

Appendix A


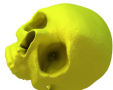

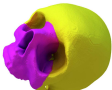

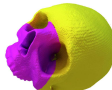








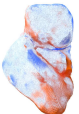









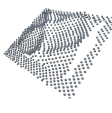
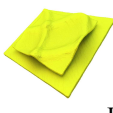
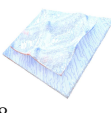
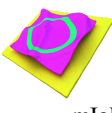

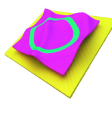
Object	Point Cloud	Reconstruction	Reconst. Error	Segmentation	Seg. Error	Reference
Skull			 IoU: 0.917		 mIoU: 0.910	
Engine			 IoU: 0.931		 mIoU: 0.729	
Liver & Gallbladder			 IoU: 0.968		 mIoU: 0.926	
Shapes			 IoU: 0.819		 mIoU: 0.791	
Sheet			 IoU: 0.948		 mIoU: 0.876	

Figure 11. Continuation of Fig. 6. Examples of reconstructions generated by conditioning on the input **Point Cloud**. The **Reconstruction** takes all non-empty classes to be the same. **Reconstruction Error** identifies **over-reconstruction** and **under-reconstruction** when compared with the reference. **Segmentation** colors each class uniquely, resulting in a **Segmentation Error** anywhere it differs from the **Reference**. IoU and mIoU values are averaged over all test data, not just the rendered examples.

Appendix B

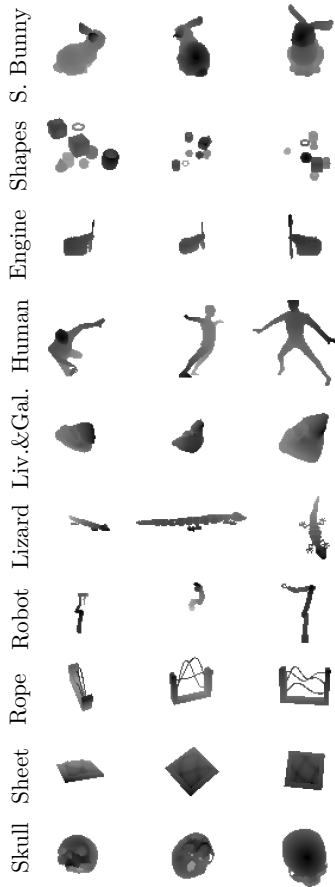


Figure 12. Examples of depth images of deformed objects and random camera perspectives that are used as input to the system.

Objects Although our method is generally applicable to 3D reconstruction tasks, we focus our experiments on objects where detailed prior information exists, but the deformation must be inferred at test time.

Each scene includes a Python script that applies a set of random deformations to the scene within hand-crafted bounds. Examples of these random deformations are shown in Figure 4.

The Lizard is copyright of Javi Rodríguez under the Creative Commons License and can be found at thingiverse.com/thing:3505006. The Stanford Bunny is copyright of the Stanford Computer Graphics Laboratory.

Some objects (*i.e.* the Robot and Lizard) feature articulated deformations, while others feature continuous deformations (*e.g.* the Human, Rope, *etc.*) or a combination of both. Several objects, such as the human, are nearly symmetrical along some axis, yet incorporate a handedness which may be difficult to detect.

Each scene contains a virtual camera that is oriented towards the objects from a random position within a cone of aperture 140 deg at a distance from the object between 100 and 200 units. A 96x96 depth image is captured from the camera’s perspective, see Fig. 12. Gaussian noise with a standard deviation of 0.1 units is added to the depth values (the shortest dimension of each scene is ~ 20 units).

Encoder Architectures In this context, an encoder is a neural network that receives a point cloud as input and distills a latent encoding of the information in it. We test our framework with 3 different encoder architectures. PointNet++ [31, 32] is a well-studied architecture for classifying and segmenting point clouds. The PointNet++ encoder contains 2 PointConv layers and a global max pool layer. The PointTransformer [44] contains 5 transformer blocks. DeepSDFs [30] autodecoder computes a latent encoding as the result of a test-time optimization process. The input point cloud is reinterpreted as the locus of points where the signed distance should be zero. This allows an optimization to find the latent code that best explains why the all points in the point cloud have a distance of zero. While it is possible to define the CE loss during training, at inference time only the L1 loss can be optimized. We found this to be enough to infer the segmentation. In all cases, the latent encoding has length 1024.

Metrics The quality of the reconstruction depends on two factors: 1) the quality of reconstruction itself, and 2) the quality of it’s segmentation. In line with previous literature [34], we evaluate these factors with IoU and mIoU [10], respectively. mIoU is defined for multi-class segmentation as the average of IoU of each individual class, ignoring free space. Multi-class IoU is computed by assigning all non-empty classes to an *occupied* class in both the reference and the prediction.

$$\text{mIoU} = \frac{1}{|C| - 1} \sum_c^{\{C\} \setminus c_{\text{empty}}} \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c + \text{FN}_c} \quad (4)$$

Both metrics are calculated over all points in the evaluation dataset.

Evaluation Data The evaluation dataset is created by discretizing the joint bounding box of all objects into a 100^3 voxel-grid. As a performance optimization, points not within the enlarged (by 50%) bounding boxes of any segments are discarded.

Hyperparameters: PontNet++ 28800 training examples, 128 test examples. Batch size of 40. Learning rate

of 0.0005. 300 epochs. 256 occupancy query points per segment, 128 inside and 128 outside. $n = k$ for SortSample. $n_{uniform}$ is chosen as 15% of total points. Variance of Gaussian noise is 0.1. Latent vector has a size of 1024. Positional encoding exponents in $\{-4, \dots, 5\}$. For more hyperparameters, see Appendix B.

Hyperparameters: Transformer Same as hyperparameters of PointNet++ method, see Sec. 5.

Hyperparameters: Autodecoder Same as hyperparameters of PointNet++ method, see Sec. 5. With exception of: Batch size of 64. Network learning rate of 0.0005. Latent vector learning rate of 0.001. 1000 epochs. 75 latent vector optimization steps at inference time. Dropout in MLP of 0.2.

Appendix C

Noise in Depth Camera We further investigate how the quality of the reconstruction is affected by increasing noise in the depth measurement. Data is generated for Robot with 3 different levels of Gaussian noise applied to the depth values: 0.1, 0.5, and 2.0 units. PointNet++ is trained on each dataset using CE+L1 loss, and all 3 models are cross-evaluated on all other noise levels.

Table 3. IoU and mIoU values from PointNet++ trained using CE+L1 loss on Robot with 3 different levels of Gaussian noise in the depth values. Models trained with one noise level (given by the row) are evaluated on another noise level (given by the column).

test →	0.1	0.5	2.0	0.1	0.5	2.0
0.1	0.882	0.581	0.346	0.864	0.541	0.289
0.5	0.852	0.852	0.763	0.829	0.830	0.734
2.0	0.808	0.816	0.815	0.781	0.789	0.789
train ↑	IoU			mIoU		

IoU and mIoU values are shown in Tab. 3. For comparison with a maximum noise level of 2 units, the shortest dimension for Robot is 5 units. Despite extremely noisy point clouds, the network is able to reconstruct Robot accurately, so long as an equivalent or higher level of noise is seen during training. This may be beneficial to real world applications, where an accurate depth measurement is not always economical or feasible. Only if the noise significantly exceeds the noise distribution in the training data is reconstruction likely to fail.

Appendix D

Point Cloud	Reconstruction	Reconst. Error	Segmentation	Seg. Error	Reference