SweepNet Supplementary Material

Mingrui Zhao[1], Yizhi Wang[1], Fenggen Yu[1], Changqing Zou[2], and Ali Mahdavi-Amiri[1]

[1] Simon Fraser University
[2] Zhejiang University

1 Neural Sweeper Training

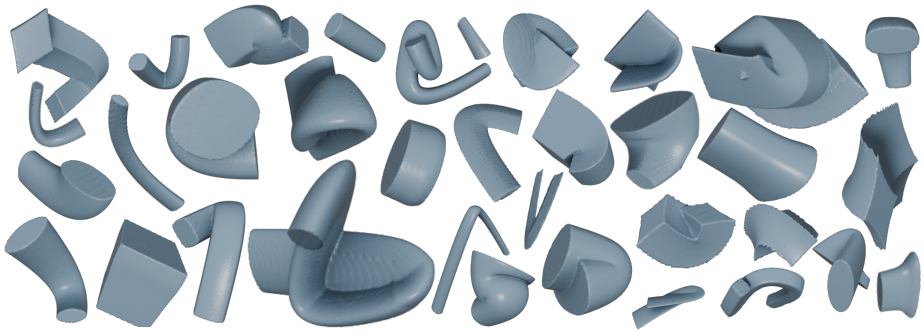

Fig. 1: Sweep surfaces data samples for neural sweeper training.

1.1 Data Preparation

To fully train our neural sweeper, we create a dataset of sweep surfaces with a variety of parameters using the technique introduced by Sèllan et al. [3]. Fig. 1 showcases some randomly selected samples in the dataset. Specifically, the parameters of a sweep surface data sample are set as follows:

Sweeping Axis. As mentioned in our main paper, the sweeping axis is a B-spline curve. Control points for the sweeping axis are randomly generated within the range $[-0.5, 0.5]^n$, where n is the number of control points. Since the input shapes to SweepNet (our main network) are normalized within the unit cube, this range selection ensures that the sweeping axis remains within the confine, preserving the integrity of the sweep surfaces.

Superellipse Profile. Parameters for the superellipse profile, including the major-minor axis and degree, are selected randomly within the ranges $[0.01, 0.5]^2$ and $[0.3, 5]$, respectively. This approach prevents the generation of overly small profiles which could potentially hinder the learning process. The degree lower bound is set to be 0.3 to avoid extreme star-shape profile with diminishing corners.

Scaling Function. We adopted quadratic function $ax^2 + bx + c$ with $c = 1$ for the profile scaling. This setup allows for constant scaling when a and b are set to zero. The parameters a and b are confined within the range $[-0.5, 0.5]$, ensuring the scaling velocity remains within a reasonable limit.

1.2 Point Cloud Sampling

After obtaining the mesh of a sweep surface, we sample a point cloud from it as a training input to our neural sweeper. For each sweep surface, we sample (a) 15 profile frames, with each consisting of 50 contour points and (b) 124 points from the sweeping axis. There are altogether $15 \times 50 + 124$ points as a point cloud fed to the neural sweeper. Traditionally, a point cloud contains only the points from the object surface. Here we include the points from the sweeping axis as auxiliary information in the network input. The sampling details are as follows:

- The axis points are uniformly sampled from the B-spline curve's parameter space.
- The contour points are uniformly sampled using the superellipse formulation with θ ranging from 0 to 2π.
- The profile frames position are uniformly sampled from the spline curve in parameter space. For each profile frame position, a 2D profile is scaled with the scaling function then transformed to the corresponding position with transformation matrix calculated by the parallel-transportation frame.

1.3 Training Strategy

The training protocol follows the implementation of POCO [1], using an Adam optimizer with a learning rate of $1e-3$. After completing training, the neural sweeper is frozen and cascaded to the swept volume head in SweepNet.

2 Additional Results

More visual results and comparisons are provided in Fig. 2. The results demonstrate the power of sweep surfaces in representing curved surfaces, and our proposed scaling function further enhances their expressive capabilities.

Fig. 3 provides additional examples to show how the number of control points affects the abstraction results. A sweeping axis with more control points exhibits more curvy features, while our method remains stable in both settings.

