Point Cloud Autoencoders for Fast, Globally-Accurate 3D Mapping

Mihir Garimella, Prathik Naidu
{mihirg, prathikn}@stanford.edu

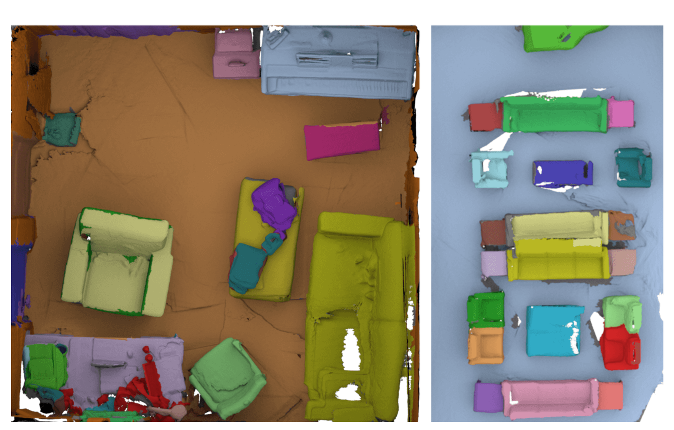
Introduction

- 3D mapping is a key building block for vision applications from AR/VR to autonomous driving
- Current methods don't work well in large scenes or on embedded processors for two reasons:
  - **Memory** required to store map grows rapidly with size of scene
  - **Drift** builds up as small errors in the map accumulate, and correcting it typically requires matching every new frame with all past frames
- **We present a series of novel deep learning architectures to enable building globally-accurate 3D maps of large scenes in real-time**

Related Work

- **SegMap** (Dubé et al., 2018) presents a learned descriptor for voxel grids that can be used for compression and drift correction ("loop closure")
  - *Limitations:* voxel grids lose resolution, network is never trained explicitly for loop closure task

Dataset



- **ScanNet** includes 1,513 indoor scene scans with camera poses, instance + semantic segmentation labels
- *Preprocessing:* Group "chunks" of *k* consecutive RGB-D frames, convert to point clouds, separate each chunk into individual object point clouds

Technical Approach

① Encoder

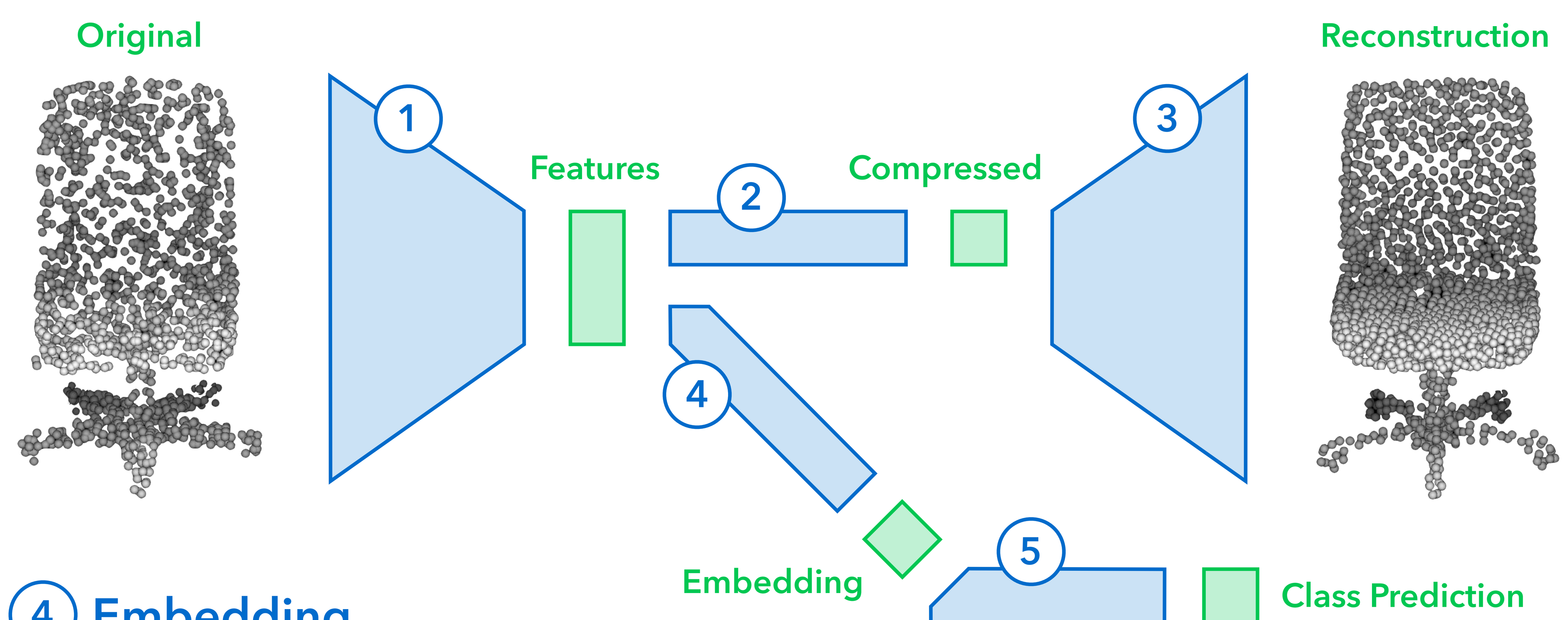
Using a **PointNet++ module** (Qi et al., 2017), we **encode each point cloud** into a single feature vector

