NeuraLeaf: Neural Parametric Leaf Models with Shape and Deformation Disentanglement

Supplementary Material

In this supplementary material, we first provide details for our DeformLeaf dataset preparation and registration methods (Sec. A). We then provide more implementation details for training and evaluation, including the network architecture and implementation of the baseline methods (Sec. B), as well as details for model fitting (Sec. C).

Additional experimental settings and more qualitative results are shown in Sec. D and the supplementary video.

A. DeformLeaf Dataset Preparation

In this section, we detail the dataset preparation process, corresponding to Sec. 4 of the main paper. Our dataset, DeformLeaf, consists of approximately 300 base-deformation leaf pairs, where each pair contains both a 3D deformed shape and a 2D base shape with dense correspondences between them. First, we describe the extraction of the base shape from image masks in Sec. A.1. We also detail the rigid (Sec. A.2) and non-rigid (Sec. A.3) registration processes. Although our method, NeuraLeaf, itself does not rely on the dense correspondences between the base and deformed shape, we consider including the correspondence information in the DeformLeaf dataset facilitates baseline comparisons and supports future research.

A.1. Base Shape Extraction from Image Masks

Base shapes used in NeuraLeaf are extracted from 2D leaf image datasets. Given the mask region of leaves by an off-the-shelf segmentation method [6], we generate an SDF representation of base shapes for better compatibility with downstream tasks.

Specifically, we first calculate a 2D unsigned distance field (UDF) on the leaf's plane (*i.e.*, $z = 0$ plane). Given an image mask, the UDF is first computed using the Jump Flooding algorithm (JFA) [11] on a 2D plane, which computes each pixel's distance to the nearest object boundary. Then, the UDFs are post-processed to ensure the correct SDF sign and normalized against the maximum distance. As we described in the main paper, base shapes can be obtained in various representations by converting the 2D SDFs to other representations, such as meshes.

A.2. Rigid Registration

We first perform rigid registration for rough alignment between the base $\mathcal{S}_b$ and deformed $\mathcal{S}_d$ shapes. We take a similar but slightly different approach from the usual PCA-based alignment. The key to leaf pose estimation is to confirm the primary and secondary veins of the surface. Firstly,

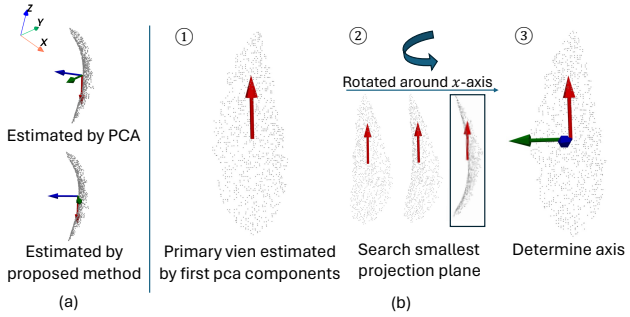

Figure S1. Rigid registration of deformed leaf shape $\mathcal{S}_d$.

we compute the largest principal component of the vertices of the deformed shape to determine the primary vein (x-axis). Although conventional PCA-based alignment methods [7] use the second principal component as the secondary vein (y-axis), as leaf deformation increases, the shape variance around the secondary vein becomes large, as shown in Fig. S1(a). Instead, we seek the secondary vein (y-axis) as the direction perpendicular to the x-axis, where the projected area of the 3D shape is minimal, as in Fig. S1(b).

A.3. Non-rigid Registration

As mentioned in Sec. 4 in the main paper, we perform a non-rigid registration pipeline using the As-rigid-as-possible (ARAP) registration followed by the Coherent Point Drift (CPD) registration. Since the leaves have flexible deformation, it is often challenging to yield a reasonable registration by a single method. We, therefore, combine two registration methods to gradually relax the degrees of freedom in the deformation, namely, rigid alignment, ARAP assuming a locally rigid shape, and CPD as non-rigid registration.

