DOI: 10.1111/j.1467-8659.2010.01805.x

Pacific Graphics 2010
P. Alliez, K. Bala, and K. Zhou
(Guest Editors)

Volume 29 (2010), Number 7

B-Mesh: A Modeling System for Base Meshes of 3D Articulated Shapes

Zhongping Ji [1†] Ligang Liu[2‡] Yigang Wang[1]

[1]Institute of Graphics and Image, Hangzhou Dianzi University, China
[2]Department of Mathematics, Zhejiang University, China

Abstract

This paper presents a novel modeling system, called B-Mesh, for generating base meshes of 3D articulated shapes. The user only needs to draw a one-dimensional skeleton and to specify key balls at the skeletal nodes. The system then automatically generates a quad dominant initial mesh. Further subdivision and evolution are performed to refine the initial mesh and generate a quad mesh which has good edge flow along the skeleton directions. The user can also modify and manipulate the shape by editing the skeleton and the key balls and can easily compose new shapes by cutting and pasting existing models in our system. The mesh models generated in our system greatly benefit the sculpting operators for sculpting modeling and skeleton-based animation.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Modeling Packages—Sculpting modeling, skeleton, base mesh

1. Introduction

Traditional modeling approaches make use of parametric patches, implicit surfaces, or subdivision surfaces and have been well integrated into 3D softwares. The user has to lay out the coarsest level patches and modify control points to generate and edit 3D shapes.

Recently, highly detailed shapes are becoming commonplace in video games and movies, in part due to 3D scanning technologies, and in part due to the digital sculpting softwares [Pix10, Aut10]. *Sculpting modeling* becomes a fairly popular approach for creating 3D character shapes with *scan quality* geometric detail in game and movie industry and has emerged to be a new trend to 3D modeling like ZBrush [Pix10]. Like multi-resolution modeling, the sculpting modeling considers a 3D modeling as a *coarse-to-fine* procedure. First, a coarse base mesh is generated to represent the rough structure of the shape. The user then imports the base mesh into some sculpting package, subdivides it into higher resolution, and adds geometric and texture details. The two phases can be iterative during the modeling process. See Figure 1 for an illustration of the modeling process.

Actually, the whole modeling process is complex, and usually needs a cooperation of using a group of packages. A common but not a fixed workflow is as follows: sketch three-view drawings, make a coarse polygonal mesh (3ds Max, Maya, etc.), import into the sculpting package to carve (ZBrush, Mudbox), make a UV map of the base mesh (UVLayout, Unfold3D, etc.), paint colors and textures

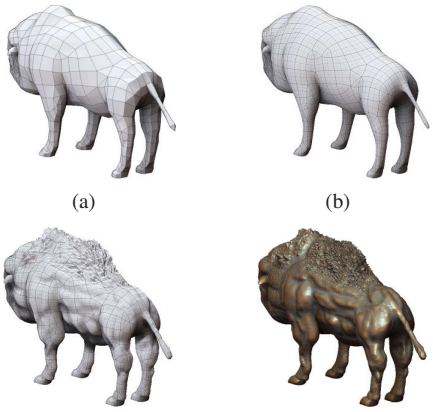


Figure 1: *The coarse-to-fine 3D modeling process in sculpting modeling. (a) A low resolution base mesh; (b) a high resolution of (a) by subdivision; (c) adding some geometric details in (b) using sculpting operators; (d) the final high detailed model after a few iterations.*

† jzp@hdu.edu.cn
‡ ligangliu@zju.edu.cn

(Bodypaint, Photoshop, etc.), and make some maps such as displacement, normal, ambient occlusion and so on (xNormal, ZBrush, Maya, etc.). The workflow is flexible and may vary with different purposes. These softwares are chosen by the designers according to their personal preferences.

Generating a base model is the first demand for 3D sculpting modeling. However, it is still not easy to build 3D base models from *scratch* for either inexperienced or professional users. Actually, 3D modeling is a time-consuming and skill-intensive procedure. The essential problem is that the space of possible geometric shape is extremely high dimensional. To form an aesthetic shape, the designer should modify the 3D positions of vertices patiently by clicking-and-dragging 2D positions on the image plane. The rich shape space, primitive interface and aesthetics make it nontrivial and challenging. Furthermore, the designer should be evenly balanced in technical and artistic ability.

In the sculpting packages, *quad meshes* are usually preferred over triangle meshes. The reasons are two folds. First, quad mesh with good *edge flow* can represent a shape in a more natural way, say, by aligning the shape features or curvature lines. The 3D modeling artists attach great importance to the edge flow especially for purpose of animation. Second, displacement adds local details at the vertices of mesh. Decomposing a surface into well-shaped quadrangles simplifies the construction of displacement map and texture atlas.

On the other hand, modeling and rigging have been completely separated in current modeling systems. A 3D model is created in some modeling software and then is passed later to the other animation software. In the animation software, the user has to rig the shape manually by placing the skeleton joints inside the shape. The rigging process should be carefully handled because the animation will be dependent on the skeleton position and is tedious in making animation [BP07].

Then, how to make a base model which is quad mesh with good edge flow and has an animation-friendly skeleton? Unfortunately, the current real-world 3D modeling workflow mentioned above and quick generation of base mesh suitable for 3D displacement-painting are widely *neglected* in the academic community.