② Compression

Using an MLP, we **further compress the encoder's output** to be compactly stored in memory

③ Decoder

Using a **coarse-to-fine decoder** inspired by PCN (Yuan et al., 2018) + **geometric reconstruction loss**, we **decompress back into full point clouds** on-demand



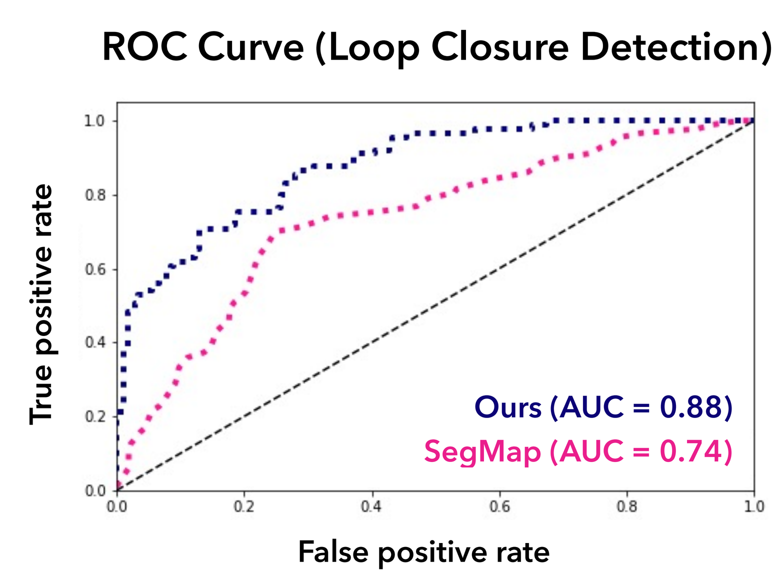
④ Embedding

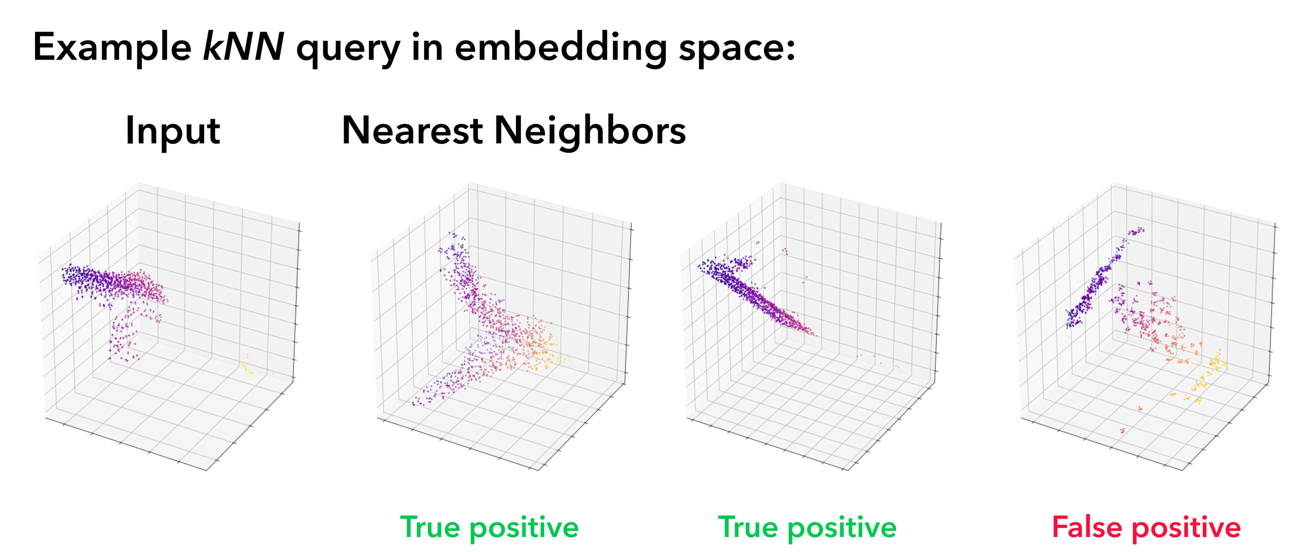
