GenerativeAi 1st Assigment

Emmanouil-Thomas Chatzakis

May 2025

1 Abstract

This project implements and compares two approaches for 3D surface reconstruction from sparse point clouds: a geometric tangent-plane method and neural implicit function approximation. Both methods leverage surface normals to estimate signed distance functions (SDFs), processed through marching cubes for mesh extraction. The geometric approach computes SDFs via nearest neighbor searches and tangent plane projections, while the neural method employs an 8-layer network with skip connections trained using clamped L1 loss. Experimental results demonstrate reconstruction quality improvements from 500 to 1000 sample points, with both methods achieving accurate sphere reconstructions.

The geometric approach computes SDF values by finding the nearest surface point and calculating the distance to its tangent plane. The neural method employs an 8-layer fully connected network to learn a continuous SDF representation. Training utilizes a clamped L1 loss on points sampled along surface normals with Gaussian displacement.

2 Geometric Reconstruction

2.1 Mathematical Formulation

The signed distance function (SDF) at any grid point $p = (x, y, z)$ is computed using:

$$f(p) = \mathbf{n}_j \cdot (p - p_j), \quad j = \text{argmin}_i \|p - p_i\| \tag{1}$$

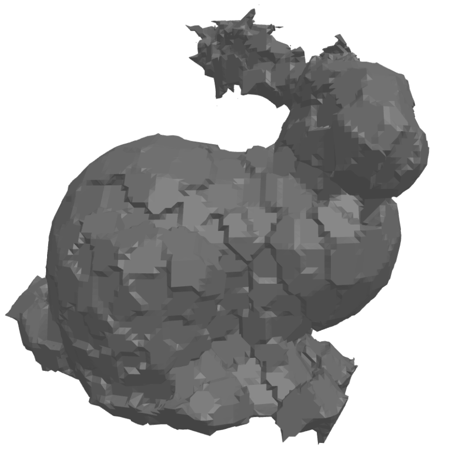
where p_j is the nearest surface point and $\mathbf{n}_j$ its associated normal.

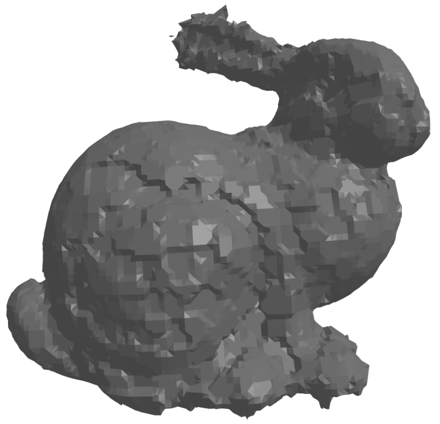
2.2 Implementation

To efficiently compute signed distances across the entire 3D grid, we first transform the grid coordinates X, Y, and Z (each of shape $N \times N \times N$) into a 2D array Q of dimension $(N^3, 3)$, where each row contains the (x, y, z) coordinates of a grid point.
A KD-tree spatial data structure is then constructed from the input point cloud.

This data structure allows rapid identification of the closest surface point p_j for each grid point p.
Next, we impliment the $f(p)$ function for each grid point p by computing the the vector to its nearest surface point p_j and project it onto $p_j's$ normal n_j
Once the signed distances for all grid points have been computed, they are initially stored in a one-dimensional array of length N^3. To prepare for the final step, we convert the flat array of signed distance values back into a 3D grid with shape (N, N, N). This way, each value lines up with its correct position in the grid, matching the original (x, y, z) coordinates. This format is needed for the marching cubes algorithm to find and extract the surface.

2.3 Results


(a) Naive Reconstruction of bunny with 500 points


(b) Naive Reconstruction of bunny with 1000 points

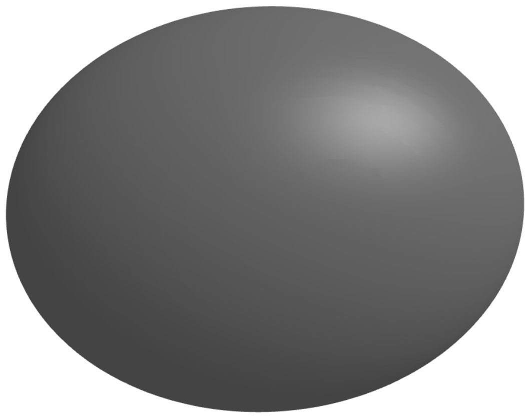

(c) Naive sphere reconstruction

Figure 1: Naive reconstructions

3 Neural Network Reconstruction

3.1 Neural Network architecture

The neural network used for signed distance function (SDF) approximation is a fully-connected network. The goal is to learn a mapping from a 3D coordinate $p = (x, y, z)$ to its corresponding SDF value $f_\theta(p)$, where θ denotes the network parameters.

