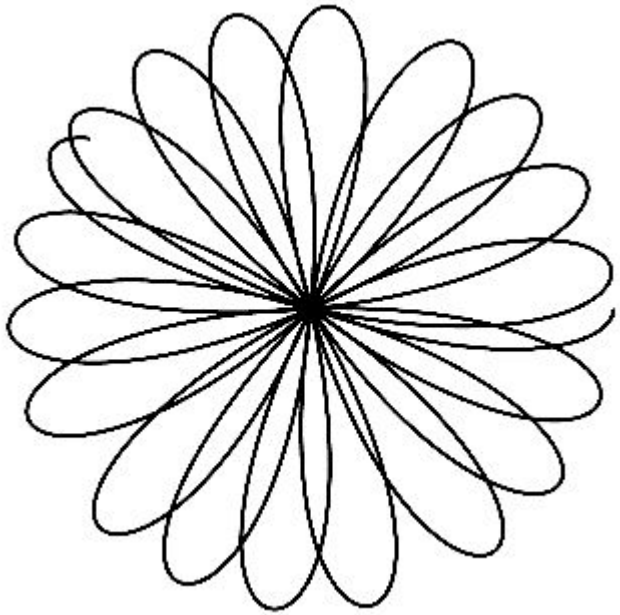
- single active chunk => not required chunks unloaded to RAM / Disk
  - quantization of latents of inactive chunks : `float32` => `int8` (range -2 to 2)
  - LRU eviction

=> volume & VRAM limitation lifted

Normal Calculation

Previously : direct calculation of normals from depth
-> not possible when the input is sparse point clouds

=> accumulation of points necessary

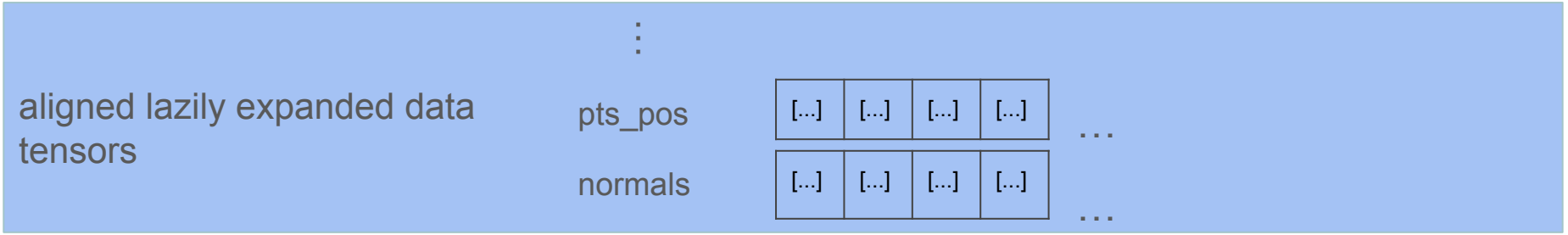


https://www.researchgate.net/figure/Scanning-pattern-of-Livox-LiDAR_fig1_344802339

Normal Calculation

Introducing additional position & normal tensors to the stored data per chunk
=> each voxel can store N (32) points

