Body-shape Transfer for Super Deformation of 3D Character Models

Peng Wang Yoshihiro Kanamori Yuki Endo Jun Mitani
Department of Computer Science, Graduate School of Systems and Information Engineering, University of Tsukuba,
Email: wangpeng@npal.cs.tsukuba.ac.jp, {kanamori,endo,mitani}@cs.tsukuba.ac.jp

***Abstract*—We present a technique for semi-automatically super-deforming 3D character models. Super deformation is a unique style of exaggeration, often seen in Japanese animation and manga, to enhance characters' cute appearances. Specifically, super-deformed (SD) characters are chubby with stubby limbs and oversized heads. Given a reference SD model with a rig and a target character model, our method automatically calculates a rig of the target model, measures the body shapes, and shortens the target limbs and enlarges the target head by transferring the reference body shape. Our method further makes the target face younger by applying a simple deformation based on a biological insight. We demonstrate that our method can create visually-pleasing SD character models quickly.**

***Keywords*-3D modeling; super deformation; rigging;**

I. Introduction

Japanese animation (or *"anime"*) and manga have attracted not only children but also adults around the world. In such Japanese culture, super deformation is a unique style of exaggeration to enhance characters' cute appearances; super-deformed (SD) characters are typically chubby with stubby limbs and oversized heads, and resemble small children [1]. SD characters often appear not only in anime or manga but also in commercials and character figures. Figure 1 shows a pair of normal and corresponding SD examples of a character "Unity-chan"[1]. SD character models are currently created by skilled artists via a burdensome manual process, which hinders widespread use of SD characters. We thus focus on semi-automatic creation of SD models from normal character models.

However, designing a "good" SD model from scratch is not easy even for professional artists, as we confirmed through an interview with a character figure company. This is because the definition of super deformation is ambiguous; each SD model has its own appropriate style of exaggeration and simplification, depending on various factors such as the character's personality, the character design in the story series, and the style in specific product series. On the other hand, if the designer can start from a base SD model that is close to the output model, the designer can skip basic modeling and focus on detail modification. We thus let the user input an appropriate reference SD model as a hint for avoiding excessive trial-and-errors in the early design stage.

[1]The models were obtained from http://www.unity-chan.com

Figure 1: "Unity-chan" (left) and its super-deformed (SD) version designed by an artist (right).

In this paper, we propose a semi-automatic method that can accelerate the modeling process of SD characters. The inputs of our method are a reference SD model with a rig and a target character model without any rig (Figure 2). Here a rig is a standard data structure commonly used in character animation, consisting of a skeleton (a tree structure of joints and bones) and weight maps (weights of bones for the mesh vertices). Our method automatically calculates the target rig, measures body-shape parameters, and shortens the target limbs as well as enlarges the target head by transferring the reference body shape. Our method further makes the target face younger by applying simple deformation based on the biological insight by D'Arcy Thompson [2]. Our method is automatic except for the face infantilization step where the user tweaks the face shape by simply tuning a single parameter. We demonstrate that our method can create visually-pleasing SD character models quickly.

II. Related Work

Shape deformation has been long studied in computer graphics, and there are deformation techniques used for human characters. For example, linear blend skinning [3] and its successors [4] have been standard techniques for animating characters. Body-shape control has been actively studied. Blantz and Vetter published a seminal work for parametric control of human face model [5]. They scanned 199 human faces and constructed a statistical morphable model of human face based on principal component analysis. This

