Received March 25, 2018, accepted May 3, 2018, date of publication May 15, 2018, date of current version June 5, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2837013

Variational Mannequin Approximation Using Spheres and Capsules

NANNAN WU[1], DONGLIANG ZHANG[2], ZHIGANG DENG[3,4], (Senior Member, IEEE), AND XIAOGANG JIN[1], (Member, IEEE)
[1]State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou 310058, China
[2]International Design Institute, Zhejiang University, Hangzhou 310058, China
[3]Virtual Reality and Interactive Technique Institute, East China Jiaotong University, Nanchang 330013, China
[4]Computer Science Department, University of Houston, Houston, TX 77004, USA

Corresponding author: Xiaogang Jin (jin@cad.zju.edu.cn)

The work of D. Zhang was supported by the National Natural Science Foundation of China under Grant 61472355. The work of X. Jin was supported in part by the National Natural Science Foundation of China under Grant 61732015, in part by the National Key Research and Development Program of China under Grant 2017YFB1002600, and in part by the Key Research and Development Program of Zhejiang Province under Grant 2018C01090.

ABSTRACT We present a method to approximate a mannequin represented as a triangle/quadrangle mesh using spheres and capsules. We formulate this 3-D approximation problem as a 2-D one, which increases its efficiency by one or even two orders of magnitude. Our algorithm first extracts cross sections from a given mannequin, approximates each cross section with circles using a variant of iterative Lloyd clustering, and finally generates capsules based on the circle sets. Compared with previous methods, our method is much faster and is able to get a tighter result with a given number of primitives. Our method can be directly applied to cloth-body collision handling in real-time cloth animation, and the simulation results by our approach are visually plausible.

INDEX TERMS Approximation, real-time cloth animation, spheres and capsules, variational.

I. INTRODUCTION

Approximating a 3D object using a set of simple geometric primitives is of great importance in shape analysis, collision detection, and shadowing. Geometric primitives include spheres, ellipsoids, axis-aligned bounding boxes (AABBs), oriented bounding boxes (OBBs), discrete oriented polytopes (k-DOPs), and so on. Among all the primitives, sphere is the simplest one; thus, it has been widely employed in computer graphics community. A capsule consisting of two spheres is also simple and efficient in collision detection. Because of this, both of two well-known, open-source physics-based engines, Bullet and PhysX, require a character to be represented by spheres and/or capsules in cloth simulation in order to provide real-time performance. However, such a representation is created manually in these engines. Therefore, there is a clear need to develop an automatic method to approximate a given mannequin using spheres and capsules as primitives.

Recently, Thiery *et al.* [1] proposed the Sphere-Meshes representation which is the linear interpolation of spheres along edges and triangles. While this method can handle a wide variety of models, the approximation error is large when the number of primitives is small, e.g., fewer than 300. Moreover, it cannot guarantee a symmetric approximation for a symmetric input mannequin.

In this paper, based on the key insight that a human body can be divided into six parts: left/right arm, left/right leg, head and torso, and that the shape of each part is similar to a capsule, we propose a novel variational approximation algorithm to represent a mannequin by spheres and capsules. By employing the semantic information of the input mannequin, our method can generate a tighter and more symmetric result while keeping the number of primitives small. Based on the anthropometric characteristics of the human body, we turn such a 3D approximation problem into a 2D one, which significantly increases the efficiency of the approximation (i.e., two orders of magnitude faster than existing methods in our experiments).

Figure 1 shows the framework of our approach. It consists of four steps: (i) Extract cross sections for each body part based on anthropometric characteristics of the human body. Each cross section is represented as a polygon which serves as a feature polygon of the input mannequin (see Figure 2). (ii) Approximate a feature polygon with circles using a variant of iterative Lloyd clustering. (iii) Generate capsules

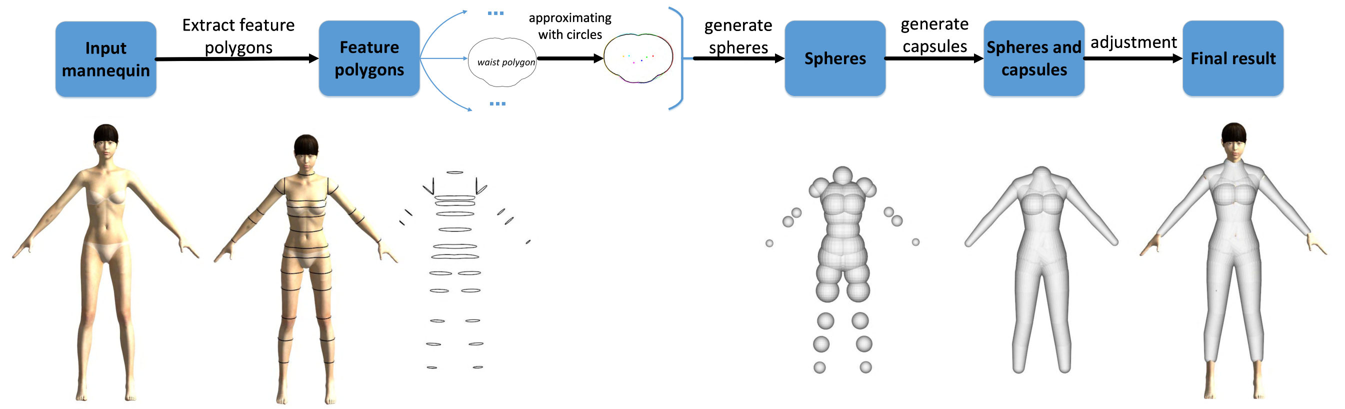


FIGURE 1. The pipeline of our method: Given a mannequin, we first extract cross sections (feature polygons) from each body part. Then, we use a variational scheme to approximate these polygonal sections with circles. After that, we generate spheres, spheres/capsules in turn for the mannequin, and obtain the final result by adjusting their centers and radii.

