RecVis final project : Single view 3D object reconstruction

Lambert Fatoux
MVA - ENS Paris Saclay
lambert.fatoux@eleves.enpc.fr

Loïck Chambon
MVA - ENS Paris Saclay
loick.chambon@eleves.enpc.fr

Abstract

This report presents methods and results to reconstruct a 3D mesh. It focuses on comparing methods grouped into two categories: those that use implicit representations such as an occupancy functions [7] or a signed distance functions [8] and those that use an hybrid representation by solving for example the Poisson equation [9]. We find out that these approaches allow to learn complex shapes and yield similar results on a specific class of the ShapeNet dataset.

1. Introduction

Dealing with shapes reconstruction leads to a shape modelization issue. Traditional 3D representation such as voxels, point clouds and meshes present a trade off between memory requirements and accuracy. If some neural networks try to refine their use with octrees [11] or poly-cube mapping [2], they still suffer from discretization and fall into disuse.

Subsequently, other representations, known as implicit representations, appeared. They aim to learn a function representing a surface [10] and are defined as the isocontour S of a volumetric scalar function on a isovalue ρ,

$$S_\rho = \{x \in \mathbb{R}^3 | f(x) = \rho\}$$

Deep neural networks try to approximate f by a function parametrized by $\theta \in \mathbb{R}^m$. For instance, given a point p, they can approximate a signed distance function SDF [8] by:

$$f_\theta : x \in \mathbb{R}^3 \rightarrow SDF(x) \in \mathbb{R}^3$$

or an occupancy function o [7] using a conditional observation x :

$$f_\theta : (p; x) \in \mathbb{R}^3 \times \mathcal{X} \rightarrow o(x) \in [0, 1]$$

Furthermore, implicit representations have gained popularity by their ability to be rendered directly from their volumetric representation with a ray-casting approach, or converted to triangle meshes for display before to be reconstructed using Marching Cube [5]. Moreover, although inference with the implicit representation is slower than with the explicit representation, the meshes obtained are much better.

2. Networks

2.1. DeepSDF

Published in "*DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation*" [8] by Jeong Joon Park in 2019, DeepSDF is a simple auto-decoder taking a point $x \in \mathbb{R}^3$ within a previously constructed point cloud, that uses a severe sampling strategy on the edges of the mesh, and a latent vector $z \in \mathbb{R}^{259}$ optimized during training. During inference, the learned parametrized function f_θ returns for each point x its signed distance function s. Then the surface is reconstructed by calculating the isosurface associated with a null isovalue.

The architecture is simply a composition of 8 fully-connected layers with one skip connection of the spatial information from the input to the half of the model. All internal layers are 512-dimensional and have ReLU non-linearities. The output non-linearity regressing the SDF value is *tanh*. The paper proposed to train their model by optimizing a clamped loss parametrized by $\delta = 0.1$ which allows to control the distance from the surface over which they expect to maintain a metric SDF.

$$\mathcal{L}(f(x;\theta)) = |clamp(f(x;\theta), \delta) - clamp(s, \delta)|$$

Our implementation is based and adapted from the official repository [1].

2.2. Occupancy Network

The occupancy network works similarly but aims at predicting the occupancy function for a given input. The network takes in entry the 3D points we are trying to classify as well as a conditioning input, that can be an image, 3D pointcloud or voxels. The training part is done by uniformly sampling points $p_i \in \mathbb{R}^3$ inside the bounding box (plus an small padding) of an object as well as its occupancy value $o_i = o(p_i)$. The loss used in the paper is the cross entropy classification loss:

$$\mathcal{L}(f_\theta, o_i) = -(o_i \log(f_\theta(p_i, x_i)) + (1 - o_i) \log(1 - f_\theta(p_i, x_i)))$$

The neural network is a fully connected residual network architecture made of 5 ResNet blocks, that outputs in the end the

[1] https://github.com/facebookresearch/DeepSDF.

occupancy probability. One important aspect is that the input is conditioned using conditional batch normalization that allows to have different types of encoder depending on the type of input : image, voxels or pointcloud.
For the single view reconstruction, a ResNet18 [4] encoder is used, for pointclouds the PointNet encoder is used and VoxNet [6] is the choosen encoder for the voxel representation.

2.3. Shape As Points

Published in "*Shape As Points: A Differentiable Poisson Solver*" [8] by Peng et al. in 2021, Shape As Points takes a noisy, unoriented point cloud $\{c\}$ as input and outputs a 3D shape using a differentiable Poisson solver.