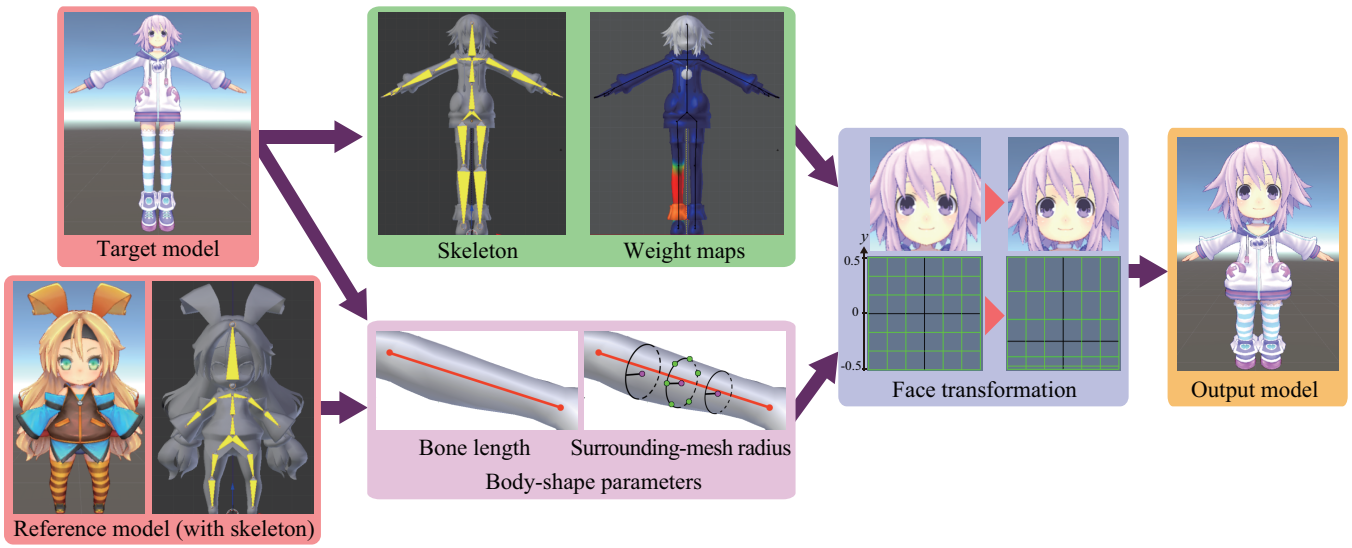


Figure 2: Overview of our system. Given target and reference SD models, our method calculates their body-shape parameters and a target skeleton as well as weight maps. After applying a simple transformation to the target face, we obtain an output SD model.

idea is later extended to morphable human body models [6], [7]. Apart from statistical models, there are also techniques for exaggerating body parts according to a user-specified importance map [8] and deforming a body shape based on an anatomical muscle model [9]. These methods are not intended for SD character models.

The most related method to ours is that proposed by Shen et al. [10]. Their method aims at super-deforming various mesh models such as human characters and monsters. The important differences with ours are as follows. (i) Their method does not use a reference model and enforces the user to manually tweak parameters for controlling shape optimization. Our method, on the other hand, does not rely on optimization, provides quick feedback to the user, and can avoid excessive trial-and-errors by obtaining body-shape parameters from a given reference SD model. (ii) Their method implicitly assumes that an input character model consists of only a single mesh. However, character models often have cloth meshes as well as body meshes, which might cause inconsistent deformation between the meshes (see Section IV-B). Our method can handle models with multiple meshes. (iii) To deform a character's face, their method deforms the head shape so that it approximates a sphere. However, this sphere approximation is not sufficient to making the face look younger. For this, our method applies a biology-inspired transformation to the head model.

III. Our Algorithm

A. Overview

Figure 2 shows the overview of our system. Our inputs are two character models; one is the *target model*, which has normal body proportion without any skeleton or weight maps. The other is the *reference model*, which is an SD model with a skeleton and weight maps, selected by the user as a reference. We assume both models are in a rest pose and have cloth and hole-free body meshes separately. A rest pose is common and simplifies the measurement and deformation of body shapes. Hole-free body mesh is required for calculating weight maps [11]. The key idea is to transfer the body shape from the reference model to the target model for automatically determining the basic shape of the output SD model. This goal is accomplished as follows. First, we calculate a skeleton and corresponding weight maps for the target model (Section IV). We then measure body-shape parameters, i.e., each bone's length and distance to the surrounding mesh, for both the target and reference models, and super-deform the target model automatically (Section V). The user can further make the target face look younger by lowering the eye positions and rounding the chin based on a biology-inspired transformation (Section VI). The user can optionally add manual modifications, e.g., deformation of hair polygons, to obtain the output model. We describe the details in the following sections.

