Supplementary Material for DeFormer: Integrating Transformers with Deformable Models for 3D Shape Abstraction from a Single Image

Di Liu[1], Xiang Yu[2], Meng Ye[1], Qilong Zhangli[1], Zhuowei Li[1], Zhixing Zhang[1], Dimitris N. Metaxas[1]
[1]Rutgers University [2]Amazon Prime Video

In this supplementary material, we first provide additional details about our DeFormer formulation due to the space limit of the main paper. We then provide more visualization and quantitative results to highlight the superiority of our approach. Furthermore, we show more ablation study results across various settings.

1. Details of DeFormer Formulations

1.1. Notation table

In Tab. 1, we provide the key variables in the paper, we list the symbol, variable name, state space, and notes.

1.2. Parameterized Deformable Models

Prior research works develop parameterized deformable models to represent object shapes with relatively few parameters. A notable example is in [5] which exploits computational physics in the modeling process and proposes snakes, a locally parameterized deformable model. The snake formulation employs a force field computed from data space to fit the target shape. Nevertheless, snakes using locally defined deformations cannot intrinsically offer shape abstractions. [9] partially addresses the problem of shape abstraction by using superquadric ellipsoids that explicitly deform using a few global parameters. [10] develops a new physics-based framework offering multi-scale global and local deformations, and demonstrates its power using deformable superquadrics. Although their framework addresses complex shape modeling and motion estimation of objects, it relies on handcrafted parameter initialization [4].

Primitive formulation. We employ superquadrics as our basic primitive formulation for the global deformation $\mathbf{s}$. Each superquadric surface $\mathbf{e}$ is explicitly defined by a set of shape-related parameters:

$$\mathbf{e} = a_0 \begin{bmatrix} a_1 \cos^{\varepsilon_1} u \cos^{\varepsilon_2} v \\ a_2 \cos^{\varepsilon_1} u \sin^{\varepsilon_2} v \\ a_3 \sin^{\varepsilon_1} u \end{bmatrix}, \tag{1}$$

where $-\pi/2 \leq u \leq \pi/2, -\pi \leq v \leq \pi$. Here, a_0 is a scaling parameter, a_1, a_2, a_3 denote the aspect ratio for x-, y-, z- axes, respectively, and $\varepsilon_1, \varepsilon_2$ are squareness parameters.

Global deformations. To improve the geometric coverage of these primitives, we introduce parameterized tapering and bending deformations. These additional global deformations are defined as continuously differentiable and commutative functions following [7]. Specifically, due to their suitability for natural objects, we integrate linear tapering and bending of the superquadric $\mathbf{e} = (e_1, e_2, e_3)^\top$ into one single parameterized deformation $\mathbf{T}$ and give the formulation of the reference shape as:

$$\begin{aligned} \mathbf{s} &= \mathbf{T}(\mathbf{e}, t_1, t_2, b_1, b_2, b_3) \\ &= \begin{pmatrix} (\frac{t_1 e_3}{a_0 a_3} + 1)e_1 + b_1 \cos(\frac{e_3 + b_2}{a_0 a_3})\pi b_3 \\ (\frac{t_2 e_3}{a_0 a_3} + 1)e_2 \\ e_3 \end{pmatrix}, \end{aligned} \tag{2}$$

where t_1, t_2 are the tapering parameters, b_1, b_2, b_3 are the magnitude, location, and influence region of bending, respectively. The learnable parameters for $\mathbf{s}$ is then denoted as $\mathbf{q}_s = (a, \epsilon, t, b)$, where $a = (a_0, a_1, a_2, a_3), \varepsilon = (\varepsilon_1, \varepsilon_2), t = (t_1, t_2), b = (b_1, b_2, b_3)$.

In this study, we only give a limited number of examples for the primitives as well as global deformations. However, global deformations are not restricted to only tapering, bending, and twisting. Any other deformations (*e.g.*, shearing) that can be given as a continuous and parameterized function can be similarly integrated into our model. In addition, the type of primitives is not restricted to only superquadric shapes. Other primitive forms (*e.g.*, spheres, convexes, supertoroids, etc.) can also be integrated into our unified framework, which opens up new possibilities for a wider application range of shape abstraction tasks.

1.3. Re-projection

We use the differentiable projection module proposed in [6] to obtain the projected image $\mathbf{x}_{\text{proj}}$ from the same viewpoint as the input image $\mathcal{X}$. The viewpoint associated with the input image $\mathbf{x}_{\text{proj}}$ is characterized by the rotation and translation parameters of the camera (*i.e.*, camera motion σ) in the image space. We obtain the estimation of camera motion parameters $\mathbf{q}_{c'}$ and $\mathbf{q}_{\theta'}$ using the encoder of MsBiT.