This same structure also allows for efficient NN searches => normal calculation



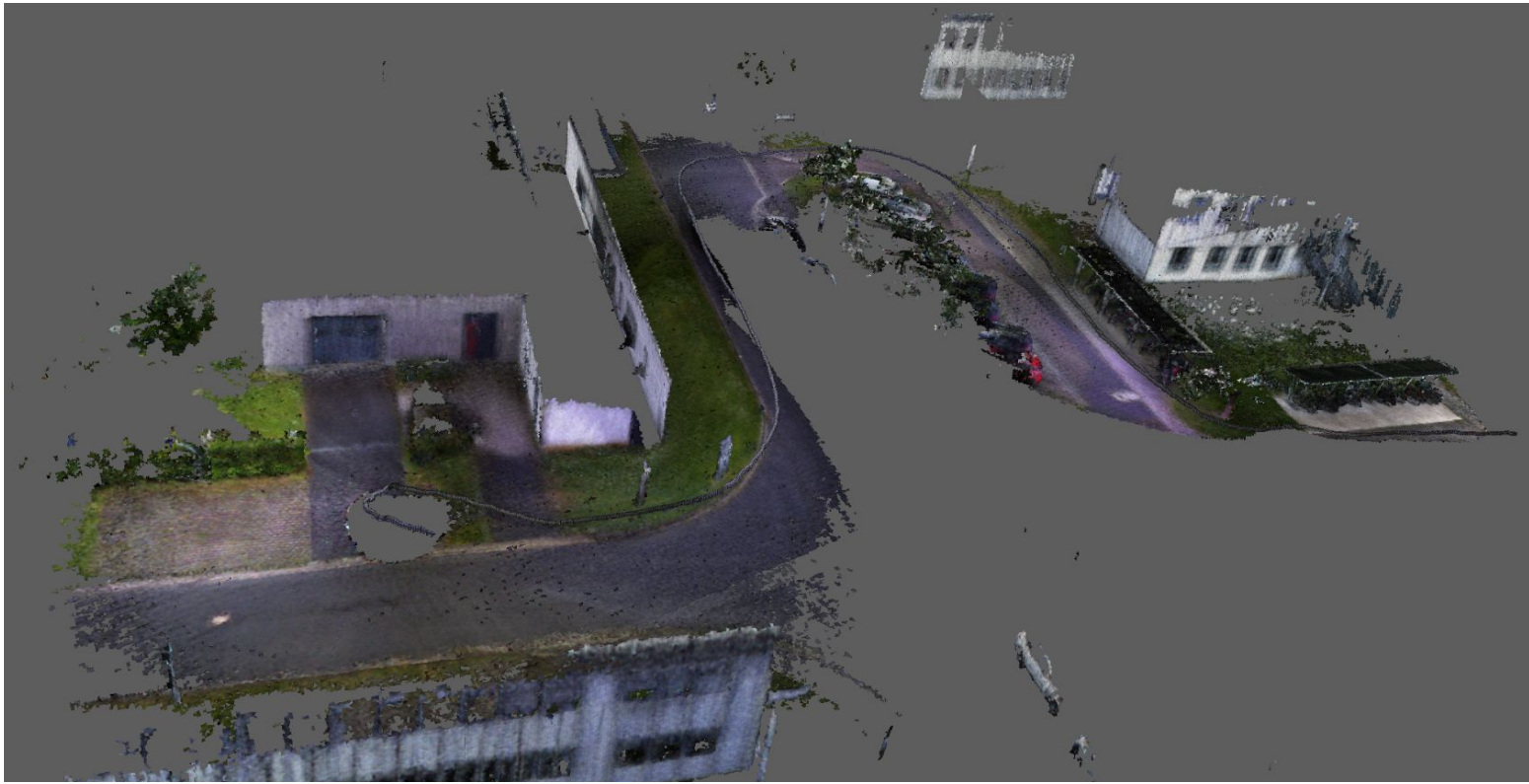
Custom handheld Scanner

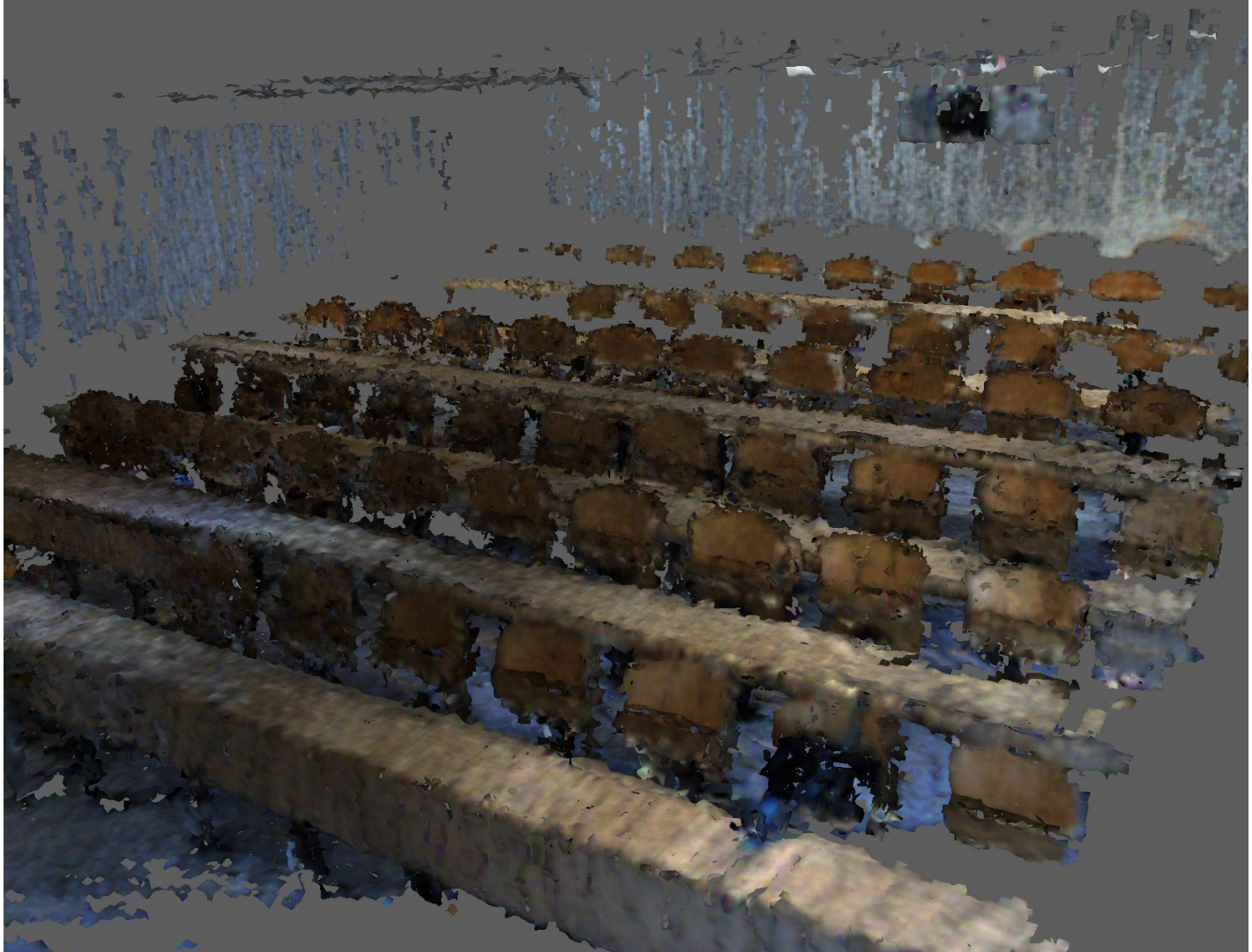
Stereo IDS global shutter camera
@30FPS

Livox Avia
@10FPS each 24k points

Small Capus Tour

ORBSLAM-3
for pose





Integrated Point-to-Plane ICP

Normal calculation + fast NN searches = 80% of PtP-ICP

No global relaxation, just a naive implementation matching the current scan to ∑ previous scans