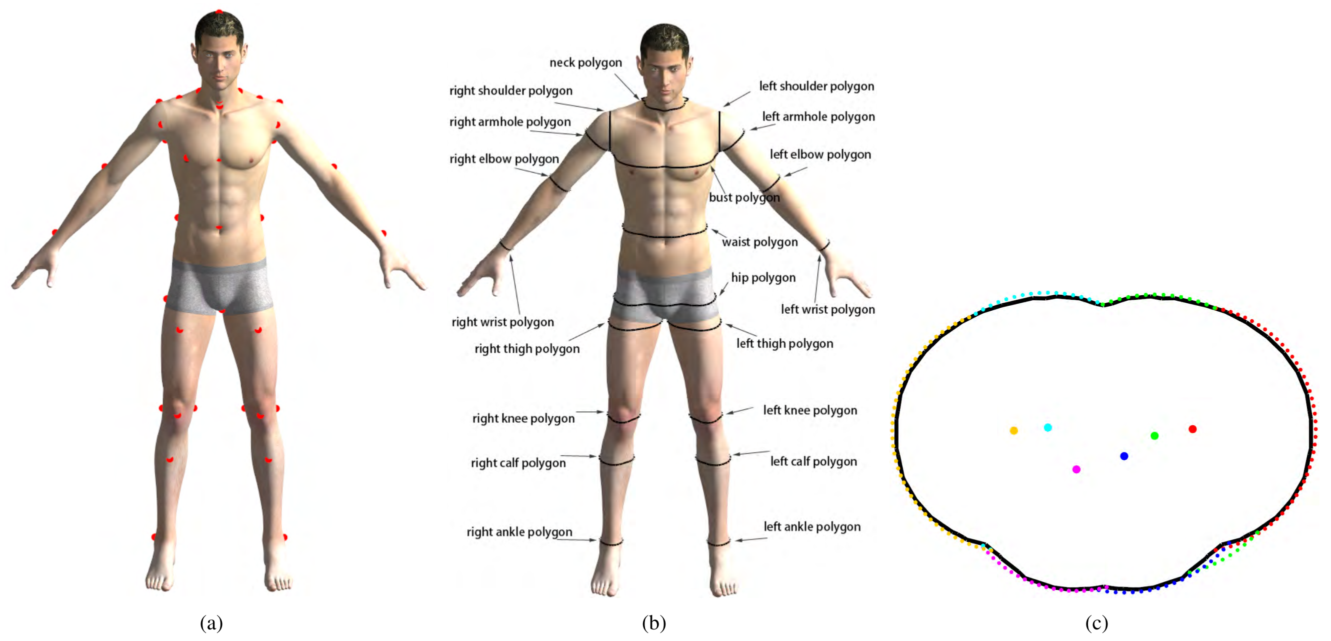


FIGURE 2. Feature polygons extraction and approximation: (a) some landmark points of a mannequin used in our algorithm, (b) twenty feature polygons of a mannequin, (c) and the approximation result of the waist polygon using six circles. The colored dotted lines outside the waist polygon (black) are the boundaries of the circles, and the colored dots inside the waist polygon are the centers of the circles.

based on the approximation results from (ii); (iv) Adjust the centers and/or radii of some spheres to completely cover the mannequin. Step (i) and step (ii) are the key steps to transform the 3D approximation problem to a 2D one. In a nutshell, our algorithm of approximating a feature polygon is a kind of variational approximation methods, where the primitives are circles, and the error metric is the outside area (i.e., the area inside the circle set but outside the polygon).

For a given number of primitives, our method is both faster and tighter than previous methods, and the result by our approach can be directly used in physics-based engines such as Bullet and PhysX for cloth-body collision handling. Our approach has two major technical contributions:

- a novel method to turn the 3D approximation problem into a 2D one by extracting feature polygons from a mannequin; and
- a variational method to approximate a feature polygon with circles with a user-specified error threshold, using an error metric that measures the area inside the circle set but outside the polygon.

II. RELATED WORK

A. SHAPE APPROXIMATION

A large amount of research works have been conducted on shape approximation. These works can be categorized by the approximation primitives: spheres [2]–[4], ellipsoids [5]–[7], axis-aligned bounding boxes (AABBs) [8], oriented bounding boxes (OBBs) [9], [10], discrete oriented polytopes (k-DOPs) [11], planar elements [12], swept sphere volumes [13], surfels [14], and a mixed set of the above primitives [1], [15].

Variational approximation methods [3], [6], [12], [15], [16] play an important role in shape approximation. A typical variational approximation framework consists of three stages: (i) a *partitioning* stage usually segments an object into some regions using a variant of Lloyd clustering; (ii) a *fitting* stage fits geometric primitives to each region, and (iii) a *teleportation* stage is triggered when the algorithm gets stuck in a local minimum, for example, it deletes a certain primitive and then inserts a new one accordingly.

The difference among variational methods mainly lies in the geometric primitives used to approximate each region and the error metric to guide the partitioning and fitting stages. The original variational shape approximation [12] uses planar elements as geometric primitives, and further work [15] extends to spheres, circular cylinders, and rolling ball blending surfaces. Both of them use an integrated surface distance as the error metric. The variational sphere set approximation [3] uses spheres as primitives and the outside volume as the error metric, and it is able to obtain a tighter approximation.

Thiery *et al.* [1] proposed an approximation method guided by spherical quadric error metrics. The resulting Sphere-Meshes representation is a linear interpolation of spheres along edges and triangles, which can be viewed as an approximation using spheres, truncated cones, and prisms as primitives. Technically, our approximation result can be viewed as a Sphere-Mesh without prisms, which is designed for real-time virtual try-on. Recently, Calderon and Boubekeur [17] proposed a shape approximation method based on an asymmetric morphological closing. Their output is a bounding polygon mesh. It is noteworthy that our method can also bound the input mannequin.

Other works also use the ellipsoid [6] or general quadric [16] as geometric primitives. Khoury *et al.* [18] use capsules to approximate humanoid robots. However, they only use one capsule to approximate each body part, and the capsule is required to be composed of two spheres of the same size. Their goal is to minimize the volume of the capsule while the capsule covers the body part. As a result, the accuracy of their method is limited. Since the body parts are segmented in advance, their method is essentially the fitting stage of variational approximation.

B. ACCELERATION STRUCTURES FOR COLLISION DETECTION

Collision detection is one of the most time-consuming parts in many computer graphic applications. To accelerate collision detection, researchers have developed various acceleration data structures including bounding-volume hierarchies, distance fields, and other spatial partitioning methods [19]. For a deforming character, its spatial data structures have to be updated when a character deforms. Although distance field-based approaches can get a high performance for a static object, it is time consuming to compute the dynamic distance field for a dynamic object. Different from them, our method can be updated through skinning in a fast way, and therefore it could deal with a dynamic mannequin for real time virtual try-on applications.