IV. Skeleton and Weight Map Calculation

A. Skeleton definition

We use a skeleton often used in linear blend skinning for character animation. A skeleton is a tree structure where joints and bones serve as nodes and edges (Figure 3, left). Let N and M be the numbers of joints and bones respectively. The position of joint $j(j = 1, 2, ..., N)$ is denoted by $\mathbf{p}_j$. Bone $i(i = 1, 2, ..., M)$ contains a parent

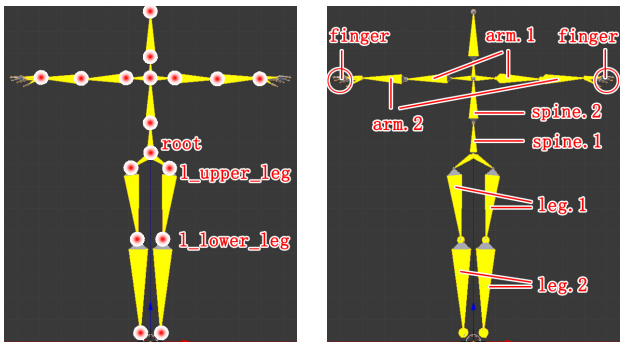


Figure 3: Tree structure consisting of joints and bones (left), and bone groups (right).

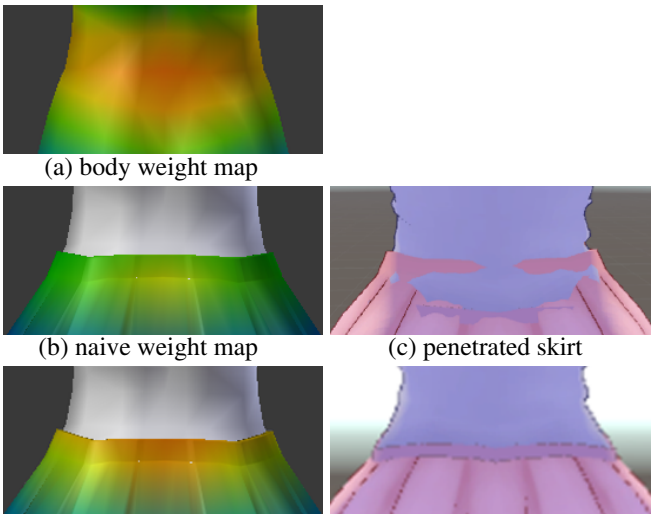
node $parent(i)$ and child node(s) $child(i)$, and its length is $l_i = ||\mathbf{p}_{parent(i)} - \mathbf{p}_{child(i)}||$. If a parent node has multiple children nodes, we calculate the bone length for each pair of parent and child nodes. We assume both skeletons of the reference and target models have the same structure, i.e., the same numbers of joints and bones with the same connections. We further separate bones into user-defined *bone groups* (Figure 3, right), and apply similar deformations to bones in each bone group G.

B. Calculation of skeleton and weight maps

Our method automatically calculates a skeleton and weight maps of the target model. For a skeleton, we employ the ray-casting based method by Lopez et al. [12] using a publicly-available implementation that can handle even fingers automatically. For weight maps, we use the optimization-based method by Baran et al. [11] as follows.

The method by Baran et al. [11] automatically calculates weight maps by solving the thermal diffusion equation. Using this method we obtain weight maps consisting of weight w_{ki} of each bone i for vertex k (where $w_{ki} \geq 0$ and $\sum_{i=1}^{M} w_{ki} = 1$). This method works well with single-mesh models, as demonstrated in the previous study [10]. However, most character models wear multiple clothes that are represented as non-manifold meshes separated from the body meshes; if we apply the method by Baran et al. [11] to both cloth and body meshes naïvely, they will have different weights at their contact part and therefore deform differently, resulting in penetrations or gaps between cloth and body meshes (Figure 4(c)).

To ensure that the weights of cloth and body vertices match at contact parts, our method first calculates the weights for the body mesh, and then transfers the weights to the cloth meshes by linearly blending weights of the nearest body vertices. This modification yields consistent deformation of the clothes and body (Figure 4(e)).



(a) body weight map

(b) naive weight map

(c) penetrated skirt

(d) transferred weight map

(e) consistently-deformed skirt

Figure 4: Comparison of cloth weight map calculation. (b) The naïvely calculated weight map of the skirt does not match (a) that of the body, which causes (c) penetration. (d) By transferring the weights from the body to the skirt, (e) the skirt deforms consistently with the body.