As-rigid-as-possible (ARAP) registration In human and animal models, the deformations rely on their skeletal structures with hinged joints; *i.e.*, the deformation can be characterized as a rigid motion connected at joints. ARAP allows the model to bend or rotate around these joints while keeping the local shapes. Since leaves lack the hinged joints of the skeleton, we artificially designate the joint points (hereafter called keypoints) by uniformly sampling along both the base and the deformed leaves' contour. By sampling the same number of points, we yield the correspondences between keypoints on the base shape and the scanned (*i.e.*, deformed) shapes.

Given the set of keypoints, $\mathcal{V} = \{\mathbf{v}_i \in \mathbb{R}^3\}_i$ and

$\mathcal{V}' = \{\mathbf{v}'_i \in \mathbb{R}^3\}_i$, defined on the base shape and deformed (*i.e.*, scanned) shape respectively, ARAP optimizes the vertex-specific offsets $\{\delta_i \in \mathbb{R}^3\}$ and rotation $\{\mathbf{R}_i \in \mathbb{R}^{3\times3}\}$. Specifically, we minimize the following energy function:

$$\sum_i \sum_{\mathbf{u}\in\mathcal{N}_{\mathbf{v}_i}} (d(\hat{\mathbf{v}}_i - \mathbf{v}'_i) + \|(\hat{\mathbf{v}}_i - \hat{\mathbf{u}}) - \mathbf{R}_i(\mathbf{v}_i - \mathbf{u})\|_2^2), \quad \text{(S1)}$$

where $\hat{\mathbf{v}}_i = \mathbf{v}_i + \delta_i$, $d(\hat{\mathbf{v}}_i - \mathbf{v}'_i)$ is the mean point-to-point distance from $\hat{\mathbf{v}}_i$ to its nearest neighbors in $\mathcal{V}'$, and $\mathcal{N}_{\mathbf{v}_i}$ denotes the nearest neighbouring vertices to $\mathbf{v}_i$.

Coherent point drift (CPD) registration We use a state-of-the-art implementation of CPD [4], which extends the original Euclidean CPD to non-Euclidean domains using the Laplace-Beltrami operator. This method addresses the challenge of a large search space using the nearest-neighbor descriptor matching to find confident correspondences. The probabilistic model of CPD is refined to manage outlier-contaminated initial correspondence sets effectively, employing an Expectation-Maximization (EM) approach for improved robustness and efficiency. See more details in [4].

B. Implementation Details

We train the NeuraLeaf model on a single RTX A6000 GPU using our DeformLeaf dataset. The dimensions of shape latent N_s, texture latent N_t, and deformation latent N_d are set to 256. For shape decoder f_{θ_s}, we apply the same architecture introduced in the Implicit Differentiable Renderer (IDR) [14] and train our models for $3,000$ epochs. For the skinning decoder and transformation decoder, we use the network introduced in [13]. We use a learning rate of 10^{-3} for the networks and the latent codes. We implement a learning rate decay every $1,000$ epochs. Similar to [8], latent inversion encoders $(f_{\Omega_s}, f_{\Omega_d})$ receive back-projected observation in the form of a partial SDF grid and output the latent code estimation through a series of 3D convolution layers and a final fully-connected layer. We hereafter detail the implementations further.

B.1. Architecture of Decoders

In this section, we provide an overview of the different networks used in our framework, each serving a specific role in capturing leaf shape, deformation, and texture. Specifically, our framework includes the following decoders: a **Shape decoder** f_{θ_s} for base shape representation, a **Texture generator** f_{θ_t} for modeling the leaf texture, a **Skinning Weights decoder** f_{θ_w} for predicting the influence weights of control points, and a **Transformation decoder** f_{θ_d} for calculating the transformations at each control point to achieve the desired deformation.

Shape decoder Shape decoder f_{θ_s} is an MLP that consists of 8 layers, with a dimension of hidden layers of 512 and a single skip connection from the input to the middle layer, which is similar to an existing NPM [3]. The initialization strategy of parameters of f_{θ_s} is the same as [1] so that f_{θ_s} produces an approximate SDF of a unit sphere.