III. SPHERE AND CAPSULE SET APPROXIMATION

In this section, we describe our algorithm in detail. After presenting the feature polygon extraction algorithm in Section III-A, we approximate the obtained feature polygons with circles in Section III-B. Based on the 2D approximation, we can generate a 3D approximation in Sections III-C and III-D. After that, we describe the refinement part in Section III-E.

A. EXTRACTING FEATURE POLYGONS

A feature polygon is the boundary of a cross section of a body part. To extract feature polygons, we first calculate landmark points of the input mannequin including the points nearby crotch, wrists, and elbows. This can be done automatically using the method in [20]. Then, we extract cross sections that go through the landmark points and are perpendicular to the corresponding bone's direction. This part takes about two-thirds of the total computation time. Figure 2(a) shows typical feature points of a mannequin, Figure 2(b) shows the corresponding feature polygons, and Figure 2(c) shows the approximation result of the *waist polygon*.

B. APPROXIMATING FEATURE POLYGONS

A *feature polygon* can be approximated by a set of circles using a variational approximation method (Figure 2(c)). We first define an error metric called *Circle Outside Area* (COA), which represents the area inside the circle set but outside the polygon (Figure 3), similar to SOV in [3]. Then, we minimize this error using a variant of iterative Lloyd clustering. All feature polygons are approximated independently. Note that with COA as the error metric, the approximation becomes a *smallest circle problem* when the number of circles equals 1, which can be solved efficiently [21].

1) CALCULATING COA

The COA error of a circle C_i is defined as:

$$Err(P, C_i) = A(E, C_i), \tag{1}$$

where E is the edge set of the polygon P, and A is the area inside the circle C_i but outside the polygon P. The total outside area of the entire circle set thus is defined as:

$$Err(P, C) = \sum_{i=1}^{n_c} Err(P, C_i) = \sum_{i=1}^{n_c} A(E, C_i), \tag{2}$$

where n_c is the number of circles, and C_i is the i-th circle.

We calculate the COA of C_i edge by edge. If an edge e_k is entirely or partially inside C_i, then there exists an area

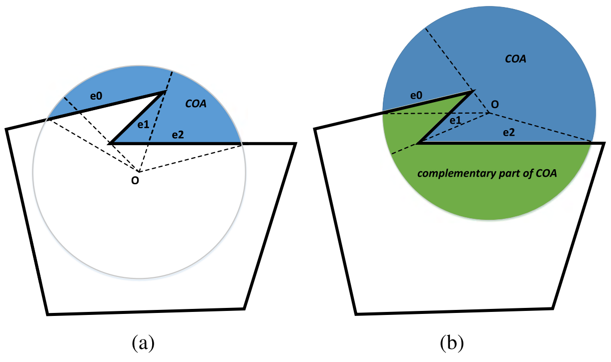


FIGURE 3. The computation of COA. (a) The circle center is inside the polygon, the COEAs of e_0 and e_2 are positive, the COEA of e_1 is negative. (b) The circle center is outside the polygon. Calculating Eq. 4 will result in the green part.

between the circumference of C_i and e_k, we call it *Circle Outside Edge Area* (COEA) and denote it by $A(e_k, C_i)$. Then the total COA is calculated by adding or subtracting these COEAs over all the edges of P.

a: COMPUTING COEA

Let p_0 and p_1 be the two vertices of the edge e_k. If both $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ are inside C_i, then the COEA is defined as:

$$A(e_k, C_i) = A_{sec}(e_k, C_i) - A_{tri}(e_k, \boldsymbol{o}_i), \tag{3}$$

where $A_{sec}(e_k, C_i)$ is the area of the sector formed by the edge e_k and C_i's center $\boldsymbol{o}_i$; $A_{tri}(e_k, \boldsymbol{o}_i)$ is the area of the triangle formed by the C_i's center $\boldsymbol{o}_i$, $\boldsymbol{p}_0$, and $\boldsymbol{p}_1$.

If any of $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ is outside C_i, then the part of e_k inside C_i (denoted as $\bar{e_k}$) is used to calculate COEA using Eq. 3. If $\bar{e_k}$ does not exist, which means e_k is completely outside C_i, then the COEA of the edge e_k is 0.

b: ACCUMULATING COEA INTO COA

The sign of each COEA is determined by two factors: whether the circle center is inside or outside the polygon, and whether the circle center is behind or in front of the edge (according to the edge's outward normal direction).

i) THE CIRCLE CENTER INSIDE THE POLYGON

COEA is positive if the circle center is behind the edge, and negative otherwise. Then, the COA of circle C_i can be computed as:

$$A_{in}(E, C_i) = \sum_{k=1}^{n_e} SGN(\boldsymbol{n}_k \cdot (\boldsymbol{p}_0^k - \boldsymbol{o}_i)) \cdot A(e_k, C_i), \tag{4}$$

where n_e is the number of the edges of the polygon, e_k is the k-th edge, $\boldsymbol{n}_k$ is the outward normal of e_k, $\boldsymbol{p}_0^k$ is one of the endpoints of e_k, and $\boldsymbol{o}_i$ is the center of the circle C_i. SGN is a sign function that returns 1 if the argument is positive, -1 otherwise.

ii) THE CIRCLE CENTER OUTSIDE THE POLYGON

We first calculate the complementary area of COA (Figure 3(b)) using Eq. 4, which is negative in this case. Adding the total circle area to the complementary area results in the following COA:

$$A_{out}(E, C_i) = \pi r^2 + A_{in}(E, C_i). \tag{5}$$

As shown in Figure 3, the blue part is the COA, the green part is the complementary area of COA, which is calculated by Eq. 4.

We have also experimented the *Hausdorff* Distance as the error metric. For a circle C, the Hausdorff Distance error is defined between the circle part outside the polygon P and the polygon part inside the circle C. However, we found that it costed more time and the result was not as good as the circle outside area.

2) MINIMIZING COA

We use a variant of iterative Lloyd clustering algorithm [22] to minimize Eq. 2. It consists of four steps: *initialization*, *partitioning*, *fitting*, and *teleportation*. With our strategy, we calculate several local minimums and then choose the best one as the final result.

Initialization. A polygon is discretized into two types of points: *boundary points* and *interior points*. We use the vertices of the polygon as boundary points and generate interior points using a scan conversion algorithm [23].

We choose n_c points that are evenly distributed on the longest diagonal of the polygon as the circle centers based on the observation that the resulting circle centers are always nearby the longest diagonal of the polygon P after the approximation process. We set all the circle radii as 0.