V. Super-deformation Using Body-shape Parameters

A. Body-shape parameter measurement

Our method automatically measures body-shape parameters based on the method by Lopez et al. [12]. The body-shape parameters used in our method are (i) the length l_i of each bone i and (ii) the radius r_i of a cylinder approximating the mesh surrounding bone i, which we call the *surrounding-mesh radius*. We label each of them with labels tar (for the target model) and ref (for the reference model). Let $L^{ref} = \{l_1^{ref}, l_2^{ref}, ..., l_M^{ref}\}$ and $R^{ref} = \{r_1^{ref}, r_2^{ref}, ..., r_M^{ref}\}$ be the sets of bone lengths and surrounding-mesh radii of the reference model, respectively. L^{tar} and R^{tar} are defined similarly for the target model.

Body-shape parameters are calculated as follows. Bone length l_i is calculated simply as the distance between parent and child nodes of bone i. Surrounding-mesh radius r_i is calculated based on ray casting. Figure 5 illustrates this process. For each bone i, our method samples three points $\mathbf{z}$ at a uniform interval along each bone. It then emits rays from each sampled point $\mathbf{z}$ in directions orthogonal to the bone equiangularly (we use 12 rays) in order to find the nearest intersections with the body mesh. Given intersections, it calculates the centroid and the average distance between each intersection and the centroid. It finally calculates the mesh-surrounding radius of bone i as an average of the average distance at each sampled point $\mathbf{z}$.

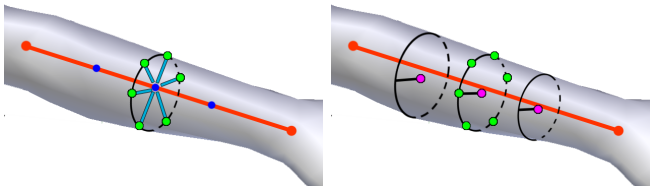


Figure 5: Calculation of surrounding-mesh radius. Left: We sample points (blue) along each bone (red) and emit rays to find mesh intersections (green). Right: We then calculate the surrounding-mesh radius as an average of averaged distances from centroids (magenta) to intersections.

B. Automatic Super Deformation

Our basic process of automatic super deformation is to shrink the target limbs and enlarge the target head so that the output model will have body proportion similar to that of the reference model, while referring to the body-shape parameters. For this, our method first modifies skeletons based on the body-shape parameters and then deforms the meshes of the target model using linear blend skinning.

1) Skeleton deformation: The target skeleton is deformed by shortening each bone recursively from the root joint. Bones belonging to the same bone group undergo similar shortening. Specifically, for each bone i in bone group G_b, the same shortening coefficient α_b ($\alpha_b < 1$) is applied:

$$\mathbf{p}'_{child(i)} = \mathbf{p}_{parent(i)} + \alpha_b \left(\mathbf{p}_{child(i)} - \mathbf{p}_{parent(i)}\right). \quad (1)$$

How to determine $\{\alpha_b\}$ is explained in Section V-B3. Also, the position of each joint j belonging to the child set of joint $child(i)$ is recursively updated:

$$\mathbf{p}'_j = \mathbf{p}_j + \Delta\mathbf{p}_{child(i)}, \quad (2)$$

$$\Delta\mathbf{p}_{child(i)} = \mathbf{p}'_{child(i)} - \mathbf{p}_{child(i)}. \quad (3)$$

As a special case, only the head bone is enlarged; by enlarging a head, limbs become relatively shorter than the head, and thus we do not have to shorten the limbs excessively. Specifically, the head bone length is updated using an enlarging coefficient α_{head} ($\alpha_{head} > 1$) as follows:

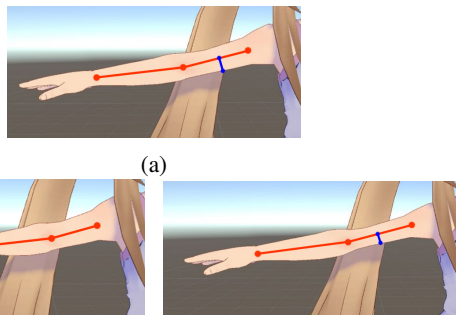
$$l'_{head} = \alpha_{head}\, l_{head}, \quad (4)$$

where l_{head} and l'_{head} are the head bone lengths before and after enlargement. In our experiments, we use $\alpha_{head} = 1.5$.