Integrated Point-to-Plane ICP

Replica-Dataset:
(ATE) RMSE = 2.16 cm

Integrated Point-to-Plane ICP

Scannet-Dataset:
(ATE) RMSE = 1.34 m

Changes in architecture

Color inference during Surface Reconstruction

Previously two separate maps

- one encoding the 1D sparse TSDF
- one encoding the 3D RGB data

=> two encoding calls, after MC step sampling into Color encoder

Now: one 4D map

- sampling gets SDF+Color at the same time, MC immediately maps color to vertex

=> less calls, less compute, slightly more memory usage

Changes in implementation

Bucketed cholesky decomposition

$$\left(K + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{y}$$

During encoding, each voxel is solved independently, but all on CUDA in parallel.

torch’s cholesky solver requires dense Matrices, with a mask where no data lives

=> The memory complexity is still O(matrix complexity ^3)

Changes in implementation

Bucketed cholesky decomposition

$$\left(K + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{y}$$

=> The memory complexity is still O(matrix complexity ^3)

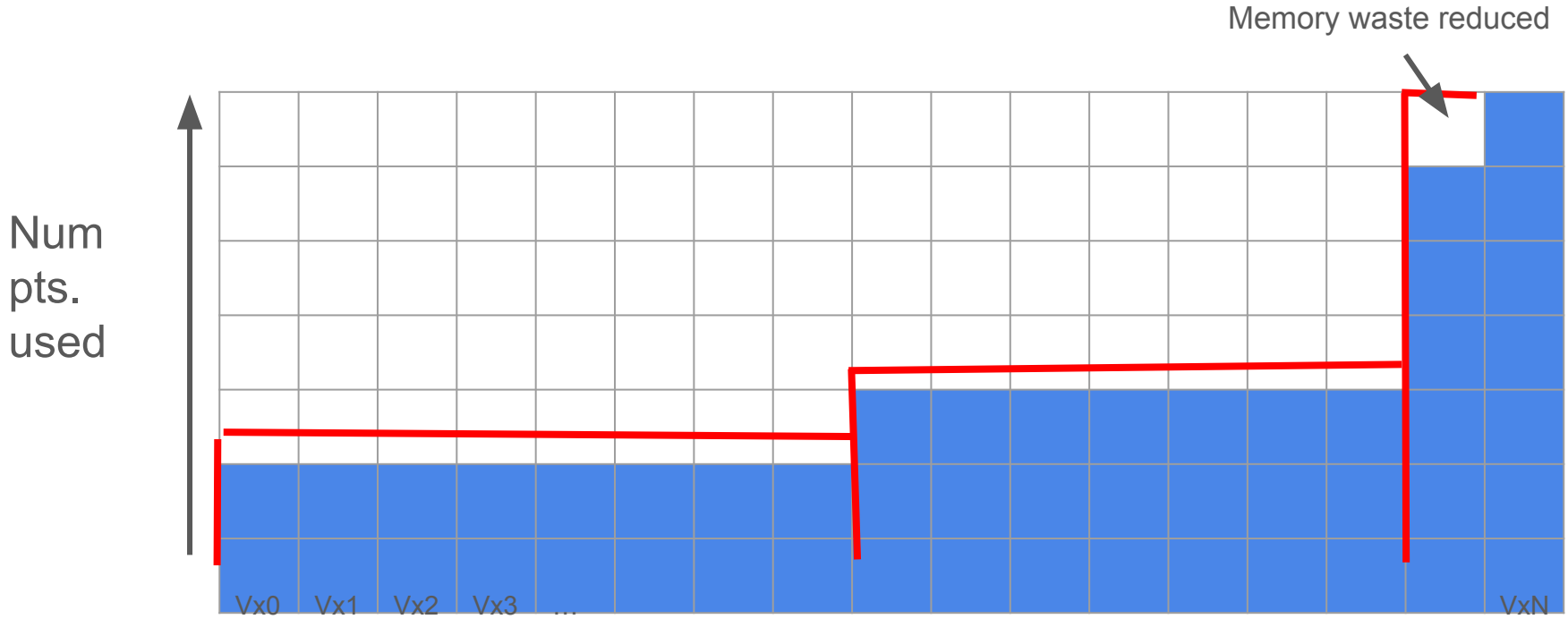
The number of points per voxel is not constant:

Changes in implementation

Bucketed cholesky decomposition

$$\left(K + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{y}$$

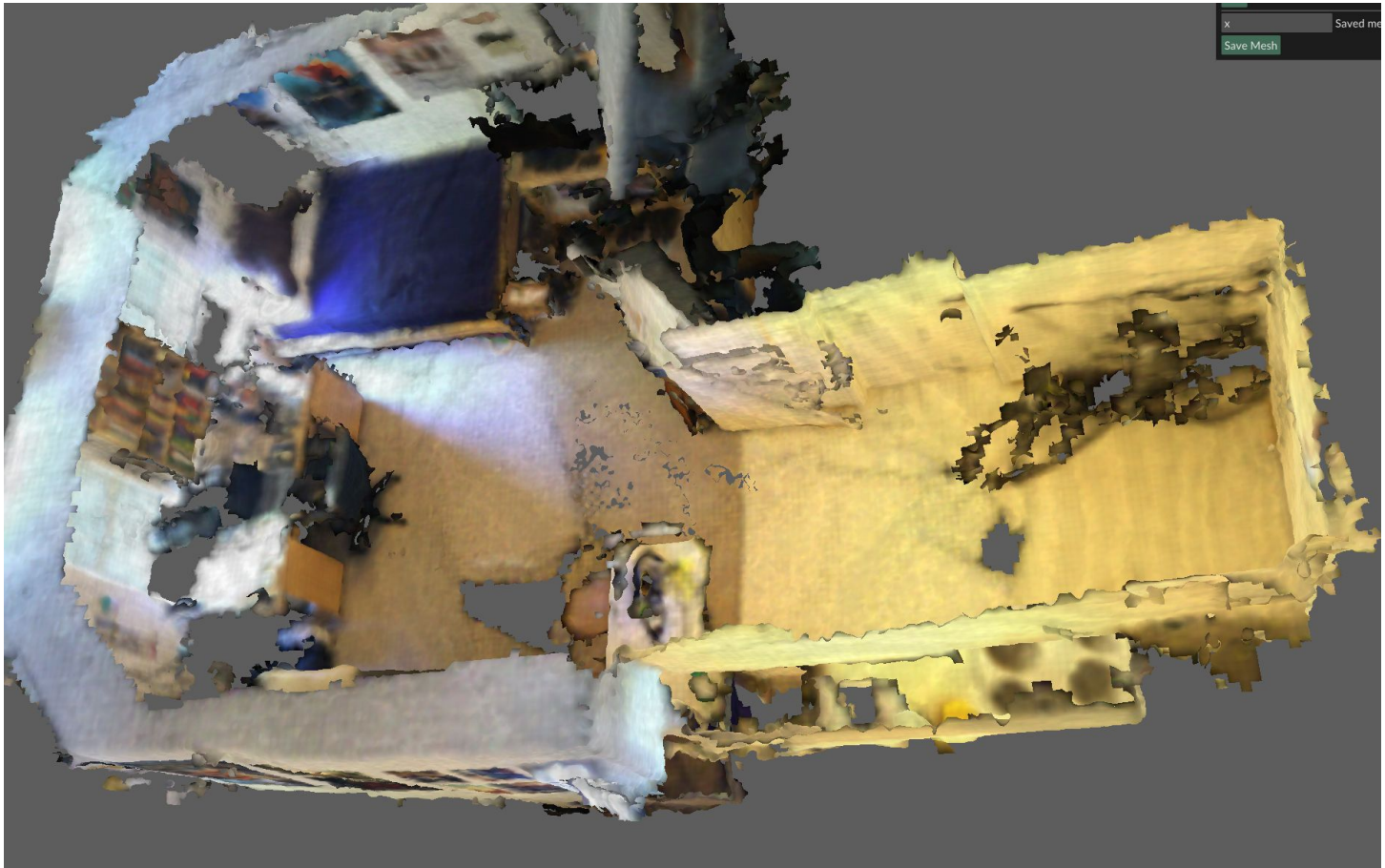
=> Call cholesky multiple times in groups of the ~same size:



Surface Reconstruction Evaluation on ScanNet

Mesh Accuracy (Median Error):
Average (mean): 1.65 cm
Median: 1.43 cm

Mesh Completion (Median Error):
Average (mean): 1.39 cm
Median: 1.30 cm

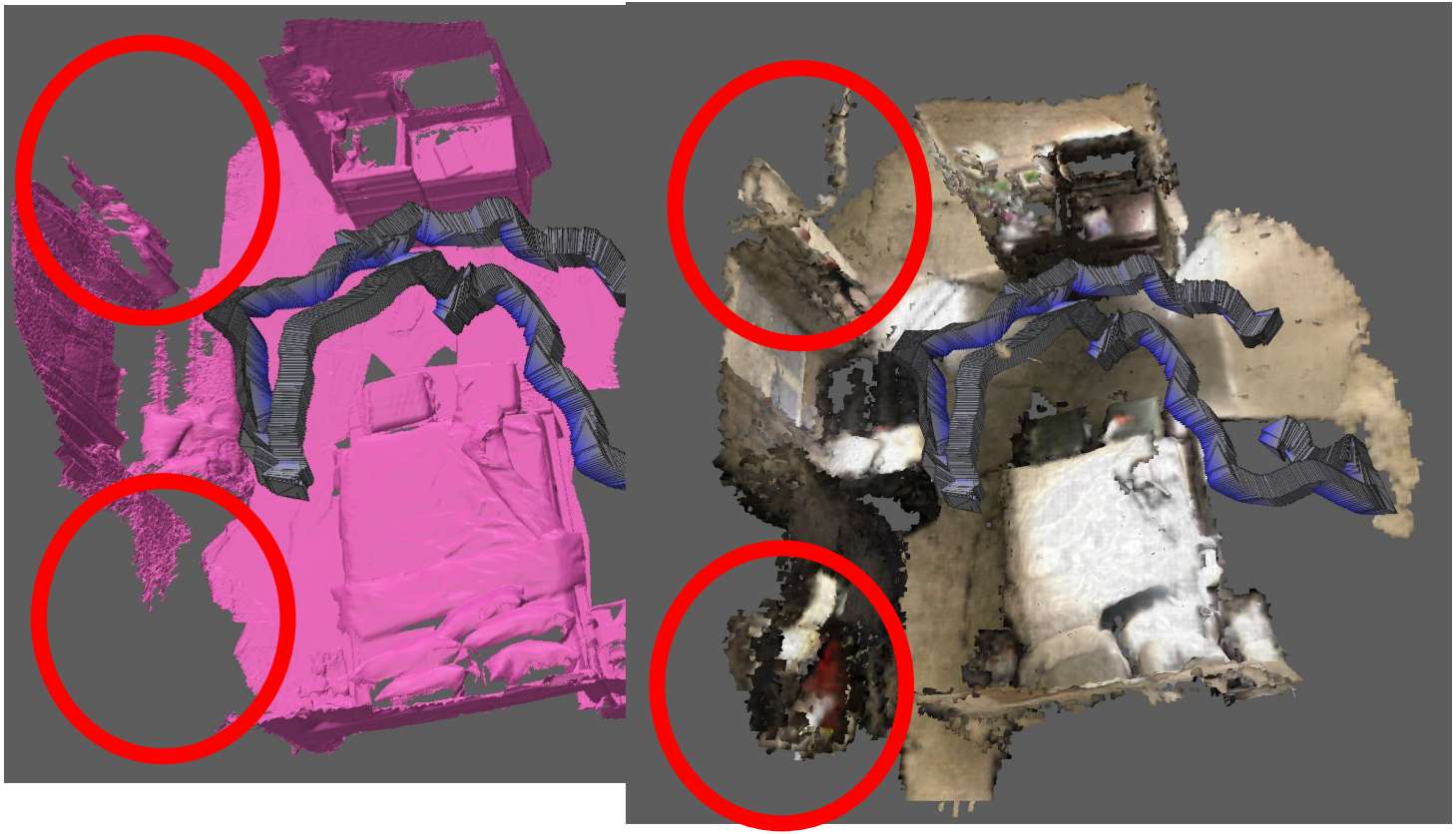


Surface Reconstruction Evaluation on ScanNet

```
Precision (<5cm):
Average (mean): 86.84%
Median: 90.15%

Recall (<5cm):
Average (mean): 98.38%
Median: 99.02%

F-Score (<5cm):
Average (mean): 91.86%
Median: 94.39%
```