Partitioning. When a point p is assigned to a circle C, C's radius will be enlarged to include p, and thus the COA of C will increase accordingly. We assign p to C_i that has the minimal increased COA.

To decide the order of points to be assigned, we calculate the distances from point p to each circle center, and the minimal distance is called the *distance value* of p. The points with smaller distance values will be assigned first. We have also experimented the *flood fill* (stack-based) algorithm [3] to order the points, but the result is not as good as the method we employ.

Fitting. After the partitioning, each circle is assigned to a set of points. Then, the center and radius of each circle are updated to best fit the points assigned to it. We minimize Eq. 1 using the *multidimensional direction set method* [24]:

$$\arg\min_{o,r} Err(P, C_i), \tag{6}$$

where o is the center of circle C, and the radius r must be large enough to include all the assigned points.

Teleportation. Like other variational approximation algorithms, our method cannot guarantee the global optimal result. When the algorithm gets stuck in a local minimum, a teleportation step is triggered to find another local minimum. If several consecutive local minimums do not improve the result sufficiently, the approximating process terminates and the best local minimum is accepted as the final result.

If several consecutive partitioning and fitting iterations did not improve the result significantly, then the algorithm would get stuck in a local minimum (possibly a global minimum) and teleportation is triggered. The circle C_o with the maximum overlap ratio is marked and will be deleted, and the circle C_e with the maximum COA is split into two circles C_e^0 and C_e^1. The overlap ratio of a circle C is defined as the ratio of COA shared with at least one of other circles to the total area of C. The initial centers of C_e^0 and C_e^1 are randomly chosen from interior points of C_i, and another partitioning and fitting iteration is then conducted.

C. GENERATING SPHERES AND CAPSULES

The capsule here means a volume formed by two spheres, and the radii of the spheres are not necessarily the same. *Generating spheres:* For each circle, we generate a sphere with the same radius and center. *Generating capsules:* We divide the mannequin into five parts: left arm, right arm, left leg, right leg, and torso, and each part will be processed independently. For each body part, we iteratively choose two adjacent feature polygons denoted as L_1 and L_2, then for each sphere C_i from L_1 and C_j from L_2, a capsule formed by C_i and C_j is generated. After all the capsules are generated, we eliminate those capsules that are completely covered by other capsules.

For most applications, the head can be approximated with one sphere, which is a minimally-enclosing sphere problem. The same method can be used for hands and feet.

D. ADJUSTING CENTERS/RADII OF SPHERES

The sphere/capsule set calculated from Section 3.3 usually cannot completely cover the mannequin. Therefore, we need to adjust the centers and/or radii of some spheres. We first calculate a point that is outside the sphere/capsule set, and assign it to the nearest sphere or capsule.

If an outside point p is assigned to a sphere S, then we increase the radius of S by $D/2$, and move the center towards p by $D/2$, where D is the distance from p to S (see Figure 4).

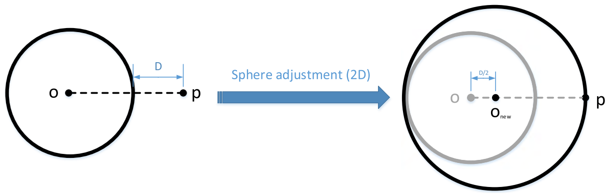


FIGURE 4. Adjust a sphere to include a new point (2D).

If an outside point p is assigned to capsule J, then we enlarge the radii of the two spheres of J by $D/2$, and move the centers of the two spheres toward the direction that points to p and is perpendicular to the axis of J by $D/2$, where D is the distance from p to J (see Figure 5).

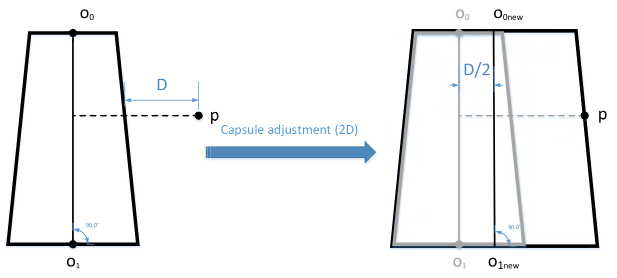


FIGURE 5. Adjust a capsule to include a new point (2D). The trapezoid is the cross section of the circular truncated cone part of the capsule.

As shown in Figures 4 and 5, the ordinal sphere/capsule is completely inside the new sphere/capsule. This means that the points inside the original sphere/capsule will also be inside the new sphere/capsule.

E. REFINEMENTS

We can further refine the approximation as follows.

1) SPLITTING THE HIP POLYGON INTO THREE PARTS

We split a hip polygon into three parts: left buttock polyline, right buttock polyline, and center butt polyline (see Figure 6). Each part will be approximated independently. Although sample points for the partitioning step are generated from the new polygon, we still calculate the COA of a circle using the original feature polygon.

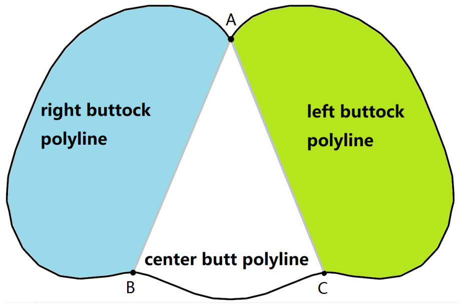


FIGURE 6. Top view of the splitting result of the hip polygon. A, B and C are local valley points.

2) SPLITTING THE BUST POLYGON INTO THREE PARTS

As for a female mannequin, two breasts can be represented by two spheres. To do so, we split the bust polygon into three parts: left breast polyline, right breast polyline, and the rest (see Figure 7), and use one circle to approximate each part.

To decide the gender of the mannequin, we calculate the ratio of the height of the breast and the width of the chest. If the ratio is larger than a threshold (we use 0.04 in our experiments), we consider the mannequin as female, and split the bust polygon as described above.

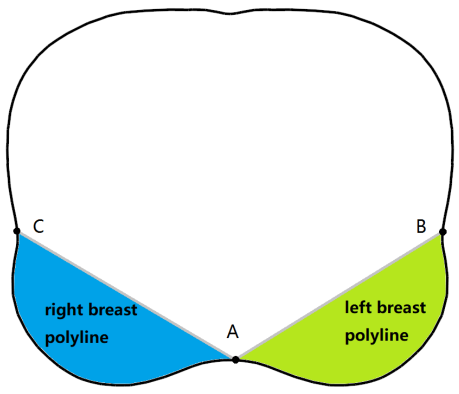


FIGURE 7. Top view of the splitting result of the bust polygon. A, B and C are local valley points.