The architecture is composed of three models. The first one takes as input $\{c\}$ and encodes it into a feature ϕ using a convolutional point encoder proposed in [1]. The second one is a MLP that takes as input $(c, \phi(c))$ and predict the offset Δc of the unoriented point cloud. Finally, the third model is a MLP that predicts the normals of the point cloud using information from $c + \Delta c$. The two MLP comprises 5 layers of ResNet blocks with a hidden dimension of 32 and they do not share weights.

Since their network uses a Differentiable Poisson Solver, the loss is at the core of the training. It is the mean squared error between the predicted and the ground truth indicator grid χ that describes the solid geometry.

$$\mathcal{L}_{DPSR} = ||\chi - \hat{\chi}||_2^2$$

To optimize this loss, we have to derivate three terms. The second term concerns the solution of the Poisson equation. The last term directly involves the optimisation of the neural network. Below p_o is an oriented point and p_{uno} is an unoriented point.

$$\frac{\partial \mathcal{L}_{DPSR}}{\partial p} = \frac{\partial \mathcal{L}_{DPSR}}{\partial \chi} \cdot \frac{\partial \chi}{\partial p_o} \cdot \frac{\partial p_o}{\partial p_{\text{uno}}}$$

Our implementation of the network relies on the official one available on github. [2]. And we have used, if not specified, the default settings available in the "configs/learning_based" folder of the official repository.

3. Experiments

Everything has been trained, validated and tested from scratch. Our trained networks and some reconstructed meshes are available at this link [3]. Without any specifications, we have used the hyper-parameters indicated in the papers.

3.1. Dataset

The dataset that has been used is an excerpt of the sofas class of the Shapenet dataset [3]. Our dataset has been carefully split into 3 subsets for training, validation and testing, making sure that the testing datasets were the same for each comparison that has been made. The training and the testing splits are the same as the one available on the DeepSDF repository. Since the validation dataset was not available, we used our own split, taking the same amount of data as the split of the test dataset.

In the end, around 1600 meshes were used for training, 410 for validation and 410 meshes were used to test and compare the results of our models over several metrics that will get discussed later.

3.2. Pre-processing

The three papers proposed three different ways of pre-processing the data. About the occupancy and the Poisson solver model, we did not change anything as the data were already pre-processed. Concerning the SDF model, due to compilation errors, we have not executed the code of the official repository but we have used a repository that mimics the complex sampling strategy [4].

To have a fair comparisons between our models, we have sampled the same amount of points (about 100k) for the three models even if the original paper suggests to sample 550k points (250k points on the surface of the mesh perturbed by two noise with variance 0.0025 and 0.00025, plus 50k points sampled in the unit sphere). We have kept the sampling methods suggested in the respective papers, as they are allegedly yielding better results.

3.3. Quantitative results

As the pre-processing steps were different, we have decided to normalized our reconstructed meshes on the unit sphere before evaluating them. As a result, the Chamfer and the Earth's mover distances were smaller than the one presented in [7] [9] of several order of magnitudes but get closer to the one presented in [8].

As it is presented in the paper [9], we have decided to train two configurations of the Poisson solver model, corresponding to the levels of noise: $\sigma = 0.005$ and $\sigma = 0.025$. Below are our results for the Chamfer-l2 distance and the Earth Mover's [12] distance using the definition and the subsampling computation strategy described in [8]

Model	Mode	Chamfer-L2	Earth Mover's
DeepSDF	mean	2.53	1.03
	mediane	1.93	0.97
ONET	mean	0.760	0.865
	mediane	0.682	0.843
SAP $\sigma = 0.005$	mean	**0.285**	**0.836**
	mediane	**0.192**	**0.825**
SAP $\sigma = 0.025$	mean	0.880	0.881
	mediane	0.543	0.866

Table 1: Comparisons for representing unknown sofas on the test set. Lower is better for all metrics. Chamfer distance has to be multiplied by 10^{-3}

[2] https://github.com/autonomousvision/shape_as_points

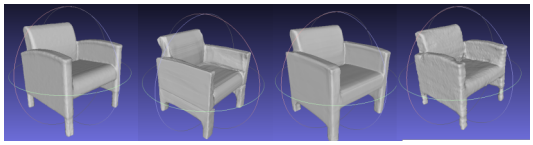
[3] https://drive.google.com/drive/folders/1qZ0kgDfGb0ejKAgrxaXHLruHJYlwFx8?usp=sharing

