SCIENCE CHINA
Information Sciences

• MOOP •

September 2018, Vol. 61 094101:1–094101:3
https://doi.org/10.1007/s11432-018-9470-4

Novel metaballs-driven approach with dynamic constraints for character articulation

Junxuan BAI[1], Junjun PAN[1,2*], Yuhan YANG[1] & Hong QIN[3]

[1]*State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing* 100191*, China;*
[2]*Beihang University Qingdao Research Institute, Qingdao* 266100*, China;*
[3]*Department of Computer Science, Stony Brook University, New York* 11794-4400*, USA*

Received 23 February 2018/Accepted 7 May 2018/Published online 18 July 2018

Citation Bai J X, Pan J J, Yang Y H, et al. Novel metaballs-driven approach with dynamic constraints for character articulation. Sci China Inf Sci, 2018, 61(9): 094101, https://doi.org/10.1007/s11432-018-9470-4

Skinning techniques are essential for character articulation in 3D computer animation. Currently, skeleton-based methods are widely used in the animation industry for its simplicity and efficiency, especially in linear blend skinning (LBS) [1] and dual quaternion skinning (DQS) [2]. However, owing to the lack of the inside volumetric representation, they suffer from joint collapse, candy-wrapper, and bulging problems.

As shown in Figure 1, we herein propose a volumetric skinning method using a set of balls (metaballs). Metaballs method is a modeling technique based on implicit surfaces. It can represent continuous, blobby-like surfaces. This method has been applied to virtual surgery simulations [3] for modeling human organs. Here, we employ a displacement representation and Laplacian coordinates to represent the surface. We also present additional joint balls to improve the deformation quality around the joints. To introduce dynamic effects such as jiggling, we couple our metaballs model with position-based dynamics (PBD) [4] and action lines [5]. The action lines are employed to represent simplified muscles and tendons that can constrain the motions of the balls. The experiments show that our metaballs model can create realistic deformations for real-time applications.

To demonstrate our method, we first describe our metaballs method for primary deformations. Subsequently, we illustrate the techniques for the enhanced animation effects, such as the handling for surface overlapping, muscle bulging effect, and secondary deformations.

Primary deformation using metaballs. The mesh deformation is composed of two components: deformation by normal balls and deformation by joint balls. The final position of the vertex $\hat{\boldsymbol{v}}_i$ is formulated as

$$\hat{\boldsymbol{v}}_i = (1 - w_i^{\alpha})\, \boldsymbol{v}_i^{\lambda} + w_i^{\alpha} \boldsymbol{v}_i^{\eta}, \tag{1}$$

where $\boldsymbol{v}_i^{\lambda}$, $\boldsymbol{v}_i^{\eta}$ represent the deformed positions by the normal ball and joint balls, respectively. w_i^{α} controls the influence of the different types of balls.

At each frame, the skeleton is modified according to the motion data; subsequently, the balls are transformed along with the skeleton pose. After the steps above, the PBD exerts its influence on these balls to constrain the motions. Finally, the mesh vertices are deformed following the displacements inferred from the metaballs.

Deformation by normal balls. We follow the initialization process in [3] to set up our balls. At each frame, the translations and rotations of the balls are first computed. Subsequently, the vertices are deformed according to the transformation of the balls. The new position of the normal ball $\hat{\boldsymbol{c}}_i^{\lambda}$ is computed as

$$\hat{\boldsymbol{c}}_i^{\lambda} = \sum_{j=1}^{m} w_{ij}^{\beta} \boldsymbol{T}_j \boldsymbol{c}_i^{\lambda}, \tag{2}$$

* Corresponding author (email: pan_junjun@buaa.edu.cn)