IV. APPLICATIONS AND RESULTS

A. APPLICATIONS

Like the sphere set from [3], our sphere/capsule set can also be used for proximity query, generating soft shadows, etc. It is obvious that out sphere/capsule set is smoother than the sphere set [3], and thus can be directly used for cloth-body collision detection in cloth simulation.

While animating other approximations [12], [16] might be complex, it is quite straightforward for our sphere + capsule set. We first calculate the mean value coordinates [3], [25] of each sphere center in the rest pose. During the animation, the centers of spheres are updated using a linear combination of the positions of the deformed vertices of the mannequin's skin.

In [26], animated mesh sequences are approximated by animated sphere-meshes, which are meshes indexing the animated sphere set. However, this method may not work when the animated mesh sequence does not exist,, e.g. real-time virtual try-on applications.

B. RESULTS

We implemented our approach in C++ and reported its performances on an off-the-shelf computer with Intel Core i7-7700 CPU 4.20GHz and 16GB memory.

To the best of our knowledge, there is no existing method to approximate a solid object using both spheres and capsules as primitives. The most relevant one is Sphere-Meshes [1] which is a linear interpolation of spheres along edges and triangles. Therefore, we compared our method with it, based on visual and geometric errors, performance, and applications in cloth simulation. Note that we do not approximate hands, feet and the head of the mannequin, since these parts typically are not used in virtual try-on applications.

1) VISUAL AND GEOMETRIC ERRORS

Geometric errors are measured by the relative outside volume E_r, i.e., the ratio of the volume outside (inside) the original model but inside (outside) the sphere/capsule set, and E_M, i.e., mean distance from the original model to the approximation. Similar to [1], E_M is expressed as the percentage of the bounding box of the input mannequin. The original mannequin M_o consists of 45k vertices and 64k faces. We deleted the hands, feet and head of the original mannequin to generate a new mannequin M_n. Both our method and Sphere-Meshes use M_o as input.

We take a capsule as 3 primitives, 2 spheres and 1 truncated cone. For Sphere-Meshes [1], each interpolated edge corresponds to a capsule and thus 3 primitives, and each interpolated triangle corresponds to three capsules and one triangle for the outer crusts of the interpolation and thus 8 primitives. Figure 8 compares our approximation with the Sphere-Meshes representation using three different numbers of primitives: $n_p = 32$, $n_p = 64$, and $n_p = 128$. From the figure we can see that, the computation time of our method is much less than that of the Sphere-Meshes method, and our result is more symmetric and tighter. The main reason is that by utilizing the semantic information of the input mannequin, our method can put spheres at the most appropriate positions.

Figure 9 illustrates E_r as a function of n_p. We can see that our results are more accurate with a given number of primitives. Although the curves of our method are close to those of Sphere-Meshes after n_p is larger than 150, both curves are flatten out. This means that, to obtain a certain error, the number of primitives required for Sphere-Meshes is much larger than our method.

There are two ways to improve our result: (1) Increase the number of circles for each feature polygon, and (2) add more feature polygons (although this will result in more circles). Generally, when a feature polygon L_{new} is newly added, more circles are needed for the feature polygons adjacent to L_{new} and for L_{new} itself.

2) PERFORMANCE

Table 1 shows the computation time t with respect to the number of primitives n_p. We can find that our method is more than two orders of magnitude faster than Sphere-Meshes. Sphere-Meshes is a bottom-up decimation algorithm, which means the smaller the desired number of primitives, the more time it costs. As shown in column one, to obtain an approximation of 22 primitives, our method is more than 200 times faster than Sphere-Meshes. As a result, our method is more suitable for real-time applications than the Sphere-Meshes method.

Figure 10 shows the time ratio t as a function of the number of faces n_f of the mannequin. Each mannequin is approximated with 64 primitives. We can see that the relative performance of our method increases even further when the number of the faces of the mannequin increases. Therefore, our method is more appropriate to approximate high resolution mannequins.

3) CLOTH SIMULATION

a: DRAPING SIMULATION

Figure 11 shows a draping simulation result using our sphere/capsule set approximation method. The female mannequin is approximated by 64 primitives. The dress consists of 8k

