RaBit: Parametric Modeling of 3D *Biped Cartoon* Characters with a Topological-consistent Dataset Supplemental Material

Zhongjin Luo[1*] Shengcai Cai[1,3*] Jinguo Dong[1] Ruibo Ming[1,4]
Liangdong Qiu[2,1] Xiaohang Zhan[3] Xiaoguang Han[1,2†]

[1]SSE, CUHKSZ [2]FNii, CUHKSZ [3] Huawei Technologies Co., Ltd. [4]Tsinghua University

1. Details of *3DBiCar*

Image Styles. Fig. 1 shows representative images for the 4 styles with different appearances. We define them based on their different sources: *picture book* - cropped from e-books of children, *computer designed* - made by artists using software, *hand drawn* - drawn by kids, *toy* - captured from real toys.

Figure 1. A representative example from different image styles. (a) picture book, (b) computer designed, (c) hand drawn, (d) toy.

Shape Modeling Procedure. We recruit six professional artists to create 3D corresponding character models using Blender according to the collected reference images. The key to building a linear parametric shape model lies in maintaining a unified mesh topology. To achieve this, all six artists are required to craft 3D models by deforming the template mesh under the constraints of the predefined landmarks. Each artist owns over six years of modeling experience and each character takes around 1 hour on average. The modeling result is required to be matched with the reference images as much as possible. To maintain visual quality and topological consistency, we have established a review committee of ten members to assess the models based on reference images and predefined landmarks.

*Equal contribution
†Corresponding Author

2. Details of *RaBit*

Shape Space. As illustrated in Fig. 2, *RaBit* is able to express the *basic geometry* of diverse shapes in *3DBiCar* with *low-dimensional vectors* (100 in our experiments). Such ability of *RaBit* can well facilitate the construction of learning-based regression methods for inferring reasonable shapes from images or sketches, as demonstrated in our downstream tasks. Biped cartoon is known as a popular character style in gaming and filming. *RaBit* spans a wider range of species than existing human model [5, 6]. However, due the use of the holistic PCA model, *RaBit* may struggle to represent local geometric details and may result in undesirable entanglement, as shown in Fig. 3. Conducting parametric modeling for diverse shapes is a fundamental problem, but it has received little methodological evolution in the past due to the lack of data. We hope that our proposed dataset can inspire further research in this area.

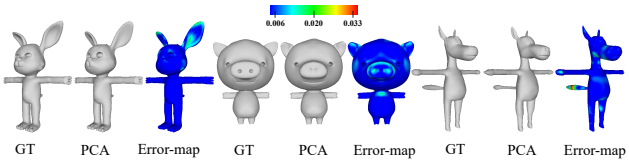

Figure 2. Comparison of shapes reconstructed by *RaBit* with GT.

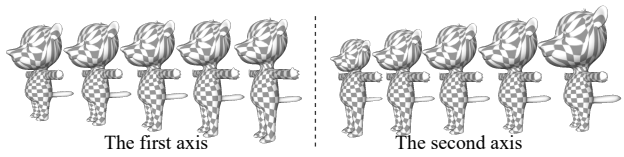

Figure 3. An illustration of the first two axes of shape space in *RaBit*.

Visualization of Topological Consistency. Good correspondence of training data is essential for constructing a linear

shape model and preserving the topological consistency of reconstructed models. Getting topological consistency in manual modeling is intrinsically challenging. To do so, we put much effort to construct *3DBiCar*, including template designing, landmark guidance, review committee for careful checking. In Fig. 4, we use checkboard texture mapping for visualizing the correspondence of representative examples sampled from *RaBit*'s shape space.

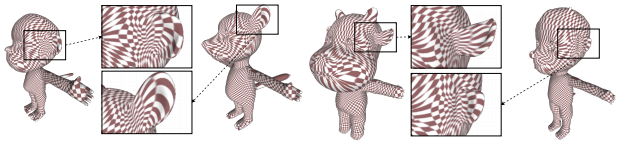

Figure 4. An illustration of the mesh correspondence.