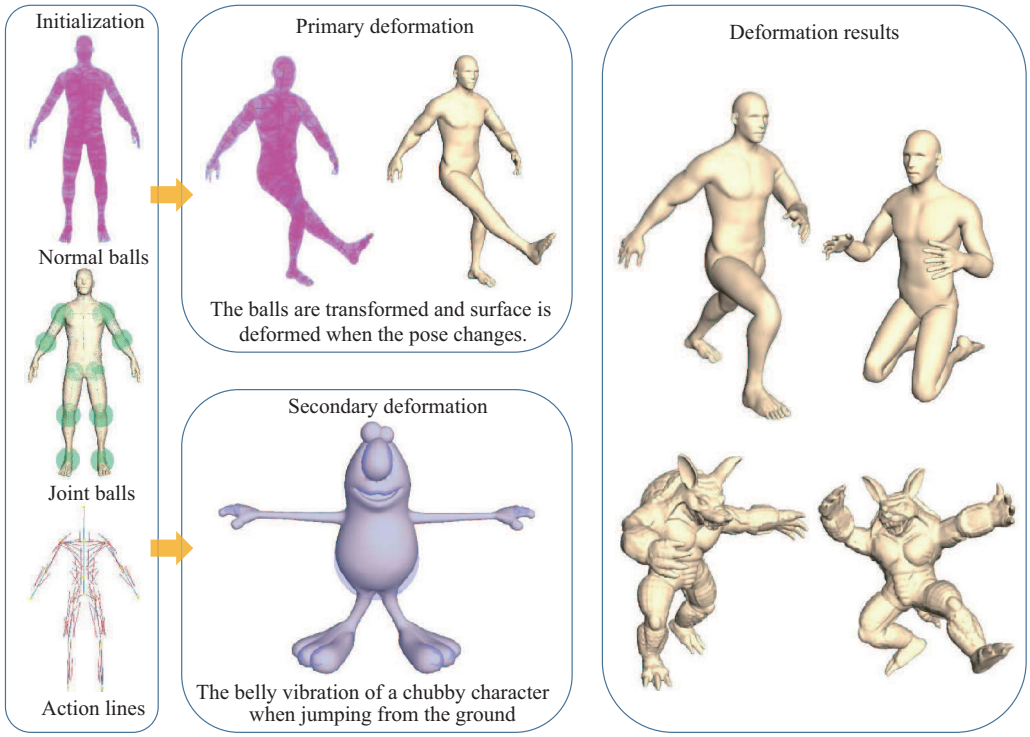

Figure 1 (Color online) The framework of our method.

where m is the number of bones, w_{ij}^{β} controls the impact of different bones; $\boldsymbol{T}_j$ denotes the transformation matrix of the bone; $\boldsymbol{c}_i^{\lambda}$ represents the initial position of the ball's center. The rotation for each ball is handled using the spherical linear interpolation (Slerp).

The positions of the normal balls are refined through the elasticity of the springs using PBD. We apply stretch constraints on these springs after the computation of the balls' transformations. The constraint function is written as

$$C_{\text{stretch}}(\hat{\boldsymbol{c}}_i^{\lambda}, \hat{\boldsymbol{c}}_j^{\lambda}) = \left|\hat{\boldsymbol{c}}_i^{\lambda} - \hat{\boldsymbol{c}}_j^{\lambda}\right| - d, \tag{3}$$

where d denotes the initial distance between the pair of balls. More computation details can be found in [4].

Our skinning method is based on a rotation-invariant displacement representation, which is inspired by the Laplacian coordinates [6], to preserve the local details of the surface. Initially, we store the initial displacement $\boldsymbol{d}_{ij}^{\lambda} = \boldsymbol{v}_i - \boldsymbol{c}_j^{\lambda}$ for each vertex, where $\boldsymbol{v}_i$ denotes the initial position of the mesh vertex.

After the transformations of the balls are calculated, the deformed vertex position by the normal balls $\boldsymbol{v}_i^{\lambda}$ is computed using

$$\boldsymbol{v}_i^{\lambda} = \sum_{j=1}^{n} w_{ij}^{\gamma} \left(\hat{\boldsymbol{c}}_j^{\lambda} + \hat{\boldsymbol{R}}_j^{\lambda} \boldsymbol{d}_{ij}^{\lambda}\right), \tag{4}$$

where n is the number of all normal balls; w_{ij}^{γ} controls the impact of different normal balls; $\hat{\boldsymbol{R}}_j^{\lambda}$ represents the self-rotation matrix of the normal ball.

Deformation by joint balls. To improve the deformation quality around the joints, we introduced additional joint balls. The radius of the joint ball r_i^{η} is set manually. Each ball is attached to at least one bone, and the influence is equally assigned to each attached bone. The mesh vertices attached to different bones have different transformations.

Initially, the displacements of the joint balls $\boldsymbol{d}_{ij}^{\eta} = \boldsymbol{v}_i - \boldsymbol{c}_j^{\eta}$ are stored. During the animation, the transformed position of the joint ball $\hat{\boldsymbol{c}}_i^{\eta}$ is computed. It is the same with the skeleton joint at all times. Meanwhile, the rotation calculations are executed. The deformed position $\boldsymbol{v}_i^{\eta}$ by the joint ball is computed using

$$\boldsymbol{v}_i^{\eta} = \hat{\boldsymbol{c}}_j^{\eta} + \hat{\boldsymbol{R}}_j^{\eta} \boldsymbol{d}_{ij}^{\eta}, \tag{5}$$

where $\hat{\boldsymbol{c}}_j^{\eta}$, $\hat{\boldsymbol{R}}_j^{\eta}$ represent the current position of joint ball and the rotation matrix of joint ball, respectively.