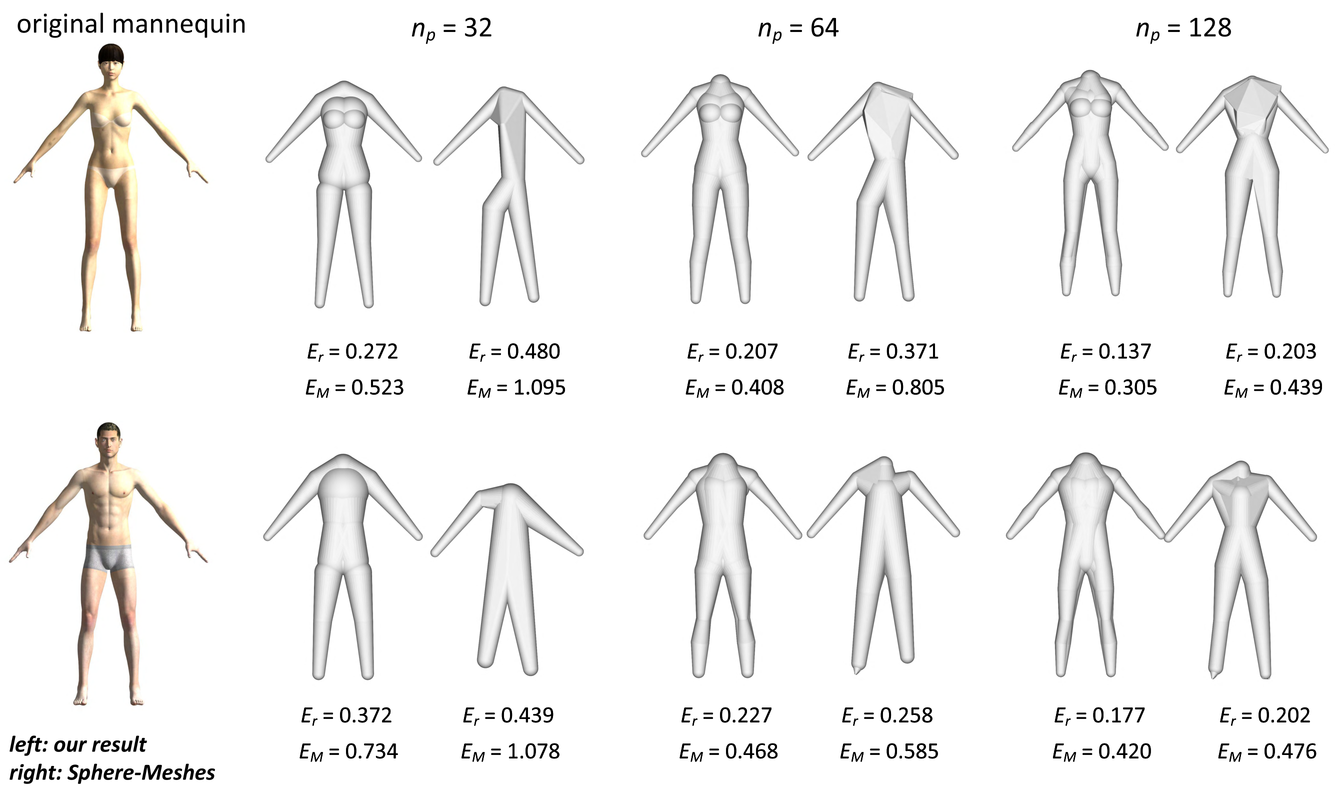

original mannequin
n_p = 32
n_p = 64
n_p = 128
E_r = 0.272 E_M = 0.523
E_r = 0.480 E_M = 1.095
E_r = 0.207 E_M = 0.408
E_r = 0.371 E_M = 0.805
E_r = 0.137 E_M = 0.305
E_r = 0.203 E_M = 0.439
E_r = 0.372 E_M = 0.734
E_r = 0.439 E_M = 1.078
E_r = 0.227 E_M = 0.468
E_r = 0.258 E_M = 0.585
E_r = 0.177 E_M = 0.420
E_r = 0.202 E_M = 0.476
left: our result
right: Sphere-Meshes

FIGURE 8. Sphere/capsule set comparison with Sphere-Meshes.

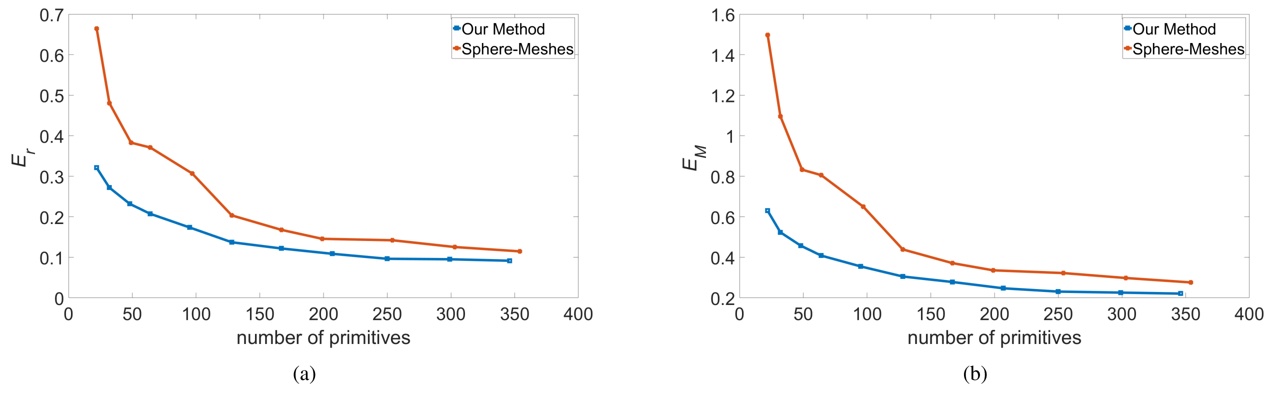


FIGURE 9. Plottings of E_r vs. n_p (a), and E_M vs. n_p (b).

TABLE 1. Computation time vs. number of primitives (our method / Sphere-Meshes).

time(s) \ n_p / method	22/22	32/32	48/49	64/64	95/97	128/128	167/167	207/199	250/254	299/303	346/354
ours	0.005	0.044	0.035	0.048	0.110	0.106	0.163	0.153	0.147	0.143	0.146
Sphere-Meshes	4.327	4.354	4.315	4.965	4.303	4.450	4.232	4.108	4.124	4.126	4.099

vertices and 14k faces. In this example, we only take the sphere/capsule set (not rendered) as a collision handling data structure.

b: REAL-TIME CLOTH ANIMATION

Collision handling is the most time-consuming process in cloth simulation. Many prior works [19], [27] have been

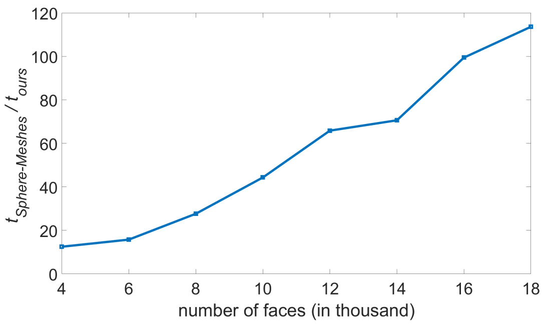


FIGURE 10. The time ratio between Sphere-Meshes and our method with respect to the number of faces of the mannequin. $t_{Sphere-Meshes}$ and t_{ours} are the computational times for Sphere-Meshes and our method, respectively.

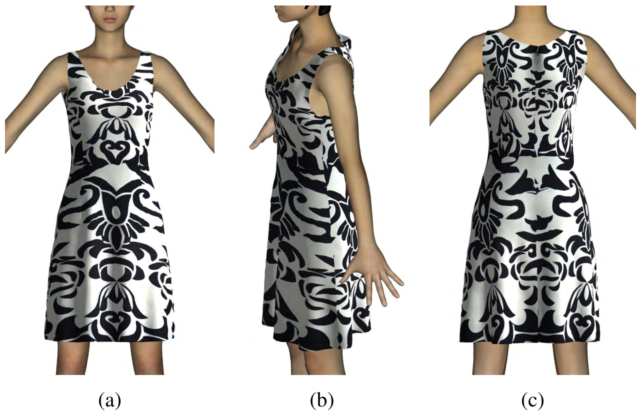


FIGURE 11. Draping simulation using our approximation algorithm. (a) front view. (b) left view. (c) back view.

done in this area. However, all of them have to re-compute a spatial data structure (e.g., bounding-volume hierarchies, distance fields, etc.) for acceleration when the character is deformed. The re-computing process will consume a lot of time, which is usually unacceptable for real-time cloth animation. Fortunately, the time cost of updating our sphere/capsule set is negligible in our approach: we first calculate the mean value coordinate [28] of each sphere center in the character's rest pose, and then update them using a linear combination of the deformed positions of the mannequin skin vertices.