2) Mesh deformation: We explain mesh deformation driven by bone shortening. The vertex position after deformation is calculated according to the change in joint positions:

$$\mathbf{v}'_k = \mathbf{v}_k + \sum_{i=1}^{M} w_{ki}(\lambda_{ki}\Delta\mathbf{p}_{parent(i)} + \mu_{ki}\Delta\mathbf{p}_{child(i)}) \quad (5)$$

where $\mathbf{v}_k$ and $\mathbf{v}'_k$ are the positions of vertex k before and after the transformation, and w_{ki} is the weight of bone i for vertex k. $\Delta\mathbf{p}_{parent(i)}$ and $\Delta\mathbf{p}_{child(i)}$ are differences of joint positions. λ_{ki} and μ_{ki} are the weights of joints $parent(i)$ and $child(i)$, and calculated as follows.



(a)

(b) (c)

Figure 6: (a) Original mesh as well as results of (b) bone shrinkage and (c) mesh thickness adjustment.

$$\lambda_{ki} = 1 - \frac{\|\mathbf{proj}_i(\mathbf{v}_k) - \mathbf{p}_{parent(i)}\|}{\|\mathbf{p}_{parent(i)} - \mathbf{p}_{child(i)}\|}, \quad (6)$$

$$\mu_{ki} = 1 - \frac{\|\mathbf{proj}_i(\mathbf{v}_k) - \mathbf{p}_{child(i)}\|}{\|\mathbf{p}_{parent(i)} - \mathbf{p}_{child(i)}\|} \quad (7)$$

where $\mathbf{proj}_i(\mathbf{v}_k)$ is the foot of a perpendicular draw from vertex position $\mathbf{v}_k$ on the line segment of bone i. These two equations indicate that the closer vertex k is to a joint, the greater the influence of the joint is.

We adjust not only the length of each bone but also the thickness of the mesh around the bone. For each bone group G_b, the vertex position of the mesh around bone $i \in G_b$ is changed as follows:

$$\mathbf{v}'_k = \mathbf{v}_k + \sum_{i \in G_b} w_{ki}(\mathbf{proj}_i(\mathbf{v}_k) - \mathbf{v}_k)(1 - \beta_b) \quad (8)$$

where β_b is the thickness coefficient assigned for bone group G_b. How to determine $\{\beta_b\}$ is described in Section V-B3. Figures 6b and 6c show the results of applying bone shortening and mesh thickness adjustment, respectively.

3) Automatic coefficient calculation: The coefficients $\{\alpha_b\}$ and $\{\beta_b\}$ are calculated based on the body-parameters (L^{ref}, R^{ref}) and (L^{tar}, R^{tar}) as follows:

$$g = \frac{l'^{tar}_{head}}{l^{ref}_{head}} = \frac{\alpha_{head}\, l^{tar}_{head}}{l^{ref}_{head}}, \quad (9)$$

$$\alpha_b = 1 + \eta_b \left(g\, \frac{l^{ref}_i}{l^{tar}_i} - 1\right),\ i \in G_b, \quad (10)$$

$$\beta_b = 1 + \phi_b \left(g\, \frac{r^{ref}_i}{r^{tar}_i} - 1\right),\ i \in G_b. \quad (11)$$

where g is the ratio of the enlarged target head length (Section V-B1) to the reference head length. η_b and ϕ_b are coefficients for suppressing deformation; shortening body parts such as shoulder and hip yields an unnatural body shape because such parts cannot be well approximated as

a cylinder. Therefore, to suppress deformation of such parts, we set $\eta_{shoulder} = \eta_{hip} = 0.2$ and $\phi_{shoulder} = \phi_{hip} = 0$. For the rest parts, we set them to 1. For fingers and toes, just shortening them makes the target parts similar to the reference ones, and thus we just set $\alpha_{finger} = \alpha_{foot} = 0.5$ and $\beta_{finger} = \beta_{foot} = 1$. The user can also adjust the coefficients, although we use the default values without manual adjustments in our experiments.