[4] https://github.com/marian42/mesh_to_sdf

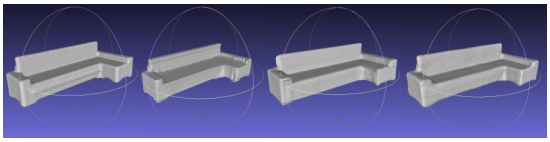
The SDF model gives us the worst results while the Poisson solver model gives us the best. As expected for this model, the less noise we have, the better the results. Concerning the DeepSDF results, one explanation could be the calculation of the initial signed distance function by the Python library. After being preprocessed, some meshes do not allow a good calculation of the signed distance function. Theoretically, the calculation of the signed distance from a point to a surface using the scalar products with its nearest neighbours is correct. But, in practice, it is not suitable for noisy data. This aspect is well known from the meshtosdf library.

3.4. Qualitative results

Generally speaking, our models were quite efficient to reconstruct simple meshes with shapes enough represented in the training dataset. The meshes are displayed using Meshlab.



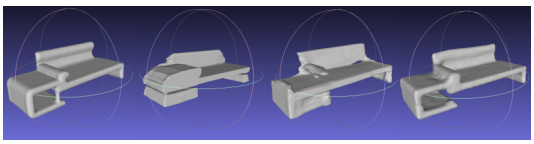
(a) Simple shape that can be seen as an outlier in the sofas dataset



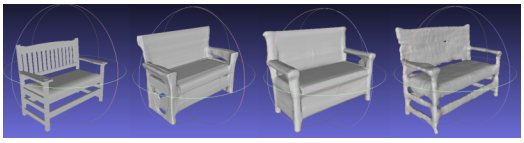
(b) Simple common shape of the sofas dataset

Figure 1: Well reconstructed mesh.

However, they present sometimes some difficulties to reconstruct thin elements, unusual elements such as pillows, holes in the meshes and they tend to bulge the edges of the object. The difficulties encountered stem from both the scarcity of items and the uncommon shapes in the dataset. Sofas tend to be filled, without holes, and pillows do not always appear in the meshes. These comments apply particularly to DeepSDF but much less to SAP.



(a) Fails to reconstruct complex shapes with holes



(b) Fails to reconstruct thin elements.

Figure 2: Failure cases.

4. Conclusion

The papers we have been working on all have the same objective: to find an efficient way of representing 3D topologies, that is less computationally expensive than the traditional voxels or pointclouds representations. As desired in the project proposal, we were able to re-train the model on a single class of the Shapenet dataset for a signed distance model and an occupancy model. In addition, we tested a more recent method based on solving the Poisson equation which provided the best results. The metrics that we have used to compare the DeepSDF, the ONet and the SAP models tend to show that all of them are able to grasp geometries, with an advantage of the SAP model with low noise on the Chamfer and Earth Mover's distances.

One pitfall that is however important to mention is that all of the training and testing processes were done on synthetic data and the articles that we have worked on mostly omitted the real life cases applications.

With more time and resources, we would have explored IGR, a method based on solving an eikonal equation. Although we tested this method on the DFaust dataset, we did not manage to obtain convincing meshes as they showed many artefacts.

References

[1] Peng et al. *Convolutional occupancy networks*. 2020.

[2] Fleuret Baque Remelli and Fua. *Geodesic Convolutional Shape Optimization*. 2018.

[3] Angel X. Chang et al. *ShapeNet: An Information-Rich 3D Model Repository*. 2015. arXiv: 1512.03012 [cs.GR].

[4] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2016. DOI: 10.1109/CVPR.2016.90.

[5] W. E. Lorensen and H. E. Cline. *Marching cubes: A high resolution 3d surface construction algorithm.* 1987.

[6] Daniel Maturana and Sebastian Scherer. *VoxNet: A 3D Convolutional Neural Network for real-time object recognition*. 2015. DOI: 10.1109/IROS.2015.7353481.

[7] Lars Mescheder et al. *Occupancy Networks: Learning 3D Reconstruction in Function Space*. 2019. arXiv: 1812.03828 [cs.CV].

[8] Jeong Joon Park et al. *DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation*. 2019. arXiv: 1901.05103 [cs.CV].