As far as we know, the previous works have not yet considered all these requirements. i) The sketch-based modeling systems, including Teddy [IMT99], FiberMesh [NISA07] and 2D-to-3D Modeling [GIZ09] etc., are promising works in progress. However, they only produce triangular meshes. Recently, a sketch-based approach to design subdivision models is proposed [NKS09]. ii) Metaball, also known as blobby object or soft body [Bli82, NHK*85, WMW86], provides an intuitive form for 3D modeling, especially for blob-like shapes, like water drops. However, metaball is not suitable for articulated shape modeling because the triangle meshes generated from metaball system using marching-cube suffer from several weaknesses, including aliasing, low triangle quality, and large numbers of triangles. The resulting mesh is not suitable for animation and details sculpting. iii) ZBrush provides a modeling tool called ZSpheres. From a single sphere, the user can grow new ones, and then scale, move and rotate into an organic form. It is noteworthy that the user needs to control the orientation of each sphere carefully. Because the spheres are replaced by cubes in the adaptive skinning phase, the distortion and self-intersection problems often occur during modeling, which results in poor edge flow for the base mesh. Therefore, many 3D artists still use traditional packages to make animation-friendly base meshes.

Strongly motivated by these observations, we propose a simple modeling framework on creating quad-dominant base meshes for 3D articulated shapes so that they can be imported into sculpting packages to easily add rich details and can directly be animated in animation package.

Contributions We develop a simple and intuitive modeling system for generating base meshes of 3D articulated shapes. The generated base meshes are quad-dominant with good edge flow and are good for sculpting modeling and skeletal animation. We make three main contributions in this direction:

(1) We have developed an easy-to-use system for modeling base meshes of 3D articulated shapes. The user only needs to focus on specifying the skeleton of shape and the key balls at the skeletal nodes.

(2) The base meshes generated by our approach are quad-dominant meshes and have good edge flows along the skeleton bones. They are quite useful for sculpting geometric details and decorating textures in sculpting modeling systems.

(3) Our system provides a uniform framework for modeling, rigging, and animation of 3D articulated models.

2. Related work

An intensive survey on shape modeling is beyond the scope of this paper. Technically we draw inspirations from two most related areas: skeleton-based implicit surface modeling and sweeping modeling.

Implicit Surface Modeling Implicit modeling based on skeleton-like primitive was first introduced by Blinn [Bli82] and its variations were developed in [NHK*85, WMW86]. Free-form design with sketching interface based on implicit surface has been investigated [KHR02, ABGC05]. Tai et al. [TZF04] proposed a hybrid modeling method from sketched silhouettes based on convolution surfaces. Based on a hierarchical structure BlobTrees, a sketch-based modeling system, called ShapeShop, was proposed [SWSJ05]. The implicit surface representation is robust and compact for designing topologically complex objects. However, the resulting meshes are generally irregular which are hard to be used in sculpting geometric details.

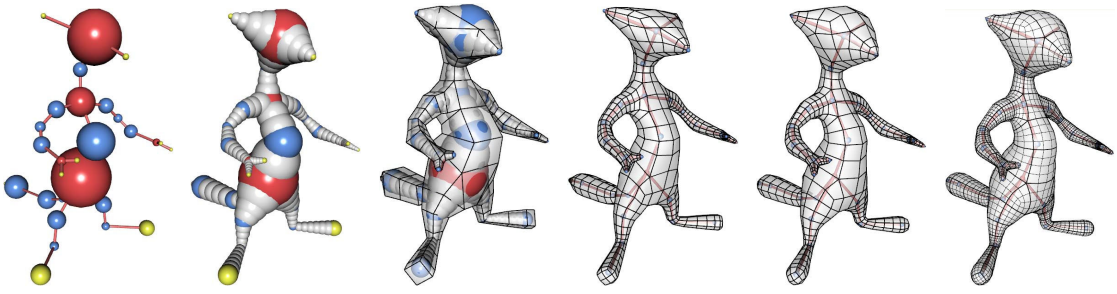


Figure 2: *Overview of our B-Mesh modeling approach. (a) Specifying the skeleton and key-balls at the nodes by users; (b) creating inbetween-balls (in gray) by interpolating the key-balls; (c) generating an initial mesh; (d) subdividing the mesh (c); (e) evolving the mesh (d); (f) obtaining the final mesh with more subdivision and evolution.*

Sweeping Modeling Most CAD commercial packages allow generation of volumes by extruding a surface along an axis. The one-to-one sweep volumes were defined by a source surface and a target surface [RSH04]. Besides the unstructured mesh generation methods [ZB04], a skeleton-based subdivision method was used in biomedical applications, such as a below-knee residual limb and bifurcation geometry in vascular flow simulation [VFLL05]. Based on ellipsoidal sweeping, an approach for deforming a virtual character's arms and legs was proposed in [HYKJ03].

3. Overview of B-Mesh modeling system

Our modeling system, called *B-Mesh*, aims to generate base meshes of articulated shapes for sculpting modeling and skeletal animation. Figure 2 shows the modeling process of our B-Mesh system. The user interfaces of B-Mesh are simple and intuitive: the user draws 1D skeleton of the shape on the screen and sets a set of balls at the skeletal nodes, see Figure2(a). Then the system automatically generates a shape whose profile matches the envelope of the balls based on iterative subdivisions and evolutions: First, more balls are generated along each skeletal edge by interpolating the corresponding key balls at skeletal nodes (see Figure2(b)). An initial mesh is then generated via sweeping and stitching operations to roughly approximate all the balls (see Figure2(c)). The system subdivides the mesh to obtain higher resolution mesh as shown in Figure2(d). As the subdivision surface results in shrinkage, an evolution process is performed to the subdivision mesh to better approximate the envelope of all the balls (see Figure2(e)). Then the system can perform more subdivision and evolution operations iteratively to obtain finer result as shown in Figure2(f). The number of iterations can be set by the user. At any phase of the modeling process, the user can edit the skeleton and the key balls and see the modified shape interactively.

4. B-Mesh modeling

Before we elaborate on the details of our B-Mesh modeling we introduce some notations.