VI. Face Infantilization

Here we try to make the target face younger like a child. Comparing faces of normal and SD models (see the examples in Figure 1), we notice that the eye positions of the SD model are lower than those of the normal model while the chin of the SD model is more rounded. Our method accomplishes both by applying a simple geometric transformation to the head mesh. Our approach is inspired by the biological theory by D'Arcy Thompson [2], which claims that a wide variety of body shapes of related species (e.g., crustacean, fishes or primate heads) can be well approximated by deforming one shape to another. Indeed, there is an example movie [13] demonstrating infantilization of character's face based on his theory, but the specific function for the transformation is not disclosed. Note that D'Arcy Thompson's theory does not mention a specific choice of transformation for each shape, and thus we have to design deformation functions by ourselves.

We design our own deformation functions as follows. In the bounding box of the head mesh, we set a normalized coordinate system where the upper vertical direction indicates $+y$ direction and $y \in [-0.5, 0.5]$. We consider the following two deformation functions:

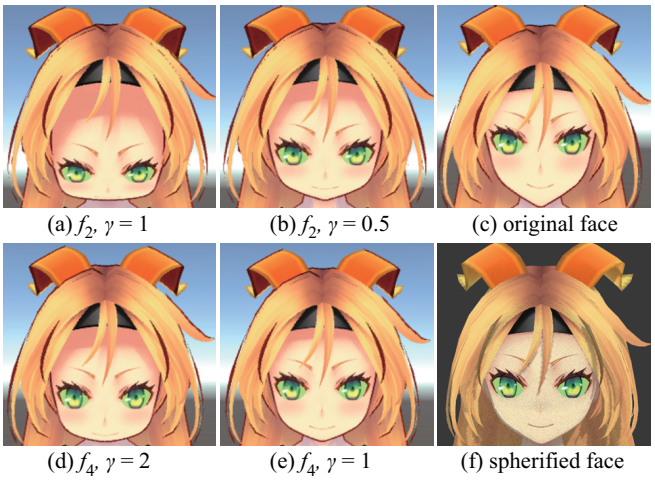
$$f_2(y) = \gamma\,(y^2 - 0.5^2) + y, \tag{12}$$

$$f_4(y) = \gamma\,(y^4 - 0.5^4) + y, \tag{13}$$

where γ is a coefficient. Both two functions are designed so that they squash the face only in the vertical direction, and always satisfy $f_2(-0.5) = f_4(-0.5) = -0.5$ and $f_2(0.5) = f_4(0.5) = 0.5$ for any γ. Figure 7 shows a comparison. While both functions $f_2(y)$ and $f_4(y)$ can lower the eye positions and make chins round, $f_2(y)$ has too strong effects (Figure 7(d) and (e)). The sphere approximation (Figure 7(f)), as done in [10], has only limited effects. We thus use function $f_4(y)$ with manually selected γ for each model in our experiments.

VII. Detailed adjustment

After the process described so far, we have a SD version of the target model. The user can further modify the target model manually. For example, we shorten long hair meshes (Figure 8(b) and (f)) manually using the modeling software 3ds Max. In our experiments, we do not apply any other manual editing unless otherwise noted.



(a) f_2, $\gamma = 1$ (b) f_2, $\gamma = 0.5$ (c) original face

(d) f_4, $\gamma = 2$ (e) f_4, $\gamma = 1$ (f) spherified face

Figure 7: Comparison of face transformations. (a)(b) Function f_2 lowers eyes and rounds chins too much, while (f) sphere approximation has limited effects. We use function f_4 in our results.

VIII. Results

We implemented our prototype system on Blender and Unity3D using Python and C# as programming languages. We use the SD model shown in Figure 1 as a reference model in our experiments. The computation time both for calculating body-shape parameters and automatic deformation is less than ten seconds. A single execution of the face transformation takes less than a second. When the user repeats the face transformation until satisfied, it typically completes within ten seconds. Figure 8 summarizes our results. We can see the generated SD models look visually pleasing with younger faces and stubby limbs.

Compared to the method by Shen et al. [10], their method provides larger degrees of freedom to the user with much more parameters than ours (i.e., only a single parameter in our face infantilization), which causes more trial-and-errors. Every time the user specifies any parameter(s), their optimization step takes time ranging from a few seconds to more than ten seconds. Longer computation time in the trial-and-error loop is not desirable because the user usually adds further manual editing as automatic process is often insufficient; for example, Figure 7(f) demonstrates sphere approximation similar to [10] does not work well.