Texture generator Similar to [16], the architecture of texture generator f_{θ_t} contains three convolution layers, multiple residual blocks, and two fractionally-strided convolution layers with stride $\frac{1}{2}$. We apply 9 blocks to 256×256 images and also use instance normalization. For the discriminator network, we apply a 70×70 PatchGAN to discriminate whether the overlapping image patches are real or fake.

Skinning weights decoder Skinning weights decoder f_{θ_w} uses the architecture introduced in [12], which is an MLP with intermediate layer shape $(256, 256, 256, 1000)$ with a skip connection from the input feature to the 2nd layer, and non-linear activations using LeakyReLU except the last layer uses softmax layer to obtain normalized outputs. As input, we take the Cartesian coordinates of a base point sampled from base shape $\mathcal{S}_b$, which is encoded into a high-dimensional feature using positional encoding [10], incorporating up to 8th-order Fourier features.

Transformation decoder The transformation decoder f_{θ_d} follows a similar network architecture as the skinning weights decoder with an intermediate layer shape (256,256,256,7), where the first four dimensions represent a quaternion for rotation and the remaining three dimensions represent translation. The input to the network consists of points sampled from the control points $\mathcal{C}$ and also followed by positional encoding.

B.2. Training Details

We use Adam optimizer for f_{θ_s}, f_{θ_w} and f_{θ_d} and also apply a learning rate decay factor of 0.5 every 500 epochs. For training of base shape space, the SDF truncation is set to $\delta_\tau = 0.01$ and spherical covariance defined by σ_s is 10. For deformation latent codes, the spherical covariance defined by σ_d is also 10. Both shape latent codes $\{\mathbf{z}_s^i\}_{i=1}^S$ and deformation latent codes $\{\mathbf{z}_d^j\}_{j=1}^D$ are initialized randomly from $\mathcal{N}(0, 0.001)$.

B.3. Implementation of Baseline Methods

PCA-based parametric model [2] We evaluate the parametric leaf model proposed by Barley *et al.* [2]. This PCA-based approach needs dense correspondence among all shapes within the dataset. To accommodate this requirement, we partition our base shape dataset into several

groups based on shape characteristics. Each group's shapes are registered to a template shape, then the parametric space is obtained based on BFM [9]. The template shapes are selected from DeformLeaf rather than using low-polygon leaf meshes, ensuring that the PCA baseline and other methods are compared at the same mesh resolution. For deformation modeling, we apply the same non-rigid fitting pipeline introduced in [2] and optimize the shape parameter simultaneously during inference.

Neural parametric model [8] We compare our method with a straightforward extension of the existing NPM [8], trained using our DeformLeaf dataset. Deformation modeling in this baseline is based on the neural displacement field. Thus, we use the dense correspondences in DeformLeaf for training. The training pipeline follows the paradigm in [8].

C. Model Fitting to 3D Obeservations

C.1. Details for Latent Code Inversion

As discussed in the main paper, we use two 3D convolutional networks f_{Ω_s} and f_{Ω_d} to map the back-projected SDF grid $\mathbf{G} \in \mathbb{R}^{128\times128\times128}$ to the initial latent code $\mathbf{z}_s$ and $\mathbf{z}_d$ at the inference stage. The convolutions are followed by a downsampling that creates growing receptive fields and channels with shrinking resolution, similar to common practices in 2D convolution. By applying this process recursively n times to input grid $\mathbf{G}$, it creates multi-scale deep feature grids $\{\mathbf{G}_1, ..., \mathbf{G}_n\}$, each with a decreasing resolution $\mathbf{G}_k \in \mathbb{R}^{K\times K\times K}$, where $K = \frac{128}{2^k-1}$. Feature grids $\mathbf{G}_k$ at early stages capture high frequencies, indicating fine shape details, whereas feature grids at later stages capture the global structure of the data.

