COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2015; **26**:469–481
Published online 4 May 2015 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.1656

SPECIAL ISSUE PAPER

Efficient sketch-based creation of detailed character models through data-driven mesh deformations

Ismail Khalid Kazmi[1]*, Lihua You[1], Xiaosong Yang[1], Xiaogang Jin[2] and Jian J. Zhang[3,4]

[1] National Center for Computer Animation, Bournemouth University, Dorset, UK
[2] State Key Lab of CAD&CG, Zhejiang University, ZheDa Road 38#, Hangzhou, Zhejiang Province, China, 310027
[3] The Media School, Bournemouth University, Dorset, UK, BH12 5BB
[4] Traction Power State Key Laboratory, Southwest Jiaotong University, Chengdu, China

ABSTRACT

Creation of detailed character models is a very challenging task in animation production. Sketch-based character model creation from a 3D template provides a promising solution. However, how to quickly find correct correspondences between user's drawn sketches and the 3D template model, how to efficiently deform the 3D template model to exactly match user's drawn sketches, and realize real-time interactive modeling is still an open topic. In this paper, we propose a new approach and develop a user interface to effectively tackle this problem. Our proposed approach includes using user's drawn sketches to retrieve a most similar 3D template model from our dataset and marrying human's perception and interactions with computer's highly efficient computing to extract occluding and silhouette contours of the 3D template model and find correct correspondences quickly. We then combine skeleton-based deformation and mesh editing to deform the 3D template model to fit user's drawn sketches and create new and detailed 