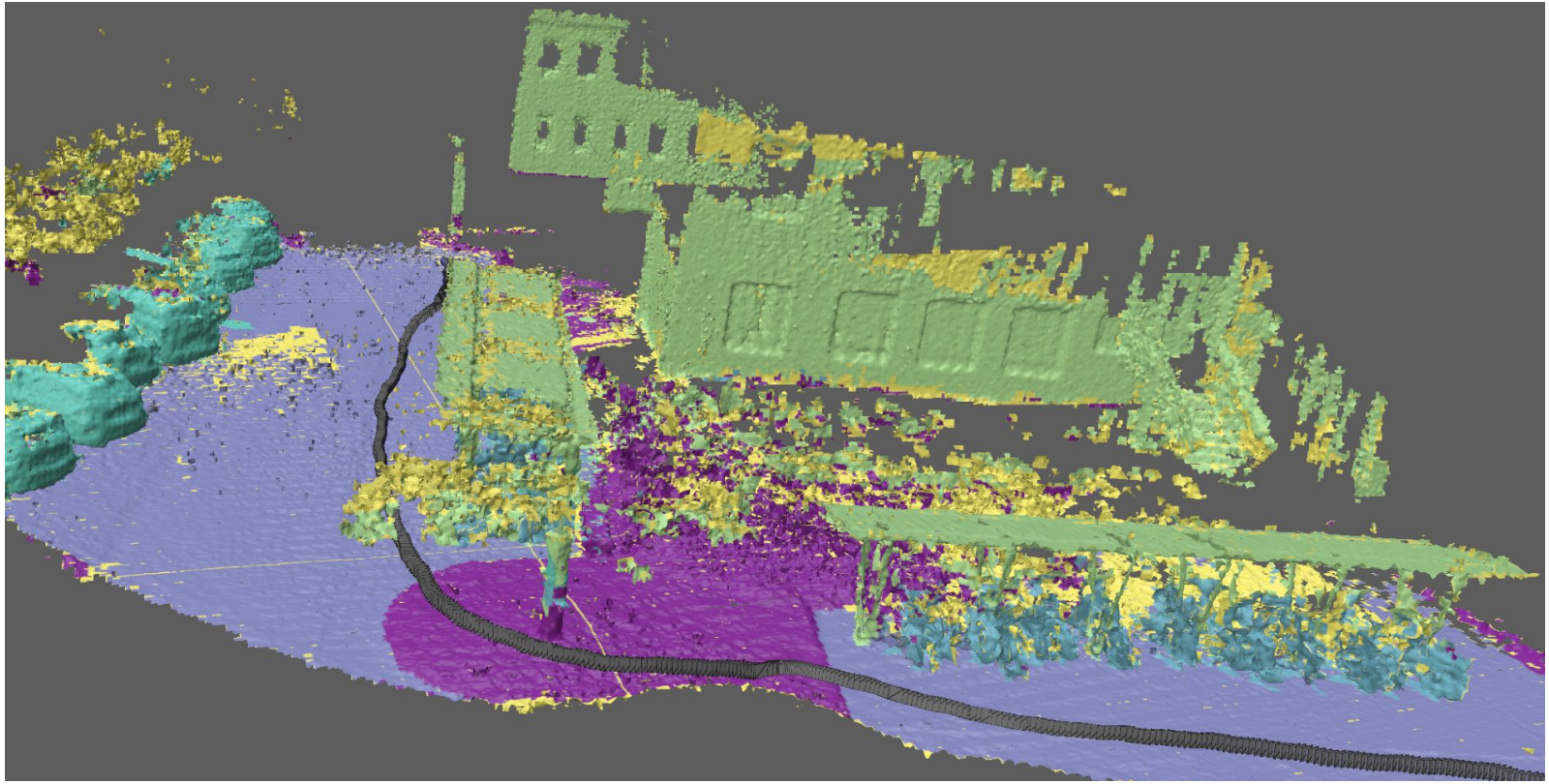
Semantic mapping

CLIP reprojection & encoding

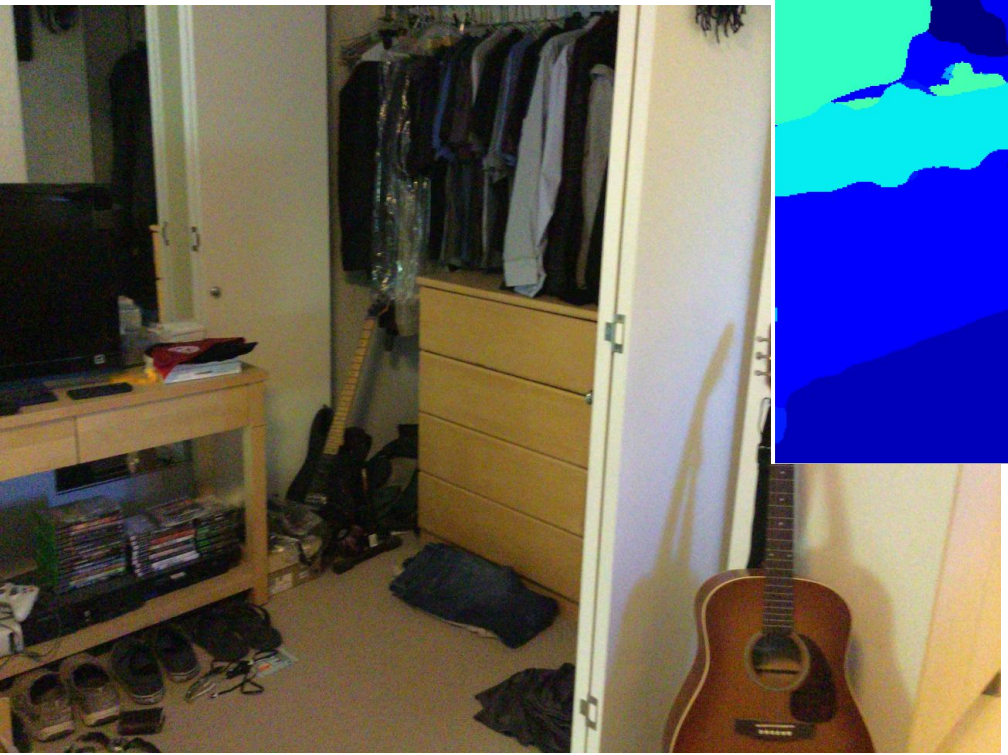
1. Encode input image into CLIP 512D - “pixels”
   tested: LSEG-minimal, CLIP-Dino, FC-CLIP, SC-CLIP, SegCLIP
2. Render a depth image at that images pose from the current Mesh
3. Reproject CLIP “pixels” into 3d using it
4. Encode these points using the Universal encoder
5. Infer at points of interest, compare to CLIP-features of label-set

