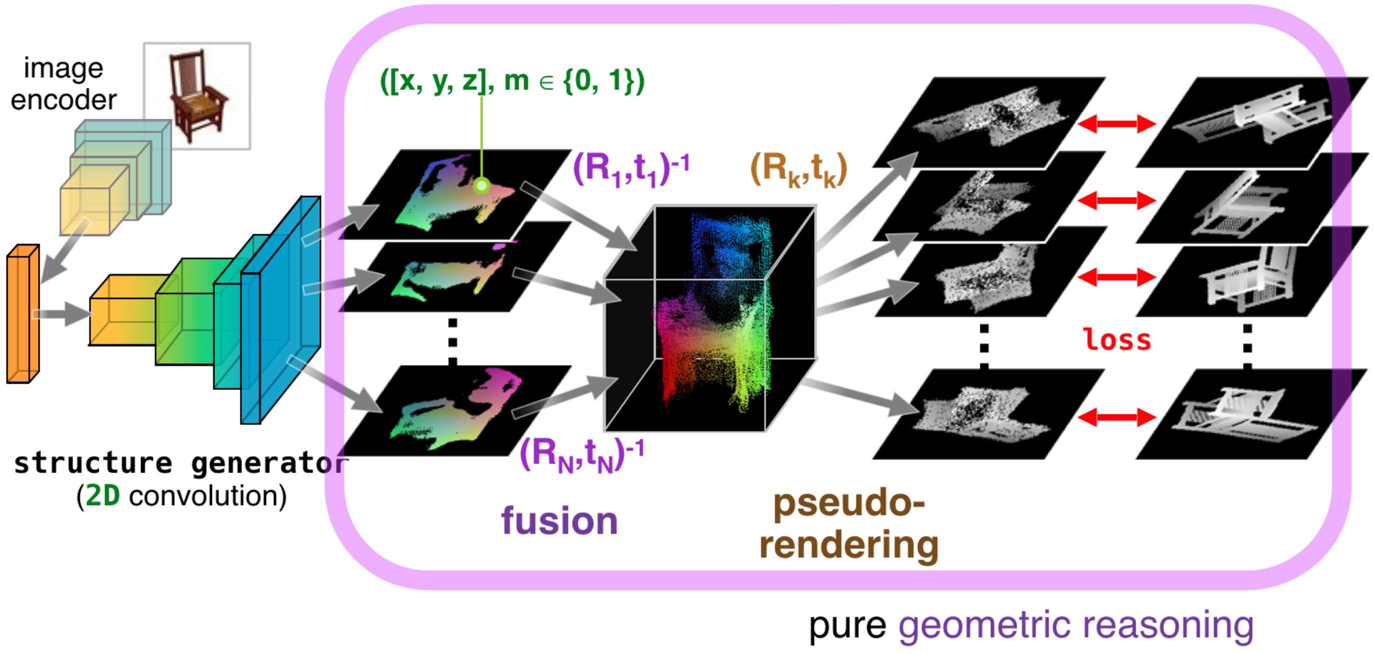
Learning Efficient Point Cloud Generation for Dense 3D Object Reconstruction

Report



Dataset used: Shapenet
Pytorch Code:
https://github.com/lkhphuc/pytorch-3d-point-cloud-generation
Paper: https://arxiv.org/pdf/1706.07036.pdf

Key Points to note:

- They are completely using 2D convolutions instead of 3D convolution since 3d convolution is a waste of memory and it captures a lot of useless features. Basically, the volume is not important, the surface which separates the object from the space is important. So we exploit that using 2D convnets.
- Instead of evaluating the model wrt to 3D volume, we evaluate instead on the projections of the 3D volume, i.e we evaluate using 2D images. If we evaluate in 3D volume it consumes a lot

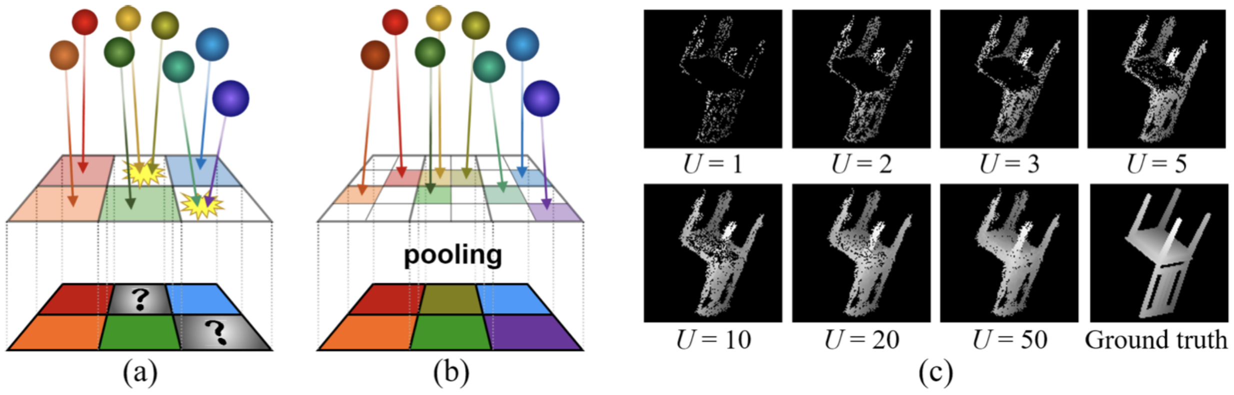
of computation time since we are comparing each pixel and hence O(n^3) but if we use 2d images then it will be O(n^2), which is so much time-efficient

Structure Generator

- The decoder part of the model which is called the structure generator outputs 8 different viewpoint images which then are merged to form the 3D images.
- The generator predicts N = 8 images of size 128×128 with 4 channels (x, y, z and the binary mask), where the fixed viewpoints are chosen from the 8 corners of a centered cube.

Pseudo Renderer

- Instead of evaluating over model over the 3D predicted model, we predict using the projected images of the 3D model. This is done by a pseudo renderer.



- So basically the pseudo renderer projects the 3D object into different 2d planes. So while doing this many of the 3D pixel points collides when getting projected onto a plane
- So to go about this we upsample the projection plane and hence there will be drastically fewer collisions. In the paper, they have upsampled with U=5. After projecting, we use max-pooling to downsample it.

Optimization

- We use the pseudo-rendered depth images and the resulting masks at novel viewpoints for optimization. Basically, in a depth image, each pixel represents the depth based on brightness. So nearer objects are black and farther objects are white.
- The mask images are those where each pixel indicate whether it belongs to the 3D model or not.
- So they apply a binary loss for Mask pixels and an L1 loss for the depth images
- There are actually 2 parts to training. First, we optimise only the structure generator and then we fine-tune the whole network using the 2d projection optimization