Table 1: **Notations.** For the key variables in the paper, we list the symbol, variable name, state space, and notes.

Symbol	Variable name	State space	Notes
$\mathbf{c}_\sigma$	Camera translation	$\mathbb{R}^3$	
$\mathbf{c}$	Primitive translation	$\mathbb{R}^3$	
$\mathbf{R}$	Primitive rotation	$\mathbb{R}^{3\times3}$	
$\mathbf{R}_\sigma$	Camera rotation	$\mathbb{R}^{3\times3}$	
$\mathbf{s}$	Global deformations	$\mathbb{R}^3$	
$\mathbf{e}$	Superquadric surface	$\mathbb{R}^3$	
$\mathbf{d}$	Local deformations	$\mathbb{R}^N$	N: sampling points on primitive surface
$\mathbf{q}_{c'}$	Parameters for camera translation	$\mathbb{R}^3$	$\mathbf{q}_{c'} = \mathbf{c}_\sigma$
$\mathbf{q}_c$	Parameters for primitive translation	$\mathbb{R}^3$	$\mathbf{q}_c = \mathbf{c}$
$\mathbf{q}_{\theta'}$	Parameters for camera rotation	$\mathbb{R}^4$	$\mathbf{q}_{\theta'}$ is a 4D quaternion related to $\mathbf{R}_\sigma$
$\mathbf{q}_\theta$	Parameters for primitive rotation	$\mathbb{R}^4$	$\mathbf{q}_\theta$ is a 4D quaternion related to $\mathbf{R}$
$\mathbf{q}_s$	Parameters for global deformations	$\mathbb{R}^{11}$	$\mathbf{q}_s = (\mathbf{a}, \varepsilon, \mathbf{t}, \mathbf{b}), \mathbf{a} \in \mathbb{R}^4, \varepsilon \in \mathbb{R}^2, \mathbf{t} \in \mathbb{R}^2, \mathbf{b} \in \mathbb{R}^3$
$\mathbf{q}_d$	Parameters for local deformations	$\mathbb{R}^N$	$\mathbf{q}_d = \mathbf{d}$ as we use one to one mapping
$\mathbf{B}$	Rotation related matrix	$\mathbb{R}^{3\times4}$	$\mathbf{B} = \partial\mathbf{Rp}/\partial\mathbf{q}_\theta$
$\mathbf{J}$	Jacobian matrix	$\mathbb{R}^{3\times11}$	$\mathbf{J} = \partial\mathbf{s}/\partial\mathbf{q}_s$, $\mathbf{J} \in \mathbb{R}^{3\times6}$ if no global deformations

The reconstructed surface $\mathbf{x}$ is then projected from the estimated viewpoint using the differentiable projection module to obtain 2D image projections $\mathbf{x}_{\text{proj}}$. If the estimated camera motion and the target shape reconstructions are correct, the 2D projections $\mathbf{x}_{\text{proj}}$ will match the input image $\mathcal{X}$. This process is formulated as a cycle-consistency regularization shown in Sec. 3.4 of the main paper.

1.4. Intersections between Primitives

We follow the physics-based deformable models (DMs) [34,47] and avoid the collisions (intersections) between primitives by checking for primitive inter-penetration in each training iteration. If two primitives penetrate each other [34], we assign two equivalent and opposite collision forces f_n and $-f_n$ that are proportional to the distance between each pair of selected points on the two primitives. These two forces are added to the respective points on the two inter-penetrating primitives, respectively, to adjust the external forces f and thus push the primitives to separate from each other.

2. Network Architecture

The architecture of Multi-scale Bi-directional Transformer (MsBiT) and Multi-scale Holistic Fusion (MHF) is given in Fig. 1. For the BiTrans module in Fig. 1(a), the local deformation map l_i is first projected to $Q/K/V$ with a depth-wise separable convolution [1] due to its computational efficiency and capability in capturing local responses. Note that we still employ 1×1 convolution to project g_i with a much smaller size to $\overline{Q}/\overline{K}/\overline{V}$, in order to avoid any additional noise introduced during the padding in the depth-wise separable convolution. Due to the symmetry of the query and key dot product, we achieve the cross-attention map by transposing the dot product matrix to aggregate the global and local information of the primitive:

$$\begin{aligned}(l_i^j, g_i^j) &= \text{BiTrans}(l_i^{j-1}, g_i^{j-1}) \\ &= (\text{softmax}(\frac{Q\overline{K}^\top}{\sqrt{d}})\overline{V}, \text{softmax}(\frac{\overline{Q}K^\top}{\sqrt{d}})V),\end{aligned} \tag{3}$$

where l_i^j and g_i^j are the "Bi-channel" j-th layer outputs. Both the two feature maps l_i and g_i are continuously updated and improved through the encoder-decoder architecture for the final prediction. This mechanism enables efficient feature aggregation of the global motion and deformations while preserving the ability to capture local responses for non-rigid deformation estimation.

For the MHF module in Fig. 1(b), we first flatten and concatenate the holistic feature maps g_1-g_3 from different scales into a 1D sequence, which is further fed into the standard Transformer block with MHSA for feature aggregation. The output sequence is then chunked and folded back to the holistic maps g_1'-g_3' at the corresponding scales in the decoder for the estimation of local non-rigid deformations. In addition, the MHF module also outputs a chunked holistic feature map g_0' for the following global parameter estimation.

Details. Each of the two Conv Stems used in the paper consists of two 3×3 convolutional layers with Batch Norm and ReLU to embed the feature maps to $4 \times$ down-sampling/up-

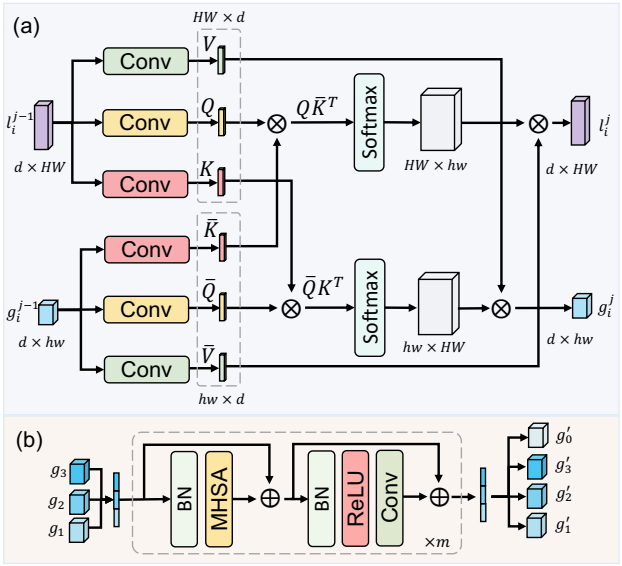

Figure 1: The architecture of (a) Bi-directional Transformer (BiTrans) and (b) Multi-scale holistic Fusion (MHF).

Table 2: Parameter details.

size	m 2	d -	l_i^j, l_i, l_i' $(d \times H \times W)$	g_i^j, g_i, g_i' $(d \times h \times w)$	Q, K, V $(HW \times d)$	$\overline{Q}, \overline{K}, \overline{V}$ $(hw \times d)$
$i=0$	-	16	$16 \times 32 \times 32$	$16 \times 2 \times 2$	1024×16	4×16
$i=1$	-	32	$32 \times 16 \times 16$	$32 \times 2 \times 2$	256×32	4×32
$i=2$	-	64	$64 \times 8 \times 8$	$64 \times 2 \times 2$	64×64	4×64
$i=3$	-	128	$128 \times 4 \times 4$	$128 \times 2 \times 2$	32×128	4×128

sampling token maps. In Tab. 2 we provide details for the parameters used.

3. Additional Results

3.1. Reconstruction Accuracy

In Figs. 2 and 3, we provide additional qualitative results on various ShapeNet categories. We train our model with 3 and 4 primitives for cars and airplanes, respectively, for accurate abstractions. We compare to CvxNets [2] using 25 primitives and NP [8] using 5 primitives, which empirically leads to their best performance. We observe that DeFormer yields more geometrically accurate and semantically meaningful abstractions than NP and CvxNets with multiple primitives.

3.2. Computational Cost

We compare the computational cost of DeFormer against the baseline methods in the main paper. We report the numbers measured on one NVidia A100 GPU.

Training. We use the same batch size and number of primitives for all methods and measure the time cost to perform 100 forward/backward passes. We load the data using a single batch and keep it in GPU memory to remove the overhead of data loading. The time consuming is given in Tab. 3.