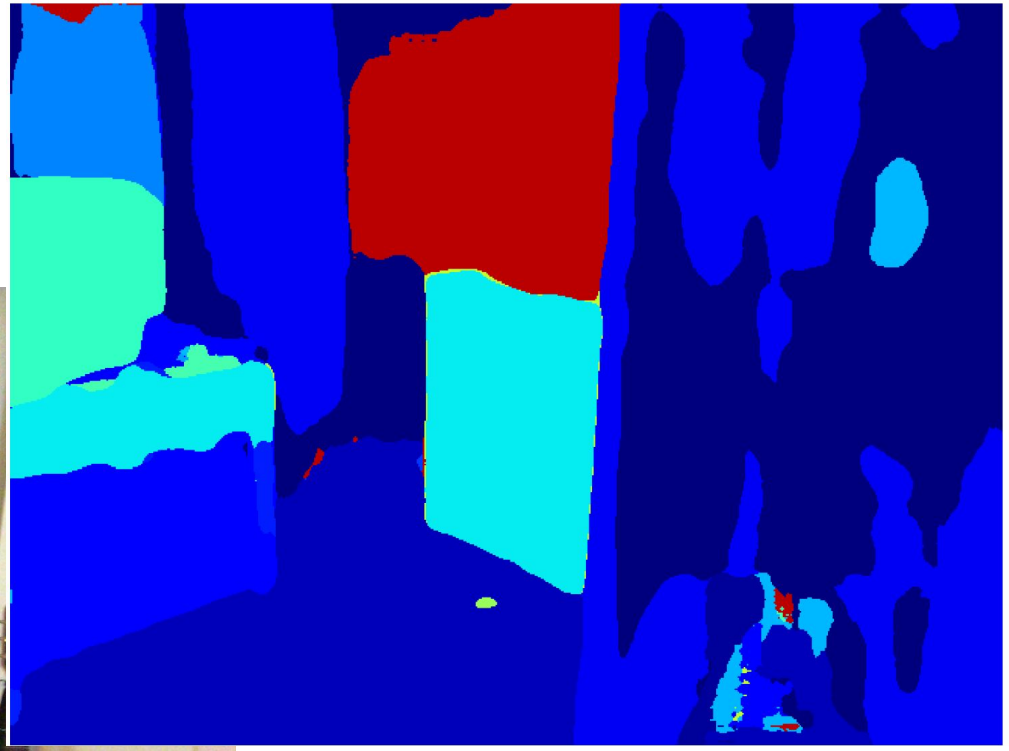
Semantic mapping

labels:
-grass
-car
-bicycle
-building
-road



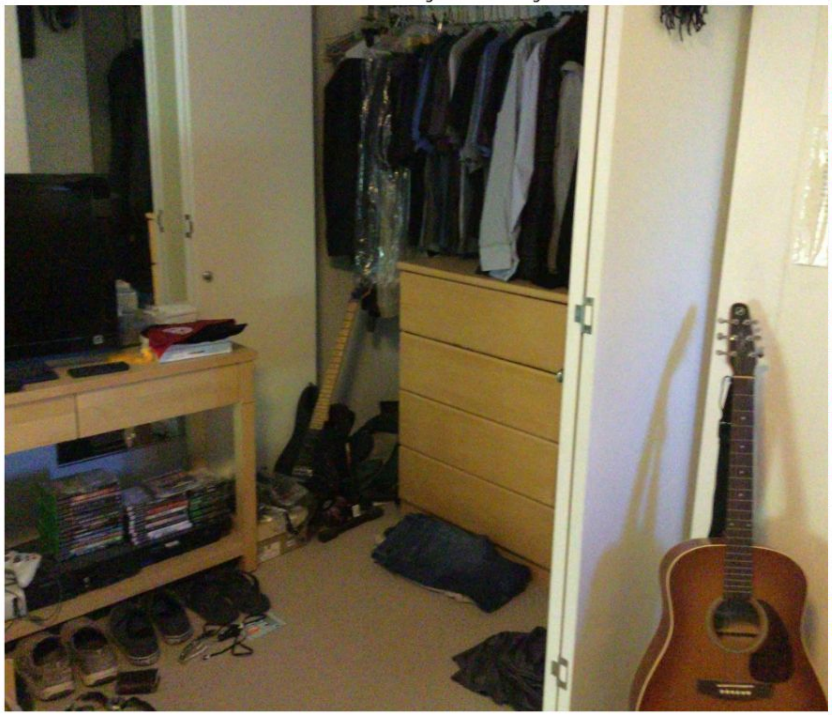
LSeg





Avg IoU:	31.06%
Head IoU:	36.06%
Common IoU:	25.06%
Tail IoU:	0.00%

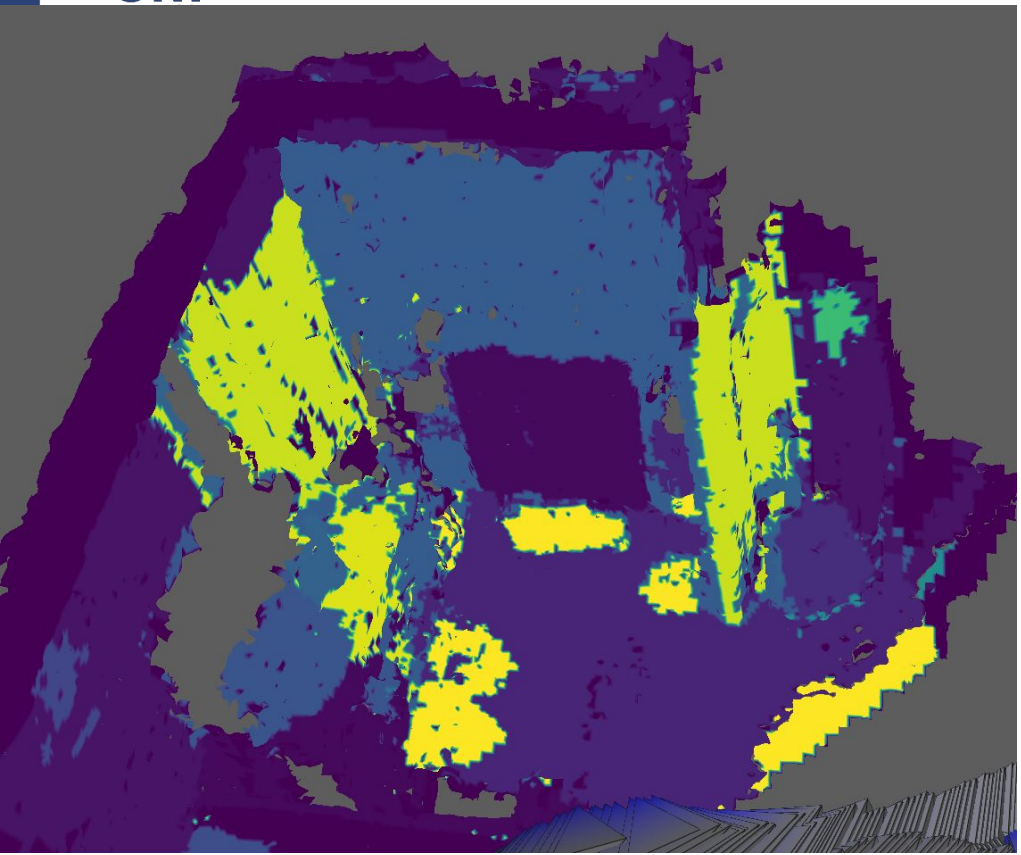
SC-CLIP


Original RGB Image


Categories
wall
chair
floor
couch
door
couch
shelf
desk
bed
pillow
sink
window
bookshelf
box
refrigerator
kitchen cabinet
towel
tv
counter
dresser
stool
ceiling
backpack
trash can
closet
microwave
stove
shoe
laptop
recycling bin
guitar
bicycle
paper towel roll
doorframe
dish rack
range hood
laundry hamper
dumbbell
clothes
Unknown