4.1. B-Mesh notations

The skeleton specified by the user is a hierarchical tree structure of nodes connected by rigid links, called *bones*, and provides a kinematic model of the shape. The skeletal nodes are classified into three types [WML*06]: *end node* (if it connects to only one bone, shown in yellow in Figure2(a)), *connection node* (if it connects to two bones, shown in blue in Figure2(a)), and *joint node* (if it connects to more than two bones, shown in red in Figure2(a)). One of the nodes is chosen as the root node of the skeleton, see the biggest red ball in Figure2(a) (also see the red ball in Figure 2(c)).

The balls associated with the skeletal nodes, which are specified by the user, are called *key-balls* (see Figure2(a)). The system generates the *inbetween-balls* centered at some sampling points on each bone by interpolating the two balls at its end nodes (see the gray balls in Figure2(b)). We call all key-balls and inbetween-balls as *shape balls*.

We call the skeleton and the shape balls as the *skeletal shape balls*. In B-Mesh system, the skeleton defines the global structure of the shape and the envelope of the shape balls defines the profile of the shape. The set of shape balls contribute to construct the initial mesh and define a scalar field to guide the evolution of its subdivided meshes. Thus the user only needs to manipulate the skeleton and the key-balls to generate the model.

4.2. B-Mesh initialization

The procedure of generating an initial mesh from the skeletal shape balls consists of two operations: *sweeping* and *stitching*. It is a recursive procedure from the root node to end nodes. First, we decompose the skeletal shape balls structure into several limbs each of which has no joint nodes. For example, there are three limbs which radiates from the root node (in red) in Figure 3(a). These limbs can be meshed by sweeping operator and then the limb meshes are stitched at joint nodes.

Sweeping Denote $\{X,Y,Z\}$ as the world frame. In order to perform the sweeping operator, a local coordinate frame

$\{x, y, z\}$ is automatically constructed for each bone between a joint node and one of its child node. The x axis is set as being parallel to the bone and then we set $y = Z \times x$ and $z = x \times y$, see Figure 3(a). If x is parallel to Z, we set y and z as X and Y respectively. As shown in Figure 3(b), the initial cross sections are created around the joint nodes, translated along the skeleton limbs and rotated around the connection nodes. The rotation axis is determined by two connected skeletal bones. For each child node of the joint node, the initial cross section (a rectangle in the yz-plane) is sweeping along the limb starting from the joint node. After the sweeping procedure, the quad mesh is created around each limb, see Figure 3(b). And the quad meshes have good edge flows along the bones.

Stitching Then we stitch the quad meshes around the joint nodes as follows. At each joint node, we construct the convex hull around it and triangulate the convex hull (see Figure 3(c)). We then merge two adjacent triangles into a quadrangle in a descending order of some score between them. The score measures how two adjacent triangles are close to a plane and is defined as $score = A(n_1 \cdot n_2)$, where A is the sum of two triangle areas, n_1 and n_2 are their unit normals respectively. In the B-Mesh system, the user might specify two limbs to be symmetric according to some plane. In this case, we amplify the score by a scale of 10 if two triangles are symmetric with respect to the symmetric plane so that they can be merged in priority.

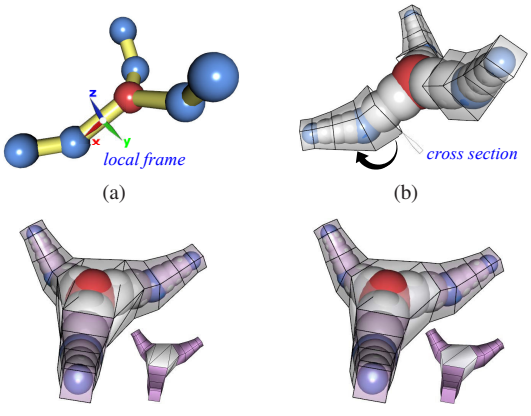


Figure 3: *Illustration of sweeping and stitching operations. (a) A local frame of a bone between a joint node and one of its children; (b) a cross section sweeping along a limb; (c) the triangular convex hull (white triangles) around a joint node; (d) the resulting quadrangle mesh after merging the triangles.*

4.3. B-Mesh evolution

Note that after stitching the result mesh is a quad-dominant mesh, that is, most of the mesh faces are quadrangles except for some triangles around the joint nodes. We then perform standard Catmull-Clark subdivision on the initial mesh and obtain quad meshes. However, the subdivision meshes will shrink and deviate from the envelope of the shape balls, as shown in Figure 2(d). We propose an evolution operation on the subdivision mesh to make it better approximate the shape.

Scalar field The evolution process is guided by a level set of a scalar field defined on the ball set. We use the following field function f_i:

$$f_i(r) = \begin{cases} (1-(\frac{r}{R_i})^2)^2, & r \le R_i \\ 0, & r > R_i \end{cases} \quad (1)$$

where

$$r^2 = (x-c_x^i)^2 + (y-c_y^i)^2 + (z-c_z^i)^2, \quad (2)$$

$R_i = \alpha r_i$, r_i is the radius of ball i, (c_x^i, c_y^i, c_z^i) is the center of ball i. We set $\alpha = 1.5$ in our system.

The scalar field is then defined by the point set satisfying the following equation:

$$\mathcal{I}(\boldsymbol{x}) = \sum_{i=1}^{n} f_i - T = 0, \quad (3)$$

where T is a threshold parameter. The threshold T controls how close the evolved mesh is to the scalar field. A larger T will generate a slimmer result mesh.

Evolution of active surfaces The initial mesh in Figure 2(c) is regarded as an active surface $S(t)$ and then is evolved by the scalar field defined above.

The active surface is an extension of active contour [KWT88] in 2D which has been studied in many contexts. Specifically, the moving trajectory of a point on the surface is considered as a function of time t, $\boldsymbol{x}(t)$.