Despite curvy-feature objects, we provide additional qualitative results of SweepNet on Thingi10K [4] and ShapeNet [2] datasets. These datasets contain many CAD-like shapes where the sweep elements are not commonly observed. Our method can provide comparable results. We would like to emphasize that no single primitive is perfect for all 3D shapes. Each primitive has its strengths and weaknesses, and the combination of various methods can often provide a more comprehensive solution. SweepNet excels in handling objects with curvy and

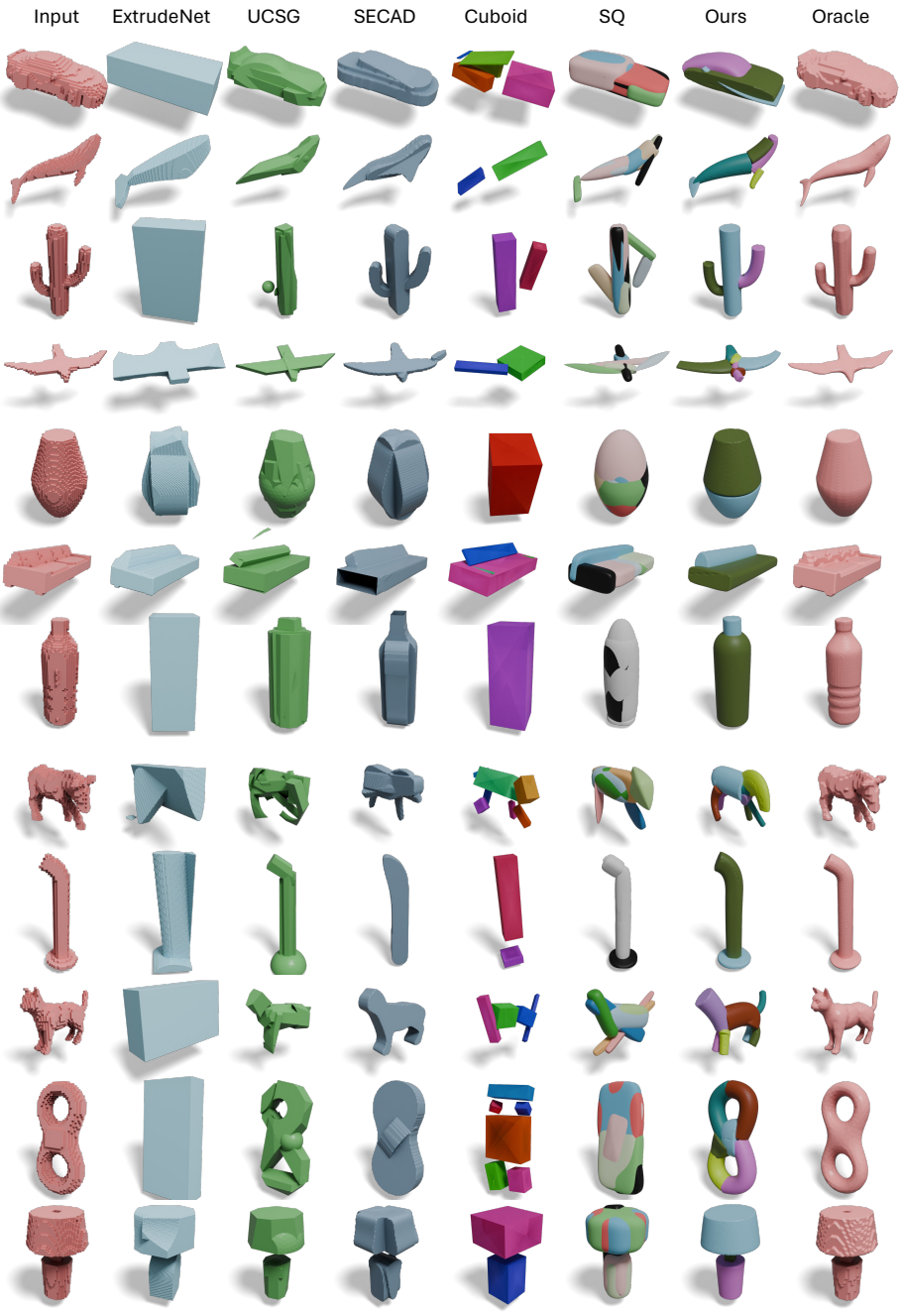

Fig. 2: Additional qualitative results on GC-objects and quadrupeds datasets. Our method better captures curves.