Categorical Mask

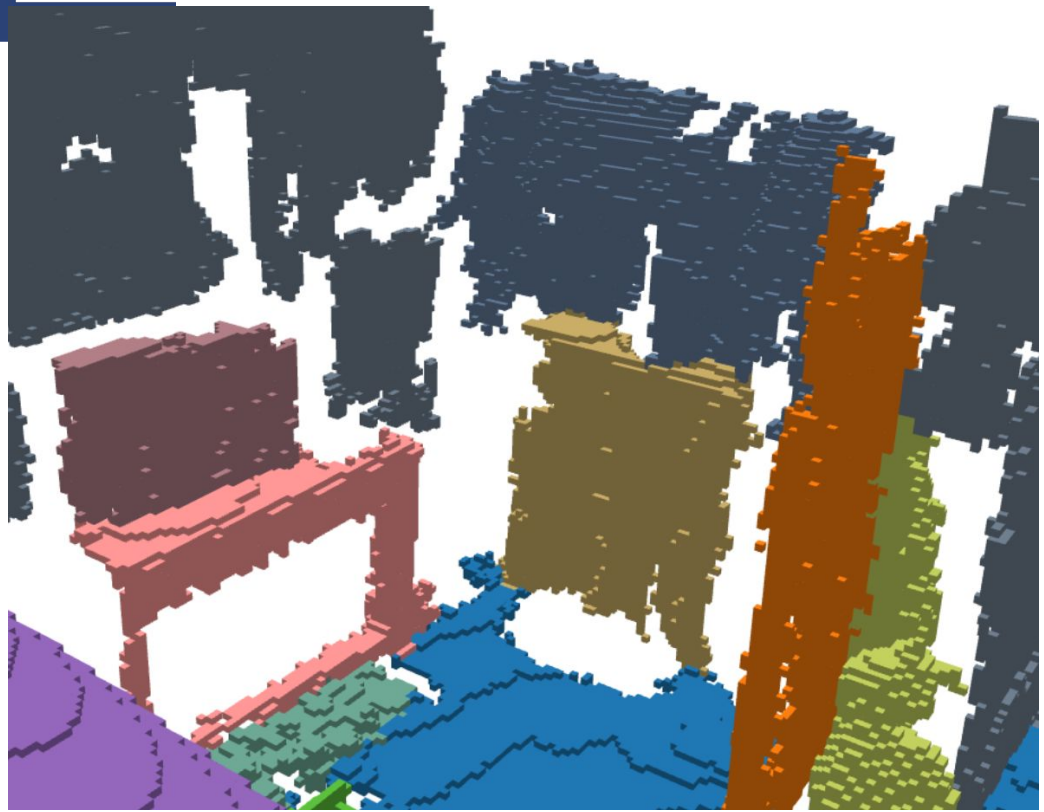
SC-CLIP



- couch[0]
- box[1]
- wall[2]
- window[3]
- floor[4]
- guitar[5]
- sink[6]
- bed[7]
- counter[8]
- pillow[9]
- tv[10]
- closet[11]
- door[12]
- stove[13]
- bookshelf[14]
- shelf[15]
- desk[16]
- laptop[17]
- stool[18]
- chair[19]
- refrigerator[20]
- microwave[21]
- kitchen cabinets[22]
- ceiling[23]
- trash can[24]
- backpack[25]
- poster tube[26]
- recycling bin[27]
- towel[28]
- dish rack[29]
- paper towel roll[30]
- bananas[31]
- range hood[32]
- dumbbell[33]
- bicycle[34]
- doorframe[35]
- dresser[36]
- laundry hamper[37]
- shoes[38]
- Uncategorized[39]

Avg IoU:	25.11%
Head IoU:	29.68%
Common IoU:	10.67%
Tail IoU:	17.74%

ScanNet Ground Truth



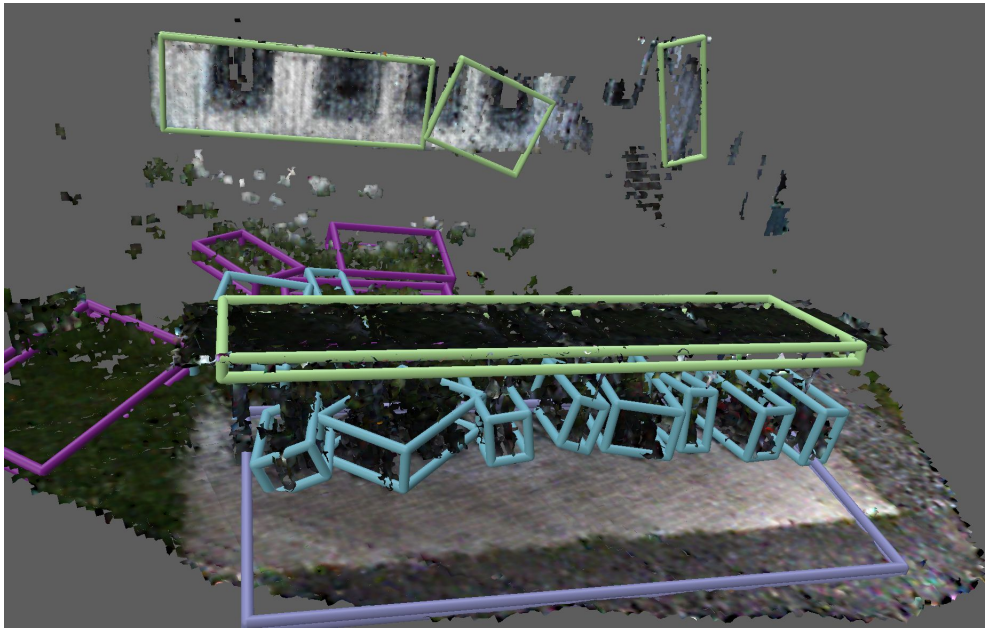
Benchmark-Leader:
DITR

Avg IoU:	49.9%
Head IoU:	62.9%
Common IoU:	39.2%
Tail IoU:	28.9%

LLM Integration

- Finding continuous surface regions in interested labels
- Calculate bounding boxes, splitting them to avoid over-coverage

- Feed Bounding Box vertex coords to LLM



LLM Integration

Abridged Question: are the bicycles covered by anything?

LLM used: Deepseek “Instant”

Step 3: Conclusion

Yes, **the bicycles are covered** by nearby building structures. In both clusters, there is a building directly above the bicycles in the vertical axis, with horizontal overlap.

Future Work: Feedback - Loop to set the search-categories