C.2. Details for Direct Optimization

After initial latent code $\mathbf{z}_s$ and $\mathbf{z}_d$ predicted, we optimize for $\mathbf{z}_s$ and $\mathbf{z}_d$ by minimizing Eqs. (13) and (14) in the main paper. We use Adam optimizer and learning rate of 1×10^{-3} for both $\mathbf{z}_s$ and $\mathbf{z}_d$. We optimize for a total of 300 iterations and perform learning rate decay by a factor of 0.5 every 50 iterations.

C.3. Texture Latent Estimation

In the limitation section of the main paper, we mention that our method has shortcomings in fitting the texture, as the texture latent code cannot be fitted to the input through fine-tuning. Therefore, we take the mean of the pre-trained texture space as the initial value, which is then updated.

C.4. Details for Multiple Leaf Reconstruction

To begin the reconstruction process, each leaf instance is initially processed by the shape encoder f_{Ω_s} to estimate the shape latent code $\mathbf{z}_s$. However, due to occlusion or overlapping regions among leaves, some of the estimated shape codes might not accurately represent the true leaf shape. To identify a representative anchor latent code in the presence of potential outliers, we first compute the similarity between latent codes using Euclidean distance and perform clustering. Specifically, we employ K-means clustering to divide the latent codes into $K = 3$ clusters and select the cluster with the largest number of latent codes as the main cluster, assuming that this cluster represents the normal shape. Once the main cluster is identified, we choose an anchor latent code $\mathbf{z}_s^{anc}$, by selecting the code closest to the cluster center. $\mathbf{z}_s^{anc}$ serves as the reference during optimization. Formally, we introduce a regularization term $\mathcal{L}_{anc}$ defined as

$$\mathcal{L}_{anc} = \|\mathbf{z}_s - \mathbf{z}_s^{anc}\|_2, \tag{S2}$$

ensuring that the shape latent codes for each leaf instance do not deviate significantly from the reference shape.

Table S1. Additional ablation studies with $\mathcal{L}_{\text{bound}}$ and $\mathcal{L}_{\text{ang}}$.

Method	C-ℓ_2 [mm] ↓	NC ↑
w/o $\mathcal{L}_{\text{bound}}$	9.4	0.971
w/o $\mathcal{L}_{\text{ang}}$	4.3	0.953
NeuraLeaf (full model)	**2.1**	**0.973**

D. More Results and Analysis

We also strongly encourage readers to refer to the supplementary video for more intuitive visualizations.

Shape interpolation We show interpolation results between base shapes with distinct differences in Fig. S2, showing the smoothness and compactness of our base latent space, which is useful for CG leaf shape modeling using NeuraLeaf.

More generation examples We also show different samples in the same deformation pattern (*i.e.*, deformed by the same deform code $\mathbf{z}_d$) in Fig. S3, to evaluate the deformation consistency among different base leaf shapes.

Visualization of skinning weights We also show the results of interpolating shape codes with fixed deformation in Fig. S4 along with their skinning weights. Although skinning weights are shape-dependent, they are mostly consistent across base shapes, allowing the model to adapt to unseen leaf shapes.

Random shape generation NeuraLeaf also allows the leaf shape generation by random sampling in the shape space, as shown in Fig. S5, showing that the shape latent is reasonably constructed.