Eyeball Reconstruction. In our implementation, we approximate an eyeball as a sphere. Generally, a sphere is determined by its center and radius. As shown in the Fig. 5, in *RaBit*, an eyeball's center $\mathbf{o_e}$ and radius r_e is computed as follows,

$$r_e = c_1 r_s, \tag{1}$$

$$d_e = c_2 r_s, \tag{2}$$

$$\mathbf{o_e} = \mathbf{o_s} - d_e \mathbf{n}, \tag{3}$$

where r_s and $\mathbf{o_s}$ is the radius and the center of the 3D circle, computed by the least square fitting with the landmark points of the eye socket. d_e denotes the Euclidean Distance between $\mathbf{o_s}$ and $\mathbf{o_e}$ and $\mathbf{n}$ the normal of the 3D circle. c_1 is the mean value of d_e/r_s of all models in *3DBiCar*, while c_2 is the mean value of r_e/r_s. Both c_1 and c_2 are precomputed constant values.

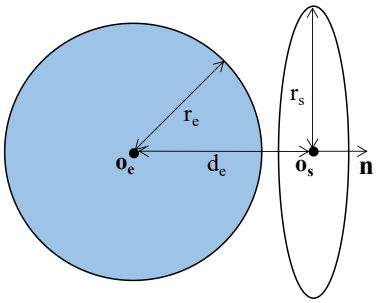

Figure 5. An illustration of eyes computation. $\mathbf{o_e}$ is the center of the eye and r_e is the radius of the eye. $\mathbf{o_s}$ and r_s are the center and the radius of the orbit, respectively.

Implementations. The shape model of *RaBit* is learned from 1,050 models of *3DBiCar* using PCA [5, 7]. For pose modeling, *RaBit* utilizes the consistent skeleton and skinning weight matrix defined in *3DBiCar*. Note that both *3DBiCar* and *RaBit* currently does not support the animation of tails, which will be explored in our future work. As for texture modeling, 1,050 raw textures from *3DBiCar* were adopted and extended to 21,000 training data with image-level augmentations (e.g., flipping, and adjusting HSV). *Rabit*'s texture generator follows the architecture of StyleGAN2 [4] and is trained with the following setting: the dimensionality of Z with 512, the output resolution with $1024 \times 1024 \times 3$, the learning rate with 3×10^{-4}, the batch size with 32, the Adam optimizer with $\beta_1 = 0$, $\beta_2 = 0.99$, $\epsilon = 10^{-8}$. The training is performed on a server with 4 Nvidia RTX 3090Ti GPUs.

3. Details of *BiCarNet*

Data Preparation. We split *3DBiCar* into a training set (1,050 image-model pairs) and a testing set (450 pairs). To support a stable training of *BiCarNet*, we augment a large number of synthetic paired data with the help of *RaBit*. Specifically, we generate a series of shape vectors by interpolating between the 1,050 models' shape parameters. Fig. 6 shows the representative results of interpolated shapes. For pose augmentation, a variety of poses from other datasets (e.g., Human3.6M [3]) are retargeted to *RaBit*'s pose space, as shown in Fig. 7. Furthermore, 1,050 raw textures are also utilized to generate synthetic texture maps by interpolating with *RaBit*, as shown in Fig. 8. The above augmentations finally produce 13,650 models with texture and pose. These models are then rendered into images from different camera views for training.

Implementations. In our implementation, for the shape and pose regression modules, we utilize two ResNet-50 blocks to embed the input image ($512 \times 512 \times 3$) to a 100-dimensional shape vector and a 69-dimensional pose vector, respectively. For the texture module, we adopt pSp-encoder [8] to learn a 512-dimensional texture vector from the image. As for the part-sensitive texture reasoner, we use pSp [8] as the basic building block and learn multiple local UV textures ($256 \times 256 \times 3$) from the input. pix2pixHD [9] is employed as the fusion module (Fuser), which takes the $1024 \times 1024 \times 3$ coarsely-blended texture map as input and outputs fine texture maps with the same resolution.