Using an MLP + **triplet loss**, we **embed the encoder's output** into a vector space over which L_2 distance represents geometric similarity, allowing us to **detect loop closures**

⑤ Classification

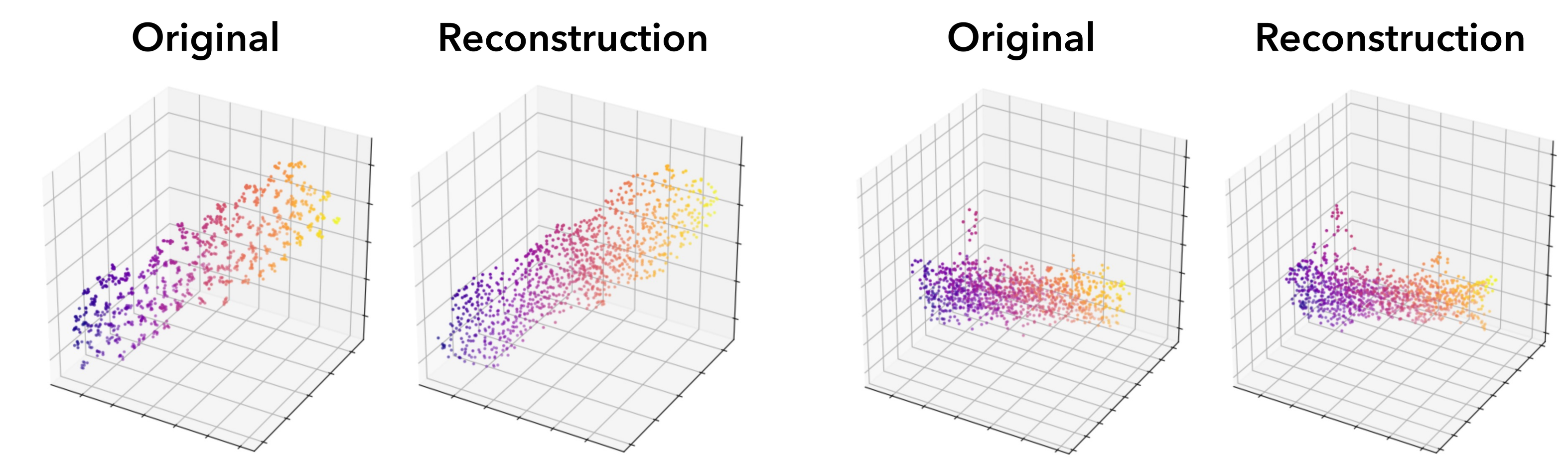
Using an MLP + **cross entropy loss**, we regress object class labels (e.g., "TV," "sofa," "table") to **force our embeddings to be semantically-meaningful**