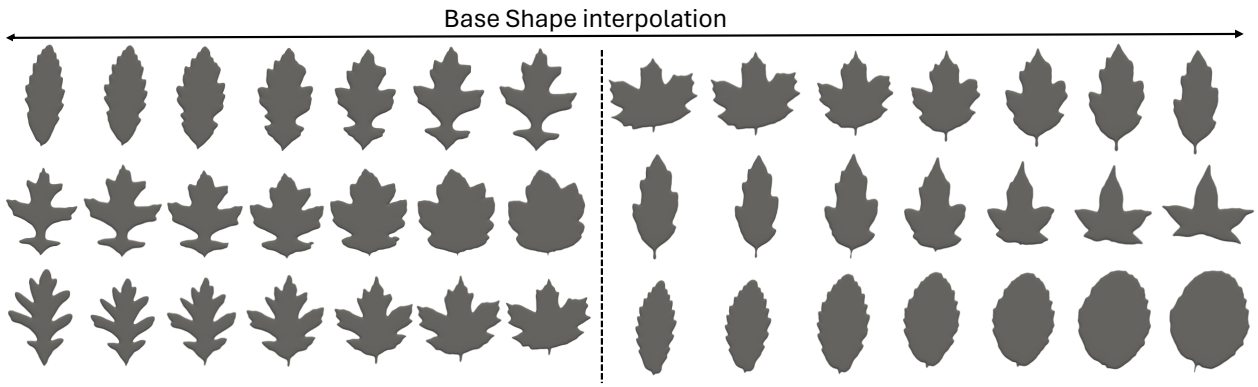

Figure S2. Interpolation between diverse base shapes, showing the smoothness of our base shape latent space.

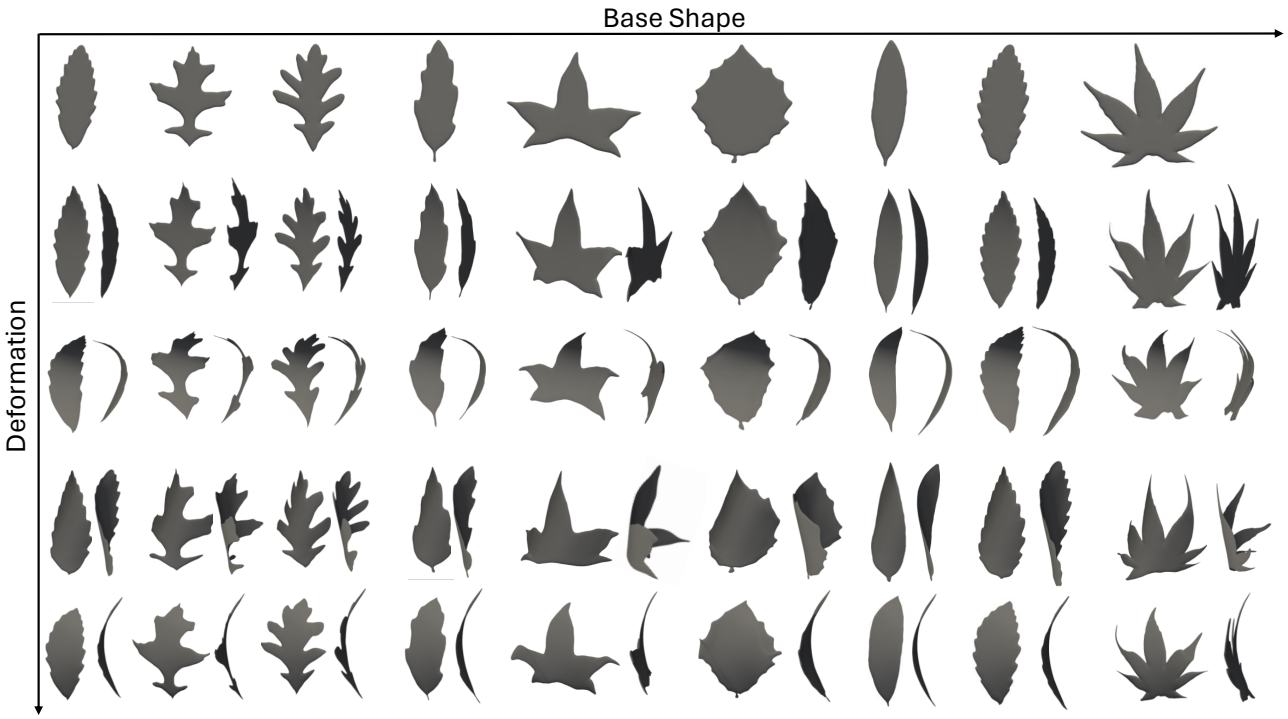

Figure S3. Deformation among different kinds of base shapes. Each row shows the results using the same deformation latent codes.