We used position based dynamics [29], which is a popular method to simulate fluid [30], cloth, etc., to perform physical simulation. The mannequin was rigged by a dual quaternion linear blending skinning [31]–[33]. The motion data was extracted from CMU motion capture data [34] or captured by Kinect v2.0. We used the method in [35] to define the skinning weights of a given character's mesh with respect to its skeleton. Unlike previous data-driven methods [32], [36]–[39], our algorithm can perform in real time without any pre-computation except for generating sphere/capsule set approximation. In our experiments, the dress consists of 8k vertices and 14k faces, the mannequin consists of 7k vertices and 13k faces, and the frame per second (fps) can be up to 55. In the future, we plan to add hair to the mannequin [40], [41] to obtain more realistic results. Please see the supplemental demo video for animation details.

V. CONCLUSION AND FUTURE WORK

In this paper, we present a novel algorithm to approximate a mannequin using a mixed set of spheres and capsules. Our method turns a 3D approximation problem into a 2D one, which increases the performance by two orders of magnitude. With the same number of primitives, our method is faster and more accurate than previous methods. When the user uses physics-based engines such as Bullet and PhysX engines, our result can be directly used for real-time cloth animation because we employ a simplified cloth-body collision handling algorithm. Our experimental results demonstrate the effectiveness and efficiency of our approach. Although our method is tailored to generate a sphere/ capsule set approximation for a mannequin, we believe that it can also be used to approximate other articulated characters when their feature polygons can be calculated.

Our current approach has the following limitations: First, as the sphere/ capsule set is larger than the mannequin, our algorithm may bring errors. Second, for a mannequin, the regions around shoulders are difficult to be approximated accurately because of their special shapes. In the future, we plan to solve this issue by adding certain special feature polygons perpendicular to each shoulder's direction.

ACKNOWLEDGMENT

Nannan Wu would like to thank Liang Guo from Zhejiang University for his great help in debuging the code. Nannan Wu also wants to thank Chunji Yin and Han Liu, both from Zhejiang University, for their valuable discussion about the implementation of the Sphere-Meshes method [1].

REFERENCES

[1] J.-M. Thiery, É. Guy, and T. Boubekeur, "Sphere-meshes: Shape approximation using spherical quadric error metrics," *ACM Trans. Graph.*, vol. 32, no. 6, 2013, Art. no. 178.
[2] G. Bradshaw and C. O'Sullivan, "Adaptive medial-axis approximation for sphere-tree construction," *ACM Trans. Graph.*, vol. 23, no. 1, pp. 1–26, Jan. 2004.
[3] R. Wang *et al.*, "Variational sphere set approximation for solid objects," *Vis. Comput.*, vol. 22, nos. 9–11, pp. 612–621, 2006.
[4] S. Stolpner, P. Kry, and K. Siddiqi, "Medial spheres for shape approximation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 6, pp. 1234–1240, Jun. 2012.
[5] S. Bischoff and L. Kobbelt, "Ellipsoid decomposition of 3D-models," in *Proc. 1st Int. Symp. 3D Data Process. Vis. Transmiss.*, 2002, pp. 480–488.
[6] L. Lu, Y.-K. Choi, W. Wang, and M.-S. Kim, "Variational 3D shape segmentation for bounding volume computation, " *Comput. Graph. Forum*, vol. 26, no. 3, pp. 329–338, 2007.
[7] Y. Meng, P. Y. Mok, and X. Jin, "Interactive virtual try-on clothing design systems," *Comput.-Aided Des.*, vol. 42, no. 4, pp. 310–321, 2010.
[8] G. van den Bergen, "Efficient collision detection of complex deformable models using AABB trees," *J. Graph. Tools*, vol. 2, no. 4, pp. 1–13, 1997.
[9] S. Gottschalk, M. C. Lin, and D. Manocha, "OBBTree: A hierarchical structure for rapid interference detection," in *Proc. 23rd Annu. Conf. Comput. Graph. Interact. Techn.*, 1996, pp. 171–180.
[10] S. Krishnan, A. Pattekar, and M. C. Lin, "Spherical shell: A higher order bounding volume for fast proximity queries," in *Proc. 3rd Workshop Algorithmic Found. Robot. Robot., Algorithmic Perspective*, 1997, pp. 177–190.

[11] J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan, "Efficient collision detection using bounding volume hierarchies of k-DOPs," *IEEE Trans. Vis. Comput. Graphics*, vol. 4, no. 1, pp. 21–36, Jan. 1998.

[12] D. Cohen-Steiner, P. Alliez, and M. Desbrun, "Variational shape approximation," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 905–914, 2004.

[13] M. Bae, J. Kim, and Y. J. Kim, "User-guided volumetric approximation using swept sphere volumes for physically based animation," *Comput. Animation Virtual Worlds*, vol. 23, nos. 3–4, pp. 385–394, 2012.

[14] Y. Fan, Y. Huang, K. Cai, F. Yan, and J. Peng, "Surfel set simplification with optimized feature preservation," *IEEE Access*, vol. 4, pp. 10258–10269, 2016.

[15] J. W. L. Kobbelt, "Structure recovery via hybrid variational surface approximation," *Comput. Graph. Forum*, vol. 24, no. 3, pp. 277–284, 2005.

[16] D.-M. Yan, Y. Liu, and W. Wang, "Quadric surface extraction by variational shape approximation," in *Proc. 4th Int. Conf. Geometric Modeling Process. (GMP)*. Berlin, Germany: Springer, 2006, pp. 73–86.

[17] S. Calderon and T. Boubekeur, "Bounding proxies for shape approximation," *ACM Trans. Graph.*, vol. 36, no. 4, Jul. 2017, Art. no. 57.

[18] A. E. Khoury, F. Lamiraux, and M. Taïx, "Optimal motion planning for humanoid robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2013, pp. 3136–3141.

[19] M. Teschner *et al.*, "Collision detection for deformable objects," *Comput. Graph. Forum*, vol. 24, no. 1, pp. 61–81, 2005.