Results (Embedding)

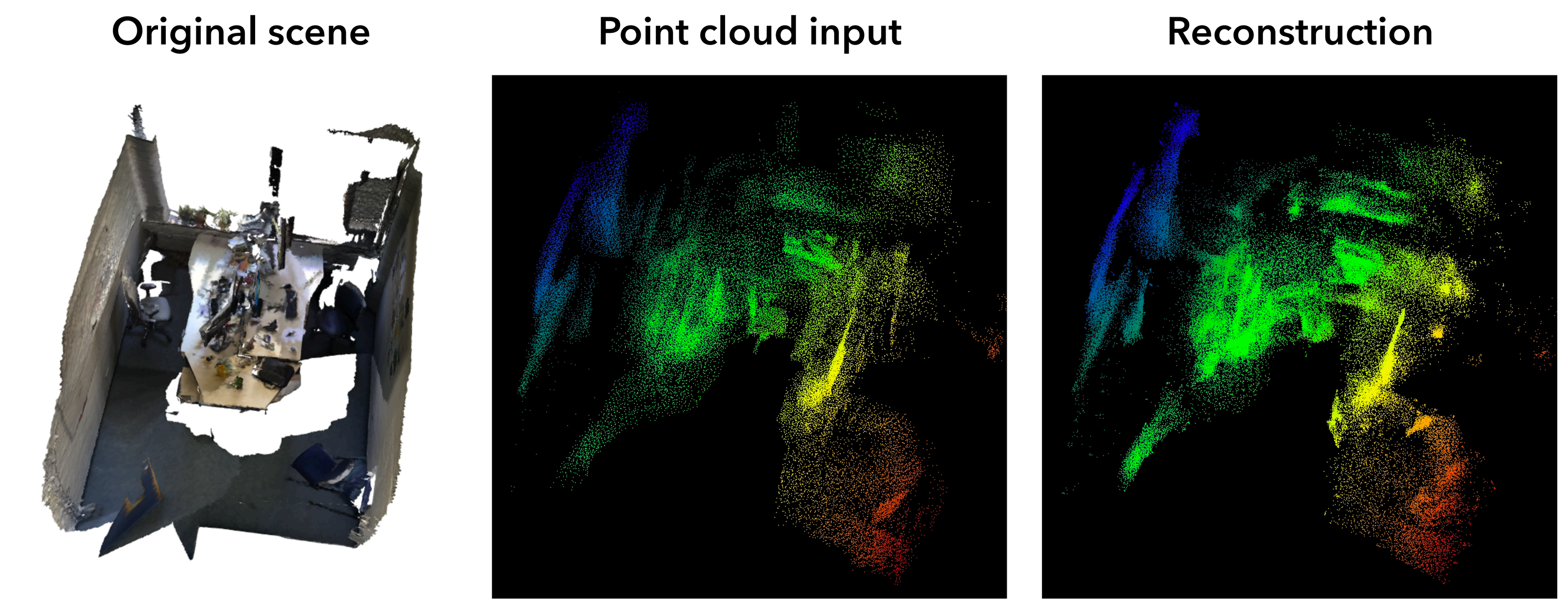




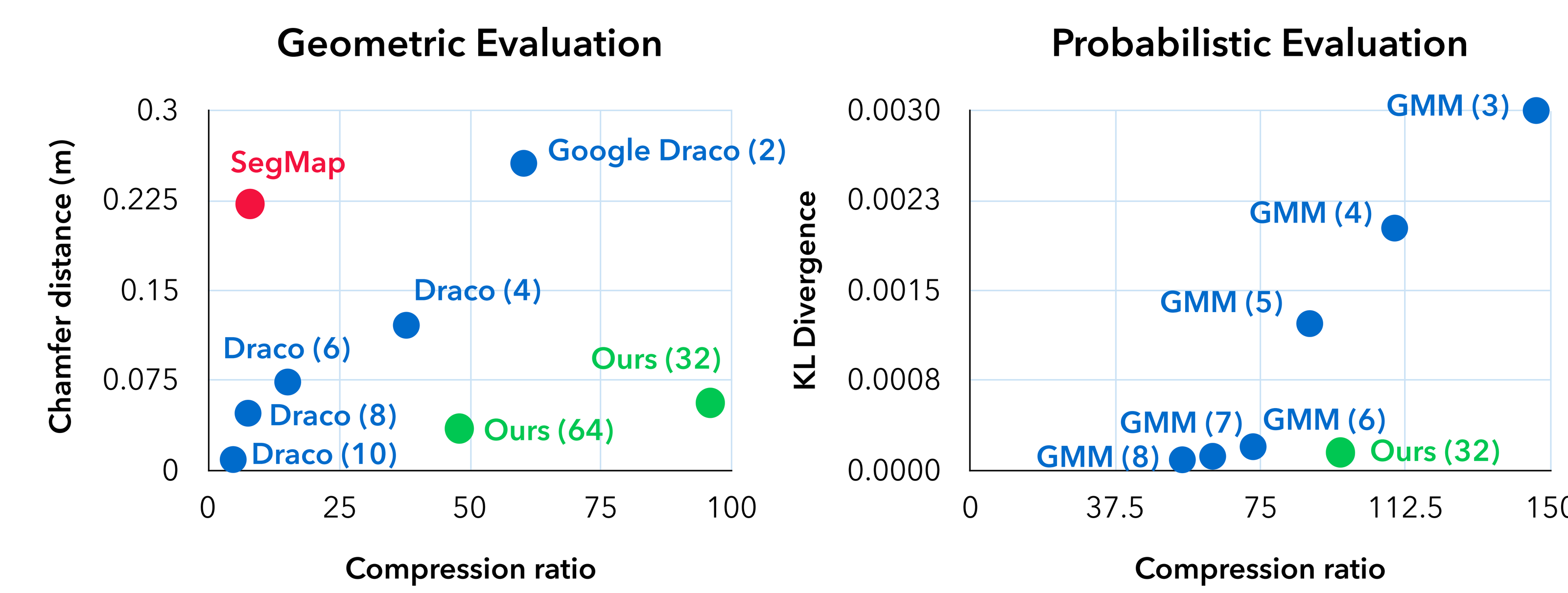
Results (Compression)

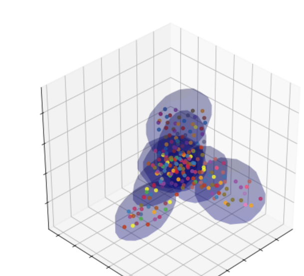


1024 input points × *(x, y, z)* → network bottleneck of size 32 = **96× compression ratio**



Stitching together individual reconstructed objects, we see that our network **preserves the overall geometric structure** of a scene, making it useful for real-time 3D mapping





We fit GMM's (←) to the original and reconstructed point clouds and use a Monte Carlo simulation (Hershey & Olson, 2007) to approximate D_{KL}. Then, we reduce the number of GMM's ("compressing" the data) and benchmark our method against GMM-based compression.

Future Work

- Integrate work into full 3D mapping pipeline for more rigorous evaluation
- Develop end-to-end deep learning-based SLAM