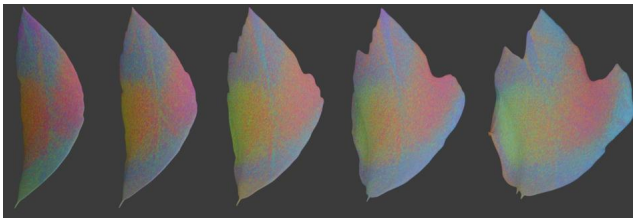

Figure S4. Visualization of skinning weights during shape interpolation.

Figure S5. Shape generation via random sampling of shape latent.

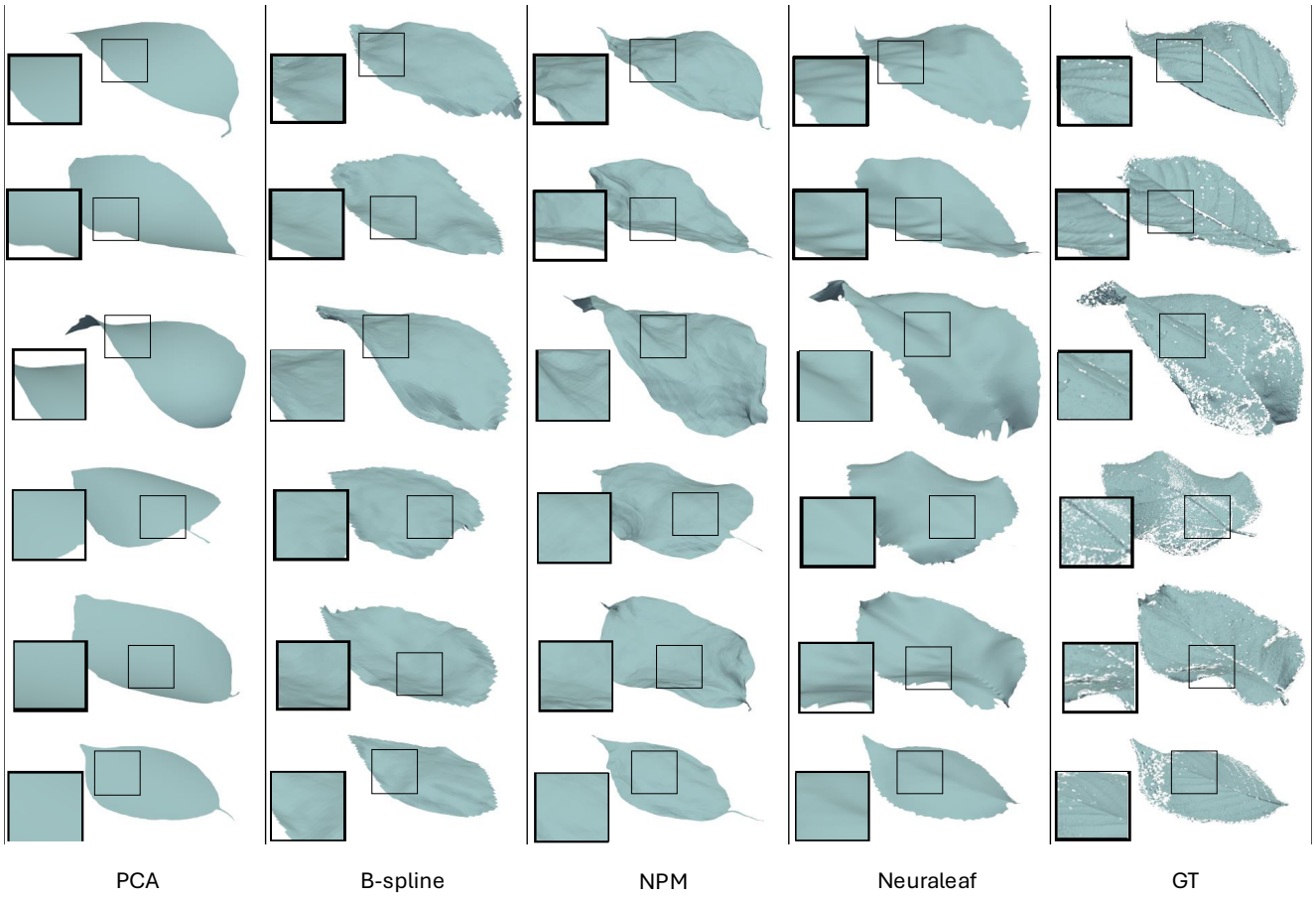

Figure S6. Additional comparison of single-leaf reconstruction.