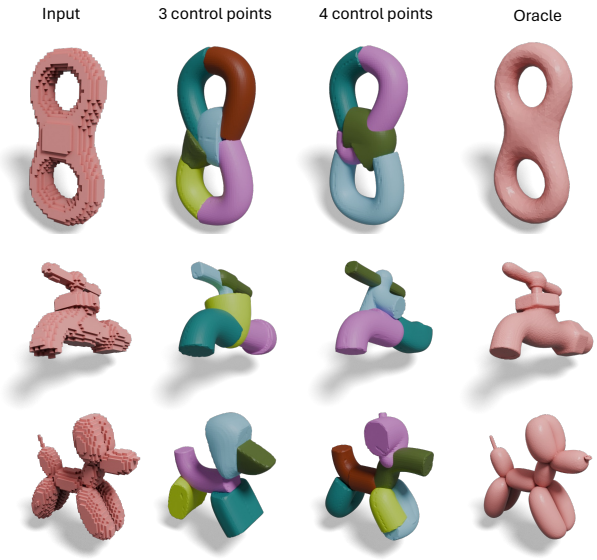

Fig. 3: Shape abstraction outcomes from SweepNet utilizing sweeping axes defined by 3 and 4 control points, respectively. Primitives characterized by 3 control points are more concise, while those with 4 control points showcase more turning curves and employ fewer primitives in abstraction.

tubular features but may not be the best fit for CAD-like shapes. Combining SweepNet with other parametric primitives can harness the strengths of each method to achieve better overall performance.

Lastly, we provide qualitative results from SweepNet in representing alphabetical letters and numbers in Figs. 9 and 10.

3 Loss function

As detailed in the main paper, we propose four distinct loss functions: $\mathcal{L}_{recon}$, $\mathcal{L}_{ol}$, $\mathcal{L}_{pars}$, and $\mathcal{L}_{axis}$. This section elucidates the rationale behind the design of each loss function and delineates the methodology for tuning their respective weights.

3.1 Reconstruction Loss Formulation

We used the Boltzmann operator to formulate the reconstruction loss to enable a smoother gradient flow. This operator calculates the occupancy value at a test point by taking the weighted sum of the occupancy values from all present primitives, ensuring that all primitive parameters can be updated during backpropagation. The weight is determined by a biased softmax function controlled by the parameter α. Alternatively, the argmax operator could be used to update only

the primitive contributing the highest occupancy, but this method empirically slows convergence. The Boltzmann operator provides a smooth approximation of the maximum function, and its sharpness toward the true maximum can be adjusted by tuning the parameter α.

$$\mathcal{L}_{recon} = \mathbb{E}_{t\sim T}\left[\left\| O_{GT}(t) - \frac{\sum_{i=1}^{q} O_i(t) e^{\alpha O_i(t)}}{\sum_{i=1}^{q} e^{\alpha O_i(t)}} \right\|_2^2\right]. \tag{1}$$

3.2 Loss Tunning Strategy

The primary loss function, $\mathcal{L}_{recon}$, measures the fidelity between the abstracted shape representations and their corresponding GT counterparts. Nevertheless, relying solely on $\mathcal{L}_{recon}$ may result in suboptimal abstraction. This is characterized by a tendency of the model to produce aggregated and cumbersome representations that merely approximate the target shape, without achieving meaningful abstraction. To address this, we introduced the overlap loss, $\mathcal{L}_{ol}$, and the parsimony loss, $\mathcal{L}_{pars}$, to encourage more parsimonious reconstructions by penalizing overlapping primitives and the excessive use of primitives. Empirical observations suggest that the overlap loss also facilitates faster convergence by introducing a repulsive force among primitives.

Furthermore, the axis loss, $\mathcal{L}_{axis}$, guides the selection of sweeping axes towards the medial axis of the target object, which can be directly extracted from the input voxel data. Incorporating the medial axis as additional supervision significantly enhances our model's performance, rapidly narrowing the solution space and establishing a robust prior for the sweeping axis.

When determining the weights assigned to each loss function, we consider the following principles: The reconstruction loss has the dominant weight to ensure a faithful shape representation, followed by the overlap loss to regularize cleanliness. The parsimony loss introduces a trade-off between parsimony and fidelity, so it is set to a comparatively small value to preserve fidelity. The axis loss is mandatory but only serves as a prior. It is set with a decaying weight, with higher importance at the beginning of the training process. The impact of each loss function is shown in Fig. 4.

4 Primitive Edits

We showcase additional examples of primitive editing from Figs. 5 to 7. The abstracted shapes demonstrate versatile editing capabilities of our method by altering the profile, axis, and scaling functions of the primitives. These examples highlight the advantages of the proposed sweep surface parametrization and its flexibility.