Part-Sensitive UVs. As shown in Fig. 9, we design five individual UV-mappings for significant parts, i.e., nose, ears, horns, eyes, and mouth. These part UVs enlarge five constant regions of the global UV mapping. Five lightweight encoder-decoder branches are adopted to learn the appearances of these local regions from the input image, respectively. The learned part UVs could then be remapped to their corresponding areas on the global UV map, resulting in a blended texture.

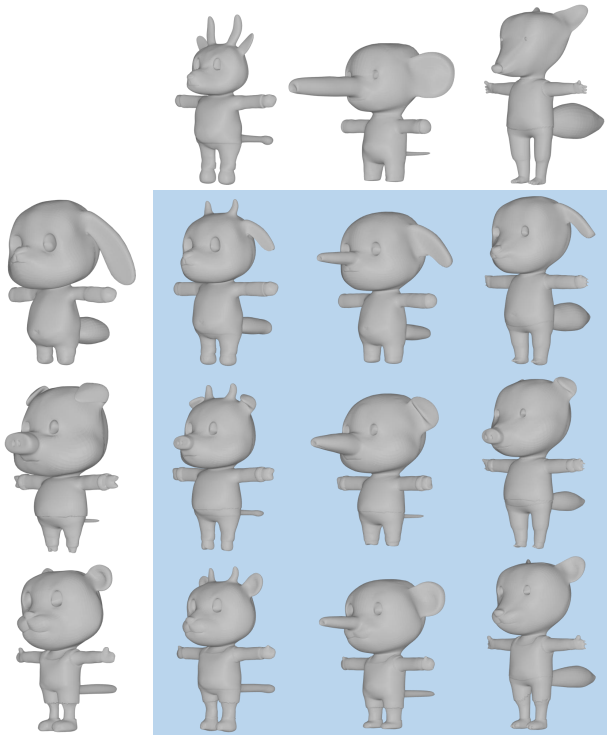

Figure 6. An illustration of interpolated shapes. Models from the top row and left column are from *3DBiCar*. Other models with blue backgrounds are obtained by interpolating the leftmost and uppermost models with the help of *RaBit*.

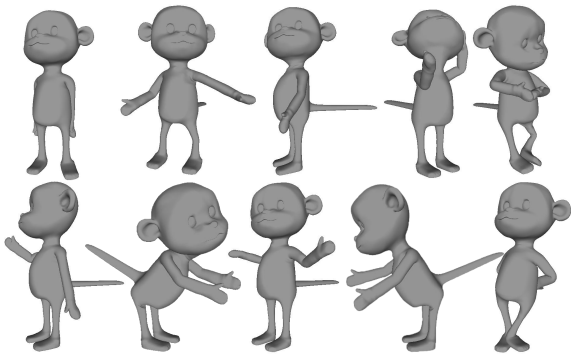

Figure 7. An illustration of diverse poses transferred from pose datasets.

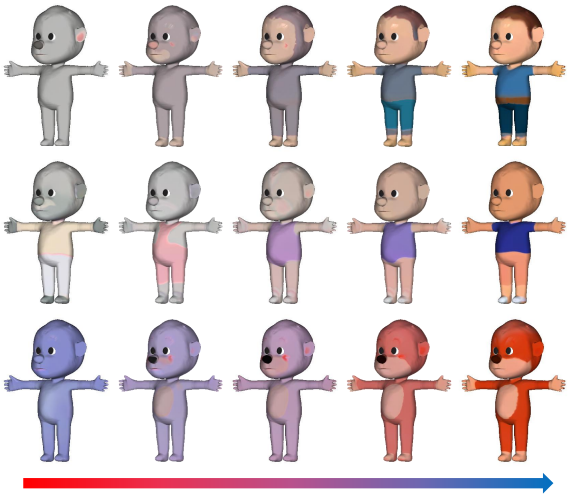

Figure 8. An illustration of synthesized texture maps. For each row, the leftmost and the rightmost textures are from *3DBiCar*, while the other three textures are interpolated results generated by *RaBit* under different weights.