[9] Songyou Peng et al. *Shape As Points: A Differentiable Poisson Solver*. 2021.

[10] Christian Sigg. *Representation and Rendering of Implicit Surfaces*. 2006.

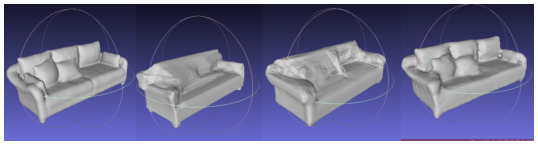
[11] M. Tatarchenko, A. Dosovitskiy, and T. Brox. *Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs*. 2017. URL: http://lmb.informatik.uni-freiburg.de/Publications/2017/TDB17b.

[12] C. Tomasi Y. Rubner and L. J. Guibas. *A metric for distributions with applications to image databases.* 1998.

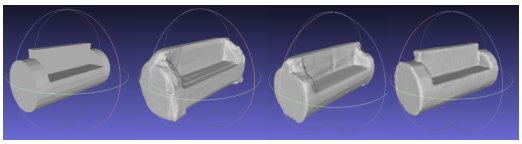
5. Annexe

5.1. Other difficult meshes

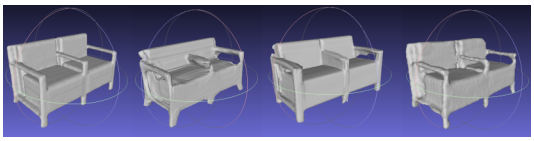
More at available at this link [5]



(a) Reconstruction of the pillows



(b) Very unusual shape



(c) Complex shape with holes and thin elements.

Figure 3: More samples of interesting shapes. Respectively: groundtruth, DeepSDF, ONET and SAP low noise.

5.2. Training

5.2.1 DeepSDF

Interestingly, the training loss falls quickly but the networks is still optimizing the magnitude of the latent vector. It seems that it is as important as the loss to indicate the convergence of the algorithm. The two peaks present on the graphs are the consequence of stopping and then resuming training. The training was the longest one and took 10 hours on a A100 GPU.

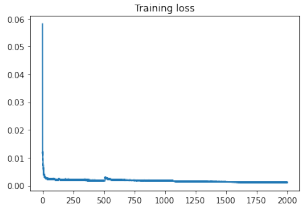


Figure 4: Train loss during the training process for the signed distance network

[5] https://drive.google.com/drive/folders/1qZ0kgDfGb0ejKAgr-xaXHLruHJYlwFx8?usp=sharing

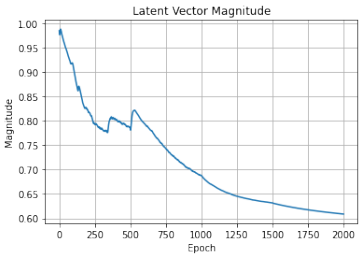


Figure 5: Magnitude of the latent vector code

5.2.2 Shape As Points

Below are our training loss for $\sigma = 0.005$. The network for $\sigma = 0.025$ present a similar curve. The networks takes 1 hour on a A100 GPU to run over 3k iterations.

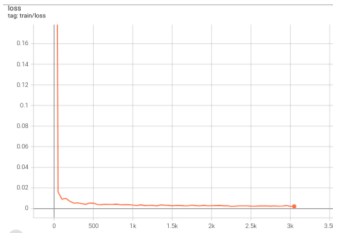


Figure 6: $\sigma = 0.005$. After 6k iterations, the validation loss display on another tensorboard is about 0.0020 whereas the final training loss is 0.0011

For $\sigma = 0.025$, after 3k iterations, the validation loss display on another tensorboard is about 0.0044 whereas the final training loss is 0.0024. In both case, we have a little overfitting, of a factor of 2.

5.2.3 Occupancy network

The occupancy network was trained on a A100 GPU for about 4 hours on the sofas extracted dataset, using a pointcloud for the conditioning additional input fed into the PointNet encoder.

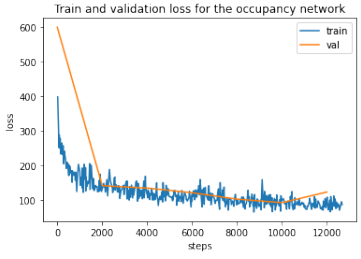


Figure 7: Train and validation loss during the training process for the occupancy network