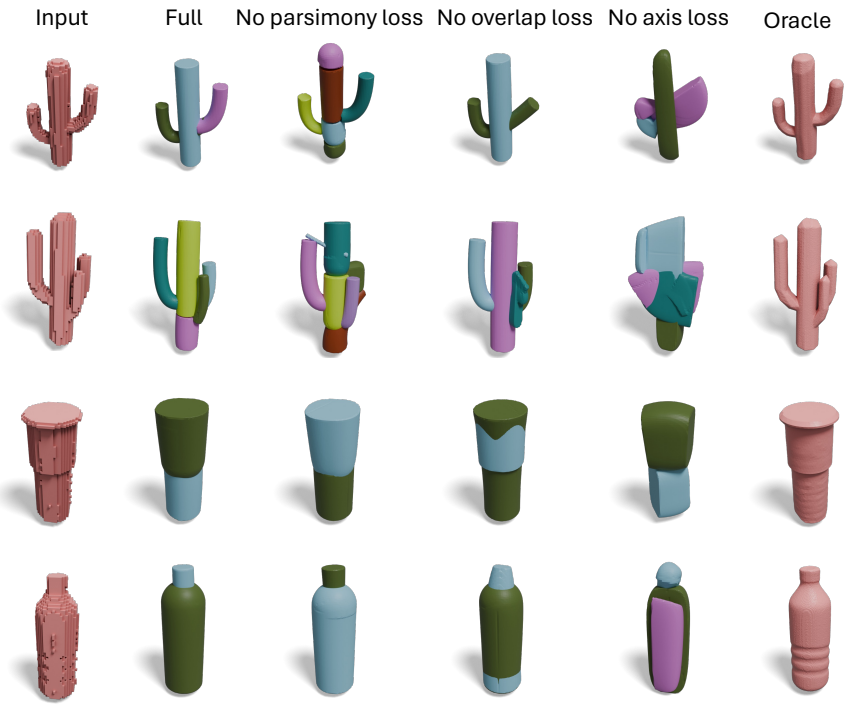

Fig. 4: Ablation study on loss functions. Parsimony loss encourages minimal use of primitives, overlap loss reduces overlaps, and axis loss provides superior guidance on sweeping axes

Fig. 5: A light bulb model is transformed into a mushroom by manipulating sweeping axis length, adjusting the scaling function, and enlarging the profile size. Each shape is represented by three primitives, with three control points each, employing a quadratic scaling function. In total, this representation requires 42 floating-point numbers.

Fig. 6: An ECCV 3D calligraphy undergoes variations in its brush style by adjusting the degree of the superellipse profile. This adjustment creates different profiles, including a star shape profile (top left), circular profile (top right), square profile (bottom left), and rhombus profile (bottom right), showcasing various font effects. Each shape is represented by five primitives, each with four control points and a constant scaling function, totaling 85 floating-point numbers. Only **one** float from each primitive needs to be updated to achieve this edit.

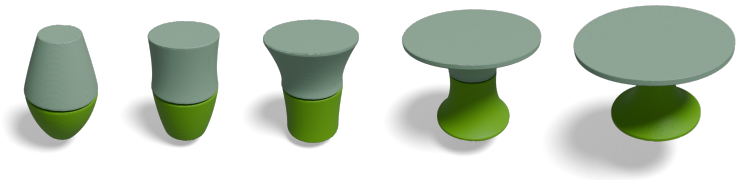

Fig. 7: A vase model transforms to a table by adjusting the sweeping axis length, and scaling function coefficients. Each shape is represented by two primitives, each with three control points and a quadratic scaling function, requiring a total of 28 floating-point numbers.

Fig. 8: Additional qualitative results on Thingi10K [4] and ShaepNet datasets [2]. Our method provides reasonable abstractions on objects lack of sweep elements.

Fig. 9: Representing letters with sweep surfaces produced by SweepNet.

Fig. 10: Representing numbers with sweep surfaces produced by SweepNet.

References

1. Boulch, A., Marlet, R.: Poco: Point convolution for surface reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6302–6314 (2022)
2. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
3. Sellán, S., Aigerman, N., Jacobson, A.: Swept volumes via spacetime numerical continuation. ACM Transactions on Graphics (TOG) **40**(4), 1–11 (2021)
4. Zhou, Q., Jacobson, A.: Thingi10k: A dataset of 10,000 3d-printing models. arXiv preprint arXiv:1605.04797 (2016)