Blending process. According to (1), the deformed position $\boldsymbol{v}_i^{\lambda}$ and $\boldsymbol{v}_i^{\eta}$ are blended and combined into a new position $\hat{\boldsymbol{v}}_i$. Because the mesh boundary of the joint regions may appear slightly rough, we applied Laplacian smoothing on these vertices. To recover the original shape around the joints, we also employed local Laplacian coordinates as a shape-preserving filter on the vertices

in this region. The detailed formulations can be found in the supplementary materials.

Surface overlapping. We use a hierarchical scheme to detect the overlapping vertices. Each ball is assigned to a bone, and the balls on the same bone are combined into a ball set. At each frame, we employ these ball sets to detect collision, and subsequently check if the vertices inside the colliding balls are located in the colliding balls on the other bone. If the vertices are located in the range of these balls, they are projected onto an interior plane orthogonally.

Muscle bulging. We design a straightforward method to simulate the muscle bulging effect. Before the animation, the bulging regions are defined and the balls in the regions are selected. The size of the selected ball is changed in relation to the angle between the connected bones at runtime.

Secondary deformation. To achieve this effect, we need to create the jiggling regions and a time window W_t before the animation. The regions define the influenced vertices and balls, and W_t affects the strength of the jiggling effects.

The vertices in the jiggling regions are deformed using secondary deformations. The balls are divided into two groups: balls in the regions (denoted as S_{in}), and balls outside the regions (denoted as S_{out}). At each frame, the balls in S_{out} are computed when the skeleton pose changes, and then the positions of the balls in S_{in} are modified through the elasticity of springs in the PBD. The jiggling behavior is controlled by the stiffness of the spring. The character body will reach a larger extent if we employed softer springs.

Results and comparisons. We compared our skinning method with LBS and DQS, which are regarded as the benchmarks of real-time applications. All the experiments were run on a desktop PC with Intel(R) Core(TM) i5-6400 CPU (2.71 GHz), 8-GB RAM, NVIDIA GeForce GT 720. Three models are employed: the Armadillo (34594 vertices and 564 balls), the chubby model (8620 vertices and 467 balls), and the human (24461 vertices and 512 balls). The experimental details are provided in the supplementary materials.

The experimental results show that the metaballs-driven skinning method can reduce the candy-wrapper effect and joint collapse artifacts, and can maintain a better rigidity. Moreover, our method can produce some enhanced effects, such as secondary deformations, handling for surface overlapping, and muscle bulging. The time comparison demonstrates that our method is suitable for real-time applications.

Limitations. Compared with the traditional skeleton-based skinning methods, our method requires more weight computations during initialization. In addition, more presetting are required, such as defining the radius of the joint ball, and defining the projection planes for surface overlapping.

Conclusion and discussion. We presented a novel volumetric skinning method based on metaballs. We employed a displacement representation and Laplacian coordinates to express the character. The character deformation relies on the transformations of the balls under the surface. The deformation quality around the joints was improved by our additional joint balls. Our method resolves the problems of traditional skeleton-based skinning techniques and provides enhanced animation effects. We believe that our method can be applied in virtual try-on or real-time games.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 61532002, 61672149), National Key R&D Program of China (Grant No. 2017YFF0106407), and Applied Basic Research Program of Qingdao (Grant No. 161013xx). We thank Jialing SHUI for the textures and art guidance, and Xinyu WANG for several experiments.

Supporting information Videos and other supplemental documents. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Magnenat-Thalmann N, Laperrière R, Thalmann D. Joint-dependent local deformations for hand animation and object grasping. In: Proceedings of Graphics Interface'88, Edmonton, 1988. 26–33
2 Kavan L, Collins S, Žára J, et al. Geometric skinning with approximate dual quaternion blending. ACM Trans Graph, 2008, 27: 1–23
3 Pan J J, Zhao C K, Zhao X, et al. Metaballs-based physical modeling and deformation of organs for virtual surgery. Vis Comput, 2015, 31: 947–957
4 Müller M, Heidelberger B, Hennix M, et al. Position based dynamics. J Vis Commun Image Represent, 2007, 18: 109–118
5 Delp S L, Anderson F C, Arnold A S, et al. OpenSim: open-source software to create and analyze dynamic simulations of movement. IEEE Trans Biomed Eng, 2007, 54: 1940–1950
6 Lipman Y, Sorkine O, Cohen-Or D, et al. Differential coordinates for interactive mesh editing. In: Proceedings of Shape Modeling Applications, Genova, 2004. 181–190