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{n}(\boldsymbol{x},t)\mathcal{F}(\boldsymbol{x},\boldsymbol{n},\kappa,\mathcal{I},\cdots), \quad (4)$$

where $d\boldsymbol{x}/dt$ is the velocity, $\boldsymbol{n} = -\nabla\mathcal{I}/|\nabla\mathcal{I}|$ is the normal vector, $\mathcal{I}$ is a scalar field. The motion speed function $\mathcal{F}(\boldsymbol{x},\boldsymbol{n},\kappa,\mathcal{I},\cdots)$ is a signed scalar function that depends on the surface properties including position, normal, curvature and scalar field $\mathcal{I}$ etc. Particularly, $S(0)$ is the initial mesh.

The motion speed function $\mathcal{F}$ is decided by the scalar field and a target level set. We formulate it as follows:

$$\mathcal{F}(\boldsymbol{x},\boldsymbol{n},\kappa,\mathcal{I},\cdots) = (\mathcal{I}(\boldsymbol{x}) - \mathcal{I}_{target}) f(\kappa). \quad (5)$$

This formulation means that the vertex should evolve faster if it is further away the target level set. The curvature function in this formulation controls the evolving velocity adaptively for different regions. The scalar field near the small ball is denser than near the large ball. In current system, the function $f(\kappa) = 1/(1+|\kappa_1|+|\kappa_2|)$ performs well. κ_1 and κ_2 are the principal curvatures at vertex $\boldsymbol{x}$ of the mesh.

A vertex $\boldsymbol{x}(t)$ on the current mesh surface $S(t)$ is evolving as follows:

$$\boldsymbol{x}(t+\Delta t) = \boldsymbol{x}(t) + \boldsymbol{n}(\boldsymbol{x},t)\mathcal{F}(\boldsymbol{x},\cdots)\Delta t, \quad (6)$$

where Δt is the time step. To avoid ripples or oscillations

occurring during the evolving procedure, we should impose a constraint on the size of the time step. It is formulated as the Courant-Friedrichs-Lewy (CFL) stability condition more precisely [MBW*05]. The size of the time step should be constrained by the detail of the surface. It depends on the reciprocal of the maximal motion speed:

$$\Delta t \leq \frac{step}{\mathcal{F}_{\max}(\boldsymbol{x}(t),\cdots)}, \tag{7}$$

where $step$ depends on the smallest detail. We approximate it using $step = min\{r_i\}/2^k$ where k is the subdivision level.

The mesh evolution terminates as the surface is close to the scalar field within some error threshold. The evolution process is fast as the initial mesh is close to the target surface.

4.4. B-Mesh fairing

After the evolution, the edge flow of the result mesh might not well reveal the feature of shape. Then we do a local fairing on the result mesh as follows. Except the umbilical points (where $\kappa_1 = \kappa_2$) and irregular vertices (valence is not 4), we adjust their positions to make their radiant edges align with their principal directions, as shown in Figure 4. For a vertex $\boldsymbol{p}$, we minimize the following objective function in its tangent plane:

$$\begin{aligned} f(\boldsymbol{x}) &= |(\boldsymbol{a}-\boldsymbol{x})\cdot\boldsymbol{e}_v|^2 + |(\boldsymbol{b}-\boldsymbol{x})\cdot\boldsymbol{e}_u|^2 + |(\boldsymbol{c}-\boldsymbol{x})\cdot\boldsymbol{e}_v|^2 \\ &\quad + |(\boldsymbol{d}-\boldsymbol{x})\cdot\boldsymbol{e}_u|^2 \\ &\quad s.t.\ |\boldsymbol{x}_u| \leq \varepsilon \ and\ |\boldsymbol{x}_v| \leq \varepsilon, \end{aligned} \tag{8}$$

where $\boldsymbol{e}_v$ and $\boldsymbol{e}_u$ are the principal directions at vertex $\boldsymbol{p}$, and $\boldsymbol{a},\boldsymbol{b},\boldsymbol{c},\boldsymbol{d}$ are projected points of the direct neighbors of $\boldsymbol{p}$ in the tangent plane. It is a quadratic optimization problem with linear inequality constraints. We use an iterative active set method to solve it [MNT04].

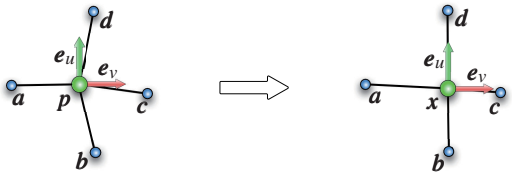


Figure 4: *An illustration of local fairing a vertex of valence 4. The point $\boldsymbol{p}$ (left) is moved to the point $\boldsymbol{x}$ (right) by minimizing the fairing energy function.*

For umbilical points and irregular vertices, we can not make their radiant edges align with their principal directions. We just modify their positions in their tangent planes. An approximated vertex is computed using the umbrella operator [KCVS98], and then projected onto the original tangent plane.

4.5. Implementation Details

Ellipsoids at skeletal nodes To generate some thin parts, our system allows the user to specify ellipsoids at skeletal nodes. Therefore, the user can specify the three radius of the ellipsoid and its orientation at the nodes. Only Equation (2) is modified as follows:

$$r^2 = \left(\frac{(\mathbf{p}-\mathbf{c})\cdot\mathbf{e}_\mathbf{x}^\mathbf{i}}{a}\right)^2 + \left(\frac{(\mathbf{p}-\mathbf{c})\cdot\mathbf{e}_\mathbf{y}^\mathbf{i}}{b}\right)^2 + \left(\frac{(\mathbf{p}-\mathbf{c})\cdot\mathbf{e}_\mathbf{z}^\mathbf{i}}{c}\right)^2, \tag{9}$$

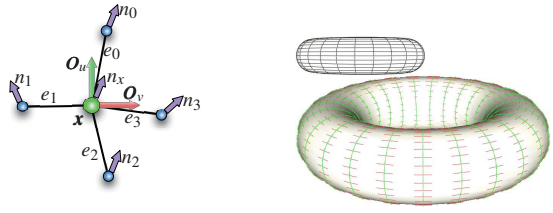
where $\mathbf{p} = (x,y,z)$, $(\mathbf{e}_\mathbf{x}^\mathbf{i},\mathbf{e}_\mathbf{y}^\mathbf{i},\mathbf{e}_\mathbf{z}^\mathbf{i})$ is the local frame of the ellipsoid, and a,b,c are the equatorial and polar radii of the ellipsoid.