[20] C. C. L. Wang, "Parameterization and parametric design of mannequins," *Comput.-Aided Des.*, vol. 37, no. 1, pp. 83–98, 2005.

[21] E. Welzl, "Smallest enclosing disks (balls and ellipsoids)," *New Results and New Trends in Computer Science*. Berlin, Germany: Springer, 1991.

[22] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, Mar. 2006.

[23] A. Kaufman, "An algorithm for 3d scan-conversion of polygons," in *Proc. Eurograph. Conf.*, 1987, pp. 197–208.

[24] W. H. Press, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed. Cambridge, U.K.: Cambridge Univ. Press, 2007.

[25] J.-M. Thiery, J. Tierny, and T. Boubekeur, "Jacobians and hessians of mean value coordinates for closed triangular meshes," *Vis. Comput.*, vol. 30, no. 9, pp. 981–995, 2014.

[26] J.-M. Thiery, É. Guy, T. Boubekeur, and E. Eisemann, "Animated mesh approximation with sphere-meshes," *ACM Trans. Graph.*, vol. 35, no. 3, 2016, Art. no. 30.

[27] J. Mezger, S. Kimmerle, and O. Etzmuß, "Hierarchical techniques in collision detection for cloth animation," *J. WSCG*, vol. 11, nos. 1–3, pp. 1–8, 2003.

[28] T. Ju, S. Schaefer, and J. Warren, "Mean value coordinates for closed triangular meshes," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 561–566, Jul. 2005.

[29] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff, "Position based dynamics," *J. Vis. Commun. Image Represent.*, vol. 18, no. 2, pp. 109–118, 2007.

[30] X. Shao, E. Liao, and F. Zhang, "Improving SPH fluid simulation using position based dynamics," *IEEE Access*, vol. 5, pp. 13901–13908, 2017.

[31] L. Kavan, S. Collins, and J. žára, and C. O'Sullivan, "Geometric skinning with approximate dual quaternion blending," *ACM Trans. Graph.*, vol. 27, no. 4, 2008, Art. no. 105.

[32] W. Xu, N. Umentani, Q. Chao, J. Mao, X. Jin, and X. Tong, "Sensitivity-optimized rigging for example-based real-time clothing synthesis," *ACM Trans. Graph.*, vol. 33, no. 4, Jul. 2014, Art. no. 107.

[33] J. Baek, H. Jeon, G. Kim, and S. Han, "Visualizing quaternion multiplication," *IEEE Access*, vol. 5, pp. 8948–8955, 2017.

[34] CMU. (2003). *CMU Graphics Lab Motion Capture Database*. Accessed: Jan. 1, 2017. [Online]. Available: http://mocap.cs.cmu.edu

[35] I. Baran and J. Popović, "Automatic rigging and animation of 3D characters," *ACM Trans. Graph.*, vol. 26, no. 3, Jul. 2007, Art. no. 72.

[36] F. Hahn *et al.*, "Subspace clothing simulation using adaptive bases," *ACM Trans. Graph.*, vol. 33, no. 4, 2014, Art. no. 105.

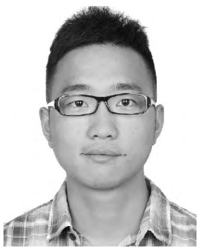
[37] D. Kim, W. Koh, R. Narain, K. Fatahalian, A. Treuille, and J. F. O'Brien, "Near-exhaustive precomputation of secondary cloth effects," *ACM Trans. Graph.*, vol. 32, no. 4, 2013, Art. no. 87.

[38] Y. A. Sekhavat, "Privacy preserving cloth try-on using mobile augmented reality," *IEEE Trans. Multimedia*, vol. 19, no. 5, pp. 1041–1049, May 2017.

[39] H. Wang, F. Hecht, R. Ramamoorthi, and J. F. O'Brien, "Example-based wrinkle synthesis for clothing animation," *ACM Trans. Graph.*, vol. 29, no. 4, Jul. 2010, Art. no. 107.

[40] Y. Bao and Y. Qi, "A survey of image-based techniques for hair modeling," *IEEE Access*, vol. 6, pp. 18670–18684, 2018.

[41] Y. Bao and Y. Qi, "An image-based hair modeling and dynamic simulation method," *IEEE Access*, vol. 5, pp. 12533–12544, 2017.



NANNAN WU received the B.Sc. degree in computer science from the Nanjing University of Science and Technology, China, in 2013. He is currently pursuing the Ph.D. degree with the State Key Laboratory of CAD&CG, Zhejiang University. His main research interests include cloth simulation, virtual try on, and computer animation.

DONGLIANG ZHANG received the B.Sc. and M.Sc. degrees from Zhejiang University and the Ph.D. degree from The Hong Kong University of Science and Technology. He is currently a Professor with the International Design Institute, Zhejiang University, China. His research interests include computer-aided design, computer graphics, interaction design, and robotics.

ZHIGANG DENG (SM'11) received the B.S. degree in mathematics from Xiamen University, China, the M.S. degree in computer science from Peking University, China, and the Ph.D. degree in computer science from the Department of Computer Science, University of Southern California, in 2006. He is currently a Full Professor of computer science with the University of Houston (UH) and the Founding Director of the UH Computer Graphics and the Interactive Media Laboratory. His research interests include computer graphics, computer animation, virtual human modeling and animation, and human–computer interaction. He is a Senior Member of the ACM. He was a recipient of a number of awards, including the ACM ICMI Ten Year Technical Impact Award, the UH Teaching Excellence Award, the Google Faculty Research Award, the UHCS Faculty Academic Excellence Award, and the NSFC Overseas and Hong Kong/Macau Young Scholars Collaborative Research Award. He was the CASA 2014 Conference General Co-Chair and the SCA 2015 Conference General Co-Chair. He currently serves as an Associate Editor for several journals, including *Computer Graphics Forum* and *Computer Animation and Virtual Worlds* journal.

XIAOGANG JIN (M'14) received the B.Sc. degree in computer science and the M.Sc. and Ph.D. degrees in applied mathematics from Zhejiang University, China, in 1989, 1992, and 1995, respectively. He is currently a Professor with the State Key Laboratory of CAD&CG, Zhejiang University. His current research interests include traffic simulation, insect swarm simulation, physically based animation, cloth animation, special effects simulation, implicit surface computing, non-photorealistic rendering, computer generated marbling, and digital geometry processing. He is a member of the ACM. He received the ACM Recognition of Service Award in 2015 and the Best Paper Award from CASA 2017.