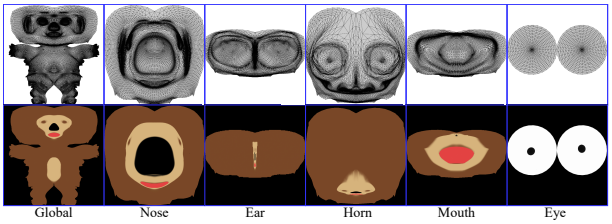

Figure 9. An illustration of our UV layouts and textures.

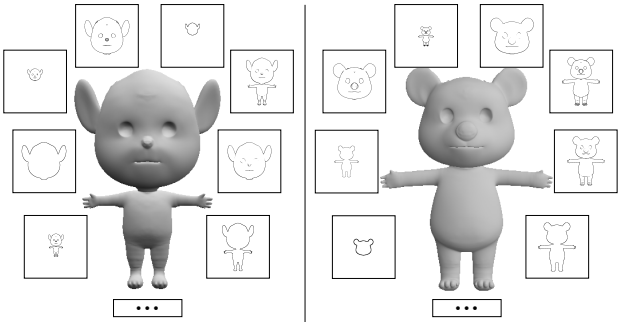

Figure 10. An illustration of rendered sketches used for training.

4. Details of Sketch-based Modeling

Data preparation. We first sample 12,000 shape vectors randomly and feed them to *RaBit* to generate 3D cartoon characters with diversified shapes. Then the suggestive contour [1, 2] is applied to render the front-view sketches with different abstraction levels and obtain 108,000 sketch-model pairs. Fig. 10 shows examples of rendered sketches.

Implementations. As shown in Fig. 11, we first adopt one ResNet-50 module and three MLPs as the encoder-decoder architecture, mapping the input sketch 512×512 to 100-dimensional shape parameters. Then the generated shape parameters are fed to *RaBit* to reconstruct the corresponding 3D model. We train the network with a batch size of 100 and a learning rate of 3×10^{-4} with the Adam optimizer. Moreover, we use the L_1 loss to measure the difference between the predicted shape parameters and the ground truth. Our sketch-based modeling interface is implemented with the QT framework. CGAL is adopted for 3D geometry processing.

As shown in the video, running on a personal computer with an Intel i7-7700 CPU, 16GB RAM, and a single Nvidia GTX 2080Ti GPU, our modeling application supports real-time feedback.

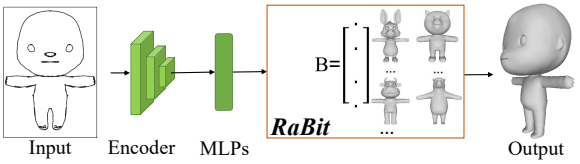

Figure 11. The pipeline of our sketch-based modeling. Given a sketch 512×512 as input, we employ one ResNet-50 module and three MLPs to embed the input to 100-dimensional shape parameters. The output shape parameters are fed to *RaBit* to reconstruct the corresponding 3D model.

References

[1] Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. Suggestive contours for conveying shape. In *ACM SIGGRAPH 2003 Papers*, pages 848–855. 2003. 3

[2] Xiaoguang Han, Chang Gao, and Yizhou Yu. Deepsketch2face: a deep learning based sketching system for 3d face and caricature modeling. *ACM Transactions on graphics (TOG)*, 36(4):1–12, 2017. 3

[3] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, 2014. 2

[4] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020. 2

[5] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, Oct. 2015. 1, 2

[6] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 10975–10985, 2019. 1

[7] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011. 2

[8] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2287–2296, 2021. 2

[9] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2