Network Structure:

- **Input:** A 3D point coordinate (x, y, z).
- **Hidden Layers:** The network consists of 8 fully-connected (FC) layers. Each of the first seven layers has 512 hidden units.
- **Skip Connection:** After the fourth FC layer, the original input (x, y, z) is combined with the original input to the hidden representation (which is of size 509), restoring the vector size to 512 before passing to the next layer.
- **Normalization and Activation:** Each of the first seven FC layers is followed by weight normalization, a leaky ReLU activation (with a learnable slope shared across all channels), and dropout with a rate of 0.1.
- **Output Layer:** The final (eighth) FC layer reduces the dimensionality from 512 to 1. A tanh activation function is applied to bound the SDF output.

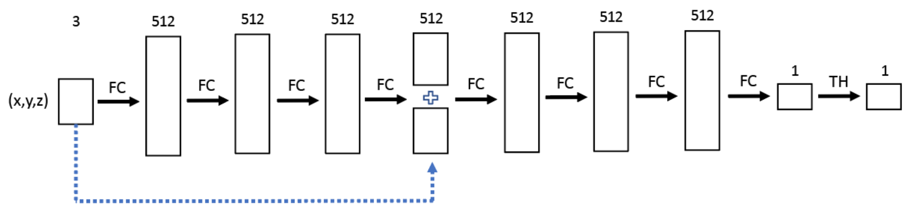


Figure 2: Network architecture

3.2 Neural Network training

The training process for the model involves several key steps to ensure accurate learning and stable optimization.

3.2.1 Value Clamping

Both the predicted and ground truth SDF values are constrained to the interval $[-\sigma, \sigma]$. This ensures that all SDF values considered during validation remain within specified bounds.

3.2.2 L1 Loss Computation

The model uses L1 loss, to measure the difference between predicted and ground truth Signed Distance Function (SDF) values. The loss is computed as:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} |y_{\text{predicted}} - y_{\text{real}}\ |$$

This formulation penalizes the average absolute difference between predictions and targets, promoting robust and stable training.

3.2.3 Gradient Propagation

To update the model parameters, an explicit call to `loss.backward()` is made. This operation computes the gradients of the loss with respect to the model parameters, enabling proper backpropagation through the decoder network. This step is essential for the optimizer to adjust the network weights and minimize the loss function effectively.

3.2.4 Loss Normalization

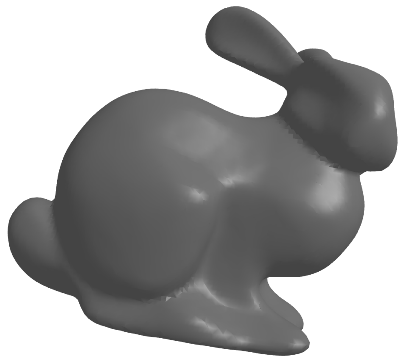
During training, the loss is accumulated as `loss_sum += loss.item() * batch_size`. This normalization ensures that the total loss accounts for variable batch sizes, allowing for correct averaging and comparison across different training steps. Proper loss normalization is crucial when handling datasets where the number of samples per batch may fluctuate.

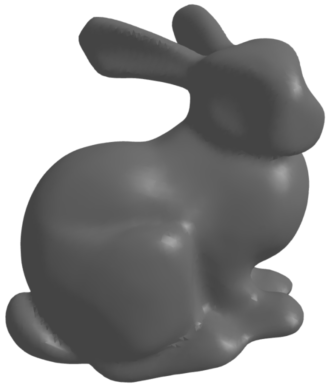
3.3 Neural Network validation

1. **Value Clamping:** Both the predicted and ground truth SDF values are constrained to the interval $[-\sigma, \sigma]$. This ensures that all SDF values considered during validation remain within specified bounds.

2. **Evaluation Mode:** The validation step is performed inside a `torch.no_grad()` context, which disables gradient computation.

3.4 Results


(a) Neural Network Reconstruction of bunny with 500 points


(b) Neural Network Reconstruction of bunny with 1000 points

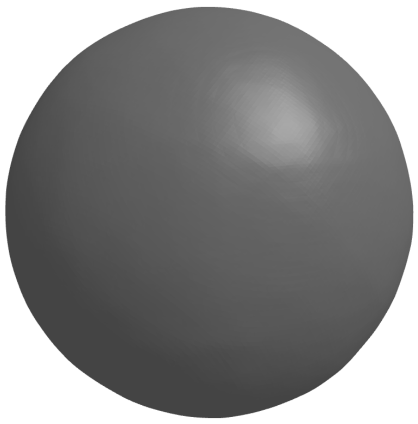

(c) Neural Network sphere reconstruction

Figure 3: Neural Network reconstructions