Figure S7. Additional result of multiple leaf reconstruction.

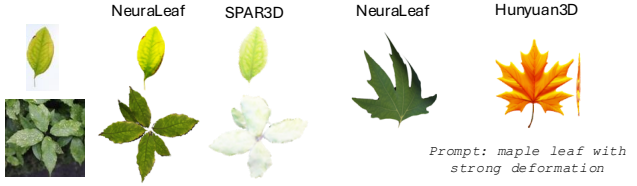

Figure S8. Comparisons with 3D model generation methods.

More reconstruction examples More reconstruction results of single and multiple leaves are shown in Fig. S6 and Fig. S7, respectively, to further emphasize NeuraLeaf's performance in the leaf reconstruction task.

Comparisons with 3D generation methods Although the domain is a bit different, we further compare NeuraLeaf to SOTA image-to-3D [5] and text-to-3D [15] models. The results in Fig. S8 show that ours produces more faithful leaf geometries, deformation, and texture.

More ablations We provide additional ablation studies specifically targeting boundary length loss and face angle loss in Table S1, demonstrating that they can help avoid overfitting to local optima.

References

[1] Matan Atzmon and Yaron Lipman. SAL: Sign agnostic learning of shapes from raw data. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[2] Derek Bradley, Derek Nowrouzezahrai, and Paul Beardsley. Image-based reconstruction and synthesis of dense foliage. *ACM Transactions on Graphics (TOG)*, 32(4):74:1–74:10, 2013. 2, 3

[3] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3D shape reconstruction and completion. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[4] Aoxiang Fan, Jiayi Ma, Xin Tian, Xiaoguang Mei, and Wei Liu. Coherent point drift revisited for non-rigid shape matching and registration. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[5] Zixuan Huang, Mark Boss, Aaryaman Vasishta, James M Rehg, and Varun Jampani. SPAR3D: Stable point-aware reconstruction of 3D objects from single images. *arXiv preprint arXiv:2501.04689*, 2025. 5

[6] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 1

[7] Gunjan Mukherjee, Arpitam Chatterjee, and Bipan Tudu. Morphological feature based maturity level identification of kalmegh and tulsi leaves. In *Proceedings of International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, 2017. 1

[8] Pablo Palafox, Aljaž Božič, Justus Thies, Matthias Nießner, and Angela Dai. NPMs: Neural parametric models for 3D deformable shapes. In *Proceedings of IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 2, 3

[9] Pascal Paysan, Reinhard Knothe, Brian Amberg, Sami Romdhani, and Thomas Vetter. A 3D face model for pose and illumination invariant face recognition. In *Proceedings of IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2009. 3

[10] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *Proceedings of International Conference on Machine Learning (ICML)*, 2019. 2

[11] Guodong Rong and Tiow-Seng Tan. Jump flooding in GPU with applications to voronoi diagram and distance transform. In *Proceedings of Symposium on Interactive 3D Graphics and Games (I3D)*, 2006. 1

[12] Shunsuke Saito, Jinlong Yang, Qianli Ma, and Michael J Black. SCANimate: Weakly supervised learning of skinned clothed avatar networks. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[13] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. RigNet: Neural rigging for articulated characters. *ACM Transactions on Graphics (TOG)*, 39 (4):58:1–58:14, 2020. 2

[14] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[15] Zibo Zhao, Zeqiang Lai, Qingxiang Lin, Yunfei Zhao, Haolin Liu, Shuhui Yang, Yifei Feng, Mingxin Yang, Sheng Zhang, Xianghui Yang, et al. Hunyuan3D 2.0: Scaling diffusion models for high resolution textured 3D assets generation. *arXiv preprint arXiv:2501.12202*, 2025. 5

[16] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017. 2