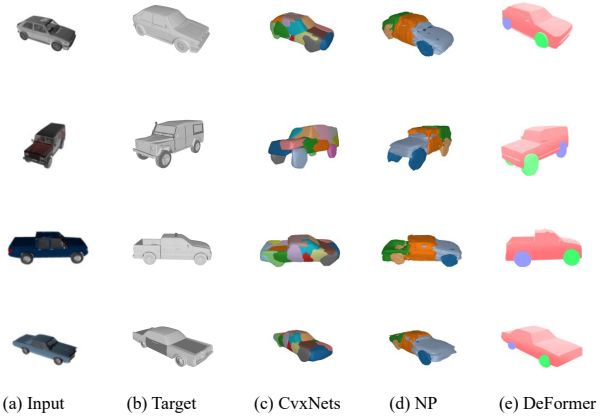

Figure 2: Abstraction visualization on cars compared to SOTA primitive-based methods, including CvxNets [2] and NP [8] with 25 and 5 primitives, respectively. Ours applies 3 primitives (1 for body and 2 for front and rear wheels) and achieves better part consistency.

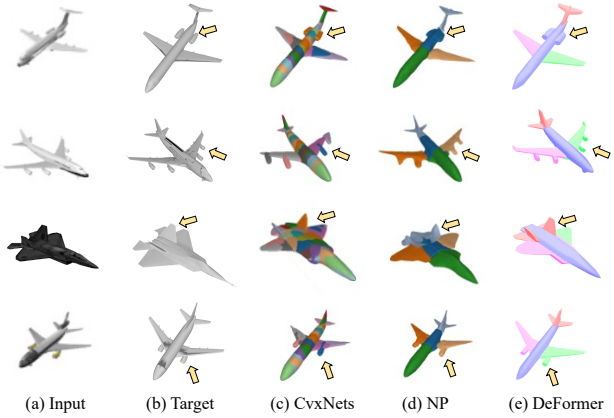

Figure 3: Abstraction visualization on airplanes compared to SOTA primitive-based methods, including CvxNets [2] and NP [8] with 25 and 5 primitives, respectively. Ours applies 4 primitives (1 body, 2 wings, and 1 tail) and achieves better part consistency. Reconstruction performance for fine details is indicated with arrows.

Inference. The network takes one input image with size 224×224 for inference. In Tab. 3 we report the average inference time over 100 runs.

4. Additional Ablation Study

We train DeFormer on three categories of *ShapeNet* using ResNet18 [3] and the proposed MsBiT backbone, respectively and compare the shape reconstruction accuracy

Table 3: Computational cost of DeFormer, compared to the baseline methods.

	Suq	H-Suq	CvxNet	NP	DeFormer
Train. time/batch (ms)	48.64	63.50	84.36	256.19	371.79
Memory (GB)	6.15	8.21	7.47	10.52	21.70
Infer. time (s)	11.23	11.78	7.87	8.85	15.36

in terms of IoU and Chamfer-L_1 distance.

Table 4: Ablation study on the network backbone. We train DeFormer with the proposed MsBiT and ResNet18 on three categories of *ShapeNet*. We provide results in terms of IoU, Chamfer-L_1 distance.

Backbone	ResNet18			MsBiT		
Category	table	lamp	sofa	table	lamp	sofa
IoU (↑)	0.535	0.417	0.715	0.546	0.422	0.729
Chamfer-L_1 (↓)	0.097	0.148	0.102	0.081	0.141	0.088

References

[1] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017. 2

[2] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 31–44, 2020. 3

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[4] Timothy N Jones and Dimitris N Metaxas. Image segmentation based on the integration of pixel affinity and deformable models. In *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No. 98CB36231)*, pages 330–337. IEEE, 1998. 1

[5] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988. 1

[6] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7708–7717, 2019. 1

[7] Dimitris N Metaxas. *Physics-based deformable models: applications to computer vision, graphics and medical imaging*, volume 389. Springer Science & Business Media, 2012. 1

[8] Despoina Paschalidou, Angelos Katharopoulos, Andreas Geiger, and Sanja Fidler. Neural parts: Learning expressive 3d shape abstractions with invertible neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3204–3215, 2021. 3

[9] Alex P Pentland. Perceptual organization and the representation of natural form. In *Readings in Computer Vision*, pages 680–699. Elsevier, 1987. 1

[10] Demetri Terzopoulos and Dimitri Metaxas. Dynamic 3 d models with local and global deformations: deformable superquadrics. *IEEE Transactions on pattern analysis and machine intelligence*, 13(7):703–714, 1991. 1