IX. Conclusion and Future Work

In this paper, we have proposed a technique for semi-automatically generating a super-deformed (SD) model from a normal character model. We designed our method so that it takes a reference SD model as input to learn body-shape parameters for avoiding repetitive parameter tuning. Our method automatically calculates the skeleton and weight maps for a target model, while accounting for consistent

Figure 8: Our results. Using the SD model shown in Figure 1 as a reference model, our method generates SD models (b), (d), (f) and (h) from normal models (a), (c), (e) and (g).

deformation between body and cloth meshes. The target model is then automatically super-deformed by transferring the body-shape parameters from the reference model. The target face becomes younger through a simple quartic transformation.

As future work, we would like to improve the appearance of generated SD characters. For example, SD legs should have less muscles with children's skeleton structure, and geometric as well as texture details should be suppressed, while discriminative features of the character should be exaggerated.

As a limitation, our method currently assumes that each of reference and target models has a hole-free skin mesh and a separated garment mesh so that we can measure body-shape parameters and calculate weight maps, which is often not the case with character models publicly available on the Internet. Considering different methods for calculating weight maps is also an interesting future direction. For example, the voxel-based method [14] is an option, but weight transfer (Section IV-B) would be required for separated skin and garment meshes. Cage-based methods [15] would be beneficial to handle separate meshes, if an automatic method [16] for generating cages is available.

References

[1] "Super deformed," https://en.wikipedia.org/wiki/Super_deformed, [Online; accessed 23-December-2016].

[2] D. W. Thompson, *On growth and form*. Cambridge University Press, 1945.

[3] N. Magnenat-Thalmann, R. Laperrière, and D. Thalmann, "Joint-dependent local deformations for hand animation and object grasping," in *Proceedings on Graphics Interface '88*, 1988, pp. 26–33.

[4] L. Kavan, S. Collins, J. Žára, and C. O'Sullivan, "Geometric skinning with approximate dual quaternion blending," *ACM Trans. Graph.*, vol. 27, no. 4, pp. 105:1–105:23, Nov. 2008.

[5] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3D faces," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '99, 1999, pp. 187–194.

[6] B. Allen, B. Curless, and Z. Popović, "The space of human body shapes: reconstruction and parameterization from range scans," in *ACM transactions on graphics (TOG)*, vol. 22, no. 3. ACM, 2003, pp. 587–594.

[7] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "SCAPE: Shape completion and animation of people," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 408–416, Jul. 2005.

[8] B. Reinert, T. Ritschel, and H.-P. Seidel, "Homunculus warping: Conveying importance using self-intersection-free non-homogeneous mesh deformation," *Comput. Graph. Forum*, vol. 31, no. 7pt2, pp. 2165–2171, Sep. 2012.

[9] S. Saito, Z.-Y. Zhou, and L. Kavan, "Computational bodybuilding: Anatomically-based modeling of human bodies," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 41:1–41:12, Jul. 2015.

[10] L.-T. Shen, S.-J. Luo, C.-K. Huang, and B.-Y. Chen, "SD models: Super-deformed character models," *Comput. Graph. Forum*, vol. 31, no. 7pt1, pp. 2067–2075, Sep. 2012.

[11] I. Baran and J. Popović, "Automatic rigging and animation of 3D characters," in *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3. ACM, 2007, p. 72.

[12] R. Lopez and C. Poirel, "Raycast based auto-rigging method for humanoid meshes," in *ACM SIGGRAPH 2013 Posters*. ACM, 2013, p. 11.

[13] "D'Arcy Thompson's transformation," https://www.youtube.com/watch?v=017Fa8Gs4bI, [Online; accessed 23-December-2016].

[14] O. Dionne and M. de Lasa, "Geodesic voxel binding for production character meshes," in *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2013, pp. 173–180.

[15] A. Jacobson, I. Baran, J. Popović, and O. Sorkine, "Bounded biharmonic weights for real-time deformation," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 78:1–78:8, Jul. 2011.

[16] B. H. Le and Z. Deng, "Interactive cage generation for mesh deformation," in *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 2017, pp. 3:1–3:9.