Variation of scalar field Note that the evolution of the mesh surface is determined by the scalar field defined by the balls. To allow local manipulation for the shape, the user can add some auxiliary balls to affect the scalar field, and thus control the evolution of the mesh surface.

Curvature tensor estimation The evolving and fairing algorithms depend on the principal curvatures and principal directions. Fortunately, our quad mesh is typically regular (the valence of each vertex is 4) except in a few points. Here we develop an efficient algorithm for estimating the curvature tensor. Let $(\boldsymbol{o}_u,\boldsymbol{o}_v)$ be an orthogonal coordinate system in the tangent space T_xS of point $\boldsymbol{x}$. Then the Weingarten curvature matrix W in this coordinate system is symmetric [DoC76] as follows:

$$W\begin{pmatrix}(e_i^p)_u\\(e_i^p)_v\end{pmatrix} = \begin{pmatrix}(n_i-n_x)_u\\(n_i-n_x)_v\end{pmatrix}, \tag{10}$$

where n_x is the normal direction of point $\boldsymbol{x}$, $((x)_u,(x)_v)$ is the coordinate of x in the coordinate system $(\boldsymbol{o}_u,\boldsymbol{o}_v)$, as shown in Figure 5(a). e_i^p is the projection of e_i on the tangent plane T_xS. In our context, there are at least three neighbors at each vertex. This provides a set of linear constraints on the elements of the matrix W. We solve it in a least squares sense. The eigenvalues and eigenvectors of W are principal curvatures and principal directions. Our method is similar to the face based method which aims at triangle meshes [Rus04]. As shown in Figure 5(b), our estimation provides a correct principal directions.



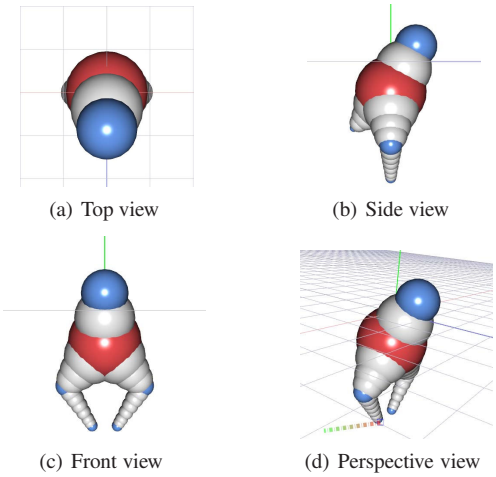
(a) (b)

Figure 5: *The neighborhood of a vertex of valence 4. (a) The radiate edges and local coordinate system; (b) the tessellation and principal directions of a torus model.*

5. Experimental results

We show a number of models generated from our modeling system in this section.

User interfaces The user interfaces of the B-Mesh system are similar to those of traditional modeling systems. However, our system allows users to focus on elements of skeletal shape balls (skeleton and key-balls) instead of elements of the mesh (vertices, edges, and faces). The user interactively specifies and edits the skeleton and key-balls of the desired shape and the system automatically generates it. Particularly, the user could specify and edit the skeletal shape balls in the *Top*, *Side*, *Front*, or *Perspective* views in the system (see Figure 6). After selecting a skeletal node, the user can insert a new node by clicking the mouse in the image plane or drawing a stroke in the image plane(see Figure 6(d)). The new point is then transformed into the object space and is attached to the selected node. If the object is reflectional symmetric, our system can also help the user create the symmetric part.



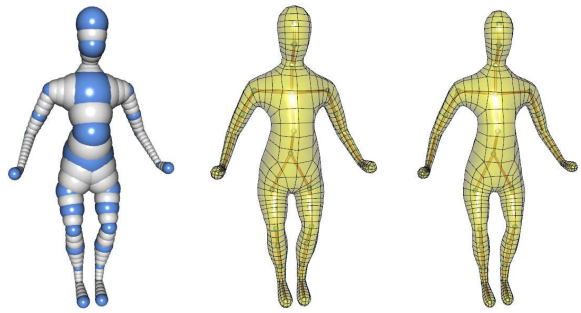
(a) Top view (b) Side view (c) Front view (d) Perspective view

Figure 6: *The user can view and edit the model in different views in our system.*

Modeling and manipulation Figure 7 shows a process of modeling an articulated shape from scratch. In this example, 27 balls were specified to build the whole model. It took about 3 minutes to create this model. Please see the accompanying video.

Generally the base mesh has only a few polygons. In all the examples shown in this paper, we set the number of subdivision iterations as 1 or 2 during the modeling process so that subdivision, evolution, and fairing of the mesh can be performed in an interactive rate. More subdivision can be performed when a fine model is needed after the interaction.

The parameter T in Eq. 3 measures the closeness between the evolved mesh and the envelope of shape balls. Larger T will generate slimmer result meshes, as shown in Figure 8. In our system we set $T = 0.3$ as in [MI87].



(a) (b) (c)

Figure 8: *The parameter T affects the evolved mesh. (a) Skeletal shape balls of a human shape; (b) the evolved mesh when we set $T = 0.3$; (c) the evolved mesh when we set $T = 0.45$.*

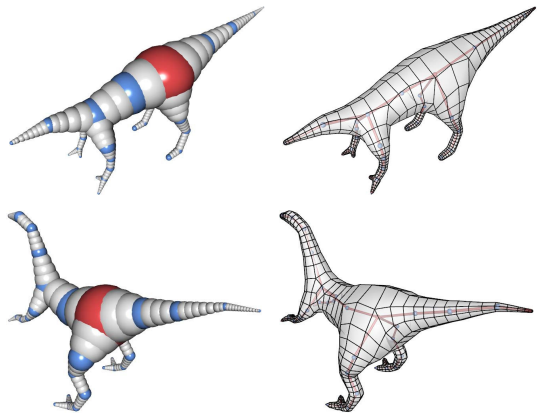
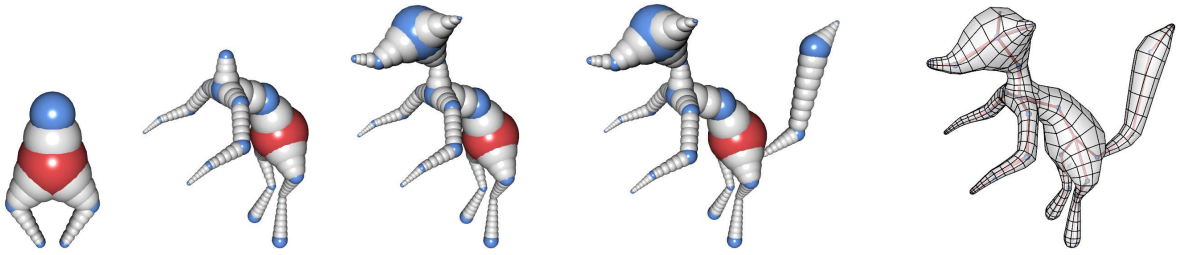


Figure 9: *The user can generate a new shape via manipulating existing skeletal shape balls. Upper: an existing model generated by our system; Lower: a new model generated by manipulating the skeleton and the key balls of the model in the upper row. Left: the skeletal shape balls; Right: the meshes.*

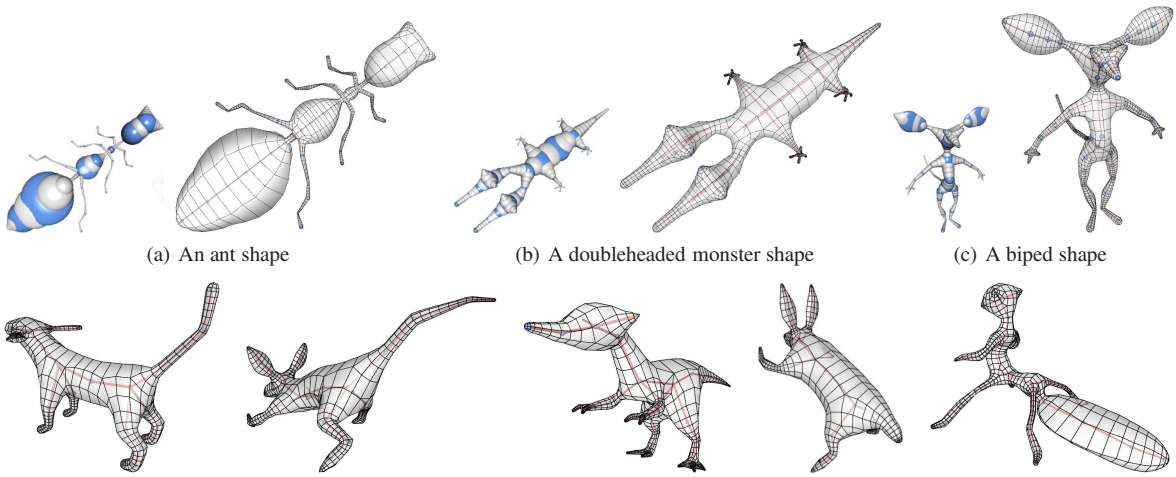
The user can also manipulate the shape during the modeling process. The user can easily edit the skeleton to change the pose of the shape. As shown in Figure 9, the user adjusts the positions of the skeletal nodes and modifies the key balls to obtain a new shape (c,d) from an old one (a,b). In Figure 10, the user adds 4 auxiliary balls near the belly part of the model and obtains an evolved mesh with a larger belly. In Figure 11, we show a model created in our system using ellipsoids. From this example, you can see that the user can generate various shapes by using ellipsoids in our system.

Figure 12 shows more results generated by our B-Mesh modeling tool. Some of them have a large number of joint nodes. It took about 5-20 minutes to create each of the models in our system. And most time is spent to *pose* them. It might take much more time for a reasonably skilled designer to generate similar shapes using modeling packages



(a) (b) (c)

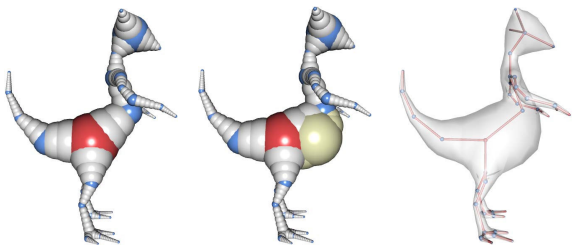
Figure 7: *An example modeling sequence. (a) Skeletal shape balls in different stages of modeling; (b) the final skeletal shape balls; (c) the result mesh.*



(a) An ant shape (b) A doubleheaded monster shape (c) A biped shape

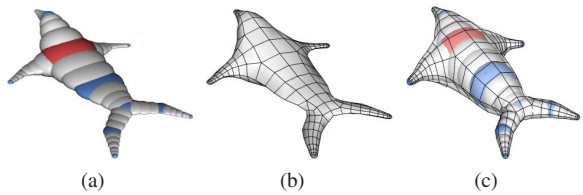
(d) More complex animal-like shapes

Figure 12: *Some models generated by our B-Mesh system.*



(a) (b) (c)

Figure 10: *The user adds auxiliary balls to locally manipulate the shape. (a) Initial skeletal shape balls; (b) adding 4 balls around the belly part of the model; (c) the skeleton and the resulting mesh.*



(a) (b) (c)

Figure 11: *An example using ellipsoids. (a) Skeletal shape balls with some ellipsoids; (b) the resulting mesh of (a); (c) changing the ellipsoids to obtain a plumper shape.*

like Maya and 3ds Max. While modeling using these traditional tools, the designer should keep a static pose of a target shape in mind, and use operations such as move, scale, rotate, and extrude to create and modify the polygon mesh according to the unknown shape in mind or the three-view drawings. Moreover it is inconvenient to modify the pose of the model in these packages.

Reusable models The skeletal shape balls structure is easy to create and edit. The most operations of skeletal shape balls are common for 3D modeling, such as *move*, *scale*, *rotate* and *remove*. *split* and *merge* operations, like "cut-and-paste" [FKS*04, JLCW06], are powerful tools in our framework. Given some existing skeletal shape balls, the user can reuse them to create new shapes in an easy manner. In doing so, one obtains the desired parts of models by *splitting*, then *merges* them at some skeletal node to form a new skeletal shape balls structure. An example is shown in Figure 13. The new model is created by mixing with skeletal shape balls of Figure 2 and Figure 11. It is very intuitive to reuse existing

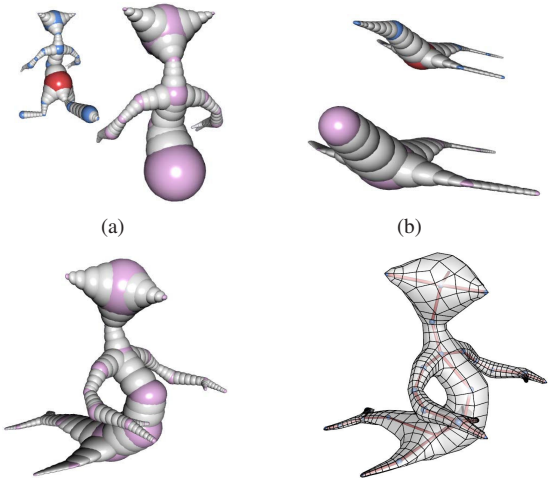


Figure 13: *An example of* Splitting *and* Merging *operations. (a,b) Two subparts (colored by pink and white) are split from two skeletal shape balls; (c) the subparts are merged into a new skeletal shape balls structure; (d) the resulting mesh.*

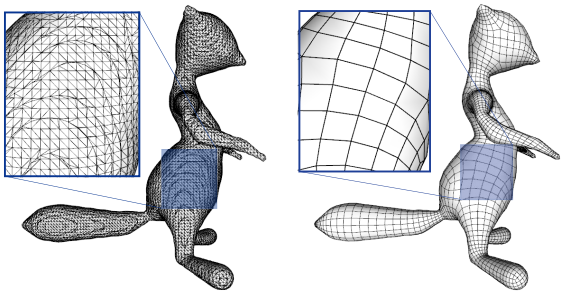


Figure 14: *Comparing with the traditional metaball modeling method. Left: the resulting tessellation generated by marching cubes in metaball is bad; Right: our method generates much better tessellation.*

designs in our system. However, it is nontrivial to merge two polygonal meshes in traditional modeling packages.

Comparisons As mentioned above, the traditional metaball system is a powerful modeling tool for modeling blob-like shapes. However, the models produced by the metaball system are of low quality triangle meshes. An example is shown in Figure 14. The resulting triangular mesh generated by the metaball system is much irregular. The marching cube mesh can be optimized to improve the quality or modified to output a quad mesh [GLR*06, LTJW07]. However, it is difficult to improve the edge flow of the mesh.

ZSpheres system provides a modeling way similar to our method. However, the user should be ware of the orientation of each sphere. When the user lays spheres freely, the resulting mesh may distort heavily. An example is shown in Figure 15(a). This drawback prevents ZSpheres from merging several parts into a new shape flexibly. Moreover, the resulting

Figure 15: *Examples of models generated by ZSpheres (upper row) and ZSpheres II (lower row). Upper row: the resulting model in the right generated from the left sphere set by ZSpheres is much distorted. Lower row: the resulting model in the middle generated from the left sphere set by ZSpheres II does not have good edge flow which does not align the skeleton.*

mesh does not enwrap the sphere set and lacks automatic rigging. The new version (ZSpheres II) has improved the performance. However, the edge flow dose not align with the skeleton (see the principal directions of surface shown in Figure 15(b)). ZSpheres does not solve these problems but asks the artists to modify the base shape in ZBrush.

Limitations Our B-Mesh system focuses on modeling base meshes of 3D articulated shapes for sculpting geometric details and skeleton based animation. It is not suitable for modeling architectures and mechanical objects with sharp features. It is also not feasible for modeling geometry features like creases, valleys and ridges. In our current implementation, users can not edit the shape by curve-based or surface-based controls. It might be interesting to explore further for the combination of different controls.

6. Conclusions and future work

We propose an easy and intuitive modeling system for generating the base meshes of 3D articulated shapes. The skeletal shape balls structure of the shapes is intuitive and reusable. The result meshes are quad dominant with good edge flows. The base meshes generated in our system can be directly used in many other 3D softwares, for both sculpting modeling and skeletal animation. To make it more flexible, we will allow users to define the profile of shapes and generate shapes by specifying the shape parameters. Another worthwhile future work is to design a skeletal shape balls library for reusable modeling purposes.

Acknowledgement. We would like to thank the anonymous reviewers for their valuable comments and suggestions. Lig-

ang Liu is supported by the 973 National Key Basic Research Foundation of China (No. 2009CB320801) and the joint grant of the National Natural Science Foundation of China and Microsoft Research Asia (No. 60776799).

References

[ABGC05] ALEXE A.-I., BARTHE L., GAILDRAT V., CANI M.-P.: Shape modeling by sketching using convolution surfaces. In *Pacific Graphics, Macau (China)* (2005).

[Aut10] AUTODESK: Mudbox 2010. *http://www.autodesk.com/-mudbox* (2010).

[Bli82] BLINN J.: A generalization of algebraic surface drawing. *ACM Transaction on Graphics 1*, 3 (1982), 235–256.

[BP07] BARAN I., POPOVIC J.: Automatic rigging and animation of 3D characters. *ACM Trans. Graph 26*, 3 (2007), 72.

[DoC76] DOCARMO M. P.: *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976.

[FKS*04] FUNKHOUSER T., KAZHDAN M., SHILANE P., MIN P., KIEFER W., TAL A., RUSINKIEWICZ S., DOBKIN D.: Modeling by example. *ACM Transactions on Graphics (Proc. of SIGGRAPH) 23*, 3 (2004), 652–663.

[GIZ09] GINGOLD Y., IGARASHI T., ZORIN D.: Structured annotations for 2D-to-3D modeling. *ACM Transactions on Graphics (TOG) 28*, 5 (2009), 148.

[GLR*06] GEIGER W., LEO M., RASMUSSEN N., LOSASSO F., FEDKIW R.: So real it'll make you wet. In *Proc. of SIGGRAPH Sketches* (2006).

[HYKJ03] HYUN D.-E., YOON S.-H., KIM M.-S., JÜTTLER B.: Modeling and deformation of arms and legs based on ellipsoidal sweeping. In *Pacific Conference on Computer Graphics and Applications* (2003), IEEE Computer Society, p. 204.

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: A sketching interface for 3D freeform design. In *Proc. of SIGGRAPH* (1999), pp. 409–416.

[JLCW06] JI Z., LIU L., CHEN Z., WANG G.: Easy mesh cutting. *Computer Graphics Forum (Proc. of Eurographics) 25*, 3 (2006), 283–291.

[KCVS98] KOBBELT L., CAMPAGNA S., VORSATZ J., SEIDEL H.-P.: Interactive multi-resolution modeling on arbitrary meshes. In *Proc. of SIGGRAPH* (1998), pp. 105–114.

[KHR02] KARPENKO O., HUGHES J. F., RASKAR R.: Free-from sketching with variational implicit surfaces. "Free-Form Sketching with Variational Implicit Surfaces", Computer Graphics Forum, September 2002,.

[KWT88] KASS M., WITKIN A., TERZOPOLOUS D.: Snakes: Active contour models. *Internation Journal of Computer Vision 1*, 4 (1988), 321–331.

[LTJW07] LIU L., TAI C.-L., JI Z., WANG G.: Non-iterative approach for global mesh optimization. *Computer-Aided Design 39*, 9 (2007), 772–782.

[MBW*05] MUSETH K., BREEN D., WHITAKER R., MAUCH S., JOHNSON D.: Algorithms for interactive editing of level set models. *Computer Graphics Forum 24*, 4 (2005), 821–841.

[MI87] MURAKAMI S., ICHIHARA H.: On a 3D display method by metaball technique. *Journal of papers given at the Electronic Communication J70-I*, 8 (1987), 1607–1615.

[MNT04] MADSEN K., NIELSEN H., TINGLEFF O.: Optimization with constraints, 2nd ed., 2004.

[NHK*85] NISHIMURA H., HIRAI M., KAWAI T., KAWATA T., SHIRAKAWA I., OMURA K.: Object modeling by distribution function and a method of image generation. *Trans. IECE Japan, Part D 68*, 4 (1985), 718–725.

[NISA07] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: Fibermesh: designing freeform surfaces with 3D curves. *ACM Trans. Graph 26*, 3 (2007), 41.

[NKS09] NASRI A., KARAM W. B., SAMAVATI F.: Sketch-based subdivision models. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling* (2009), pp. 53–60.

[Pix10] PIXOLOGIC: Zbrush. *http://www.zbrush.com* (2010).

[RSH04] ROCA X., SARRATE J., HUERTA A.: Surface mesh projection for hexahedral mesh generation by sweeping. In *International Meshing Roundtable* (2004), pp. 169–180.

[Rus04] RUSINKIEWICZ S.: Estimating curvatures and their derivatives on triangle meshes. In *3DPVT* (2004), IEEE Computer Society, pp. 486–493.

[SWSJ05] SCHMIDT R., WYVILL B., SOUSA M., JORGE J.: Shapeshop: Sketch-based solid modeling with blobtrees. In *Sketch Based Interfaces and Modeling* (2005), pp. 53–62.

[TZF04] TAI C.-L., ZHANG H., FONG J. C.-K.: Prototype modeling from sketched silhouettes based on convolution surfaces. *Computer Graphics Forum 23*, 1 (2004), 71–84.

[VFLL05] VERMA C. S., FISCHER P. F., LEE S. E., LOTH F.: An all-hex meshing strategy for bifurcation geometries in vascular flow simulation. In *International Meshing Roundtable* (2005), pp. 363–375.

[WML*06] WU F.-C., MA W.-C., LIANG R.-H., CHEN B.-Y., OUHYOUNG M.: Domain connected graph: the skeleton of a closed 3D shape for animation. *The Visual Computer 22*, 2 (2006), 117–135.

[WMW86] WYVILL G., MCPHEETERS C., WYVILL B.: Data structure for soft objects. *The Visual Computer 2*, 4 (8 1986), 227–234.

[ZB04] ZHANG Y., BAJAJ C. L.: Adaptive and quality quadrilateral/hexahedral meshing from volumetric imaging data. In *International Meshing Roundtable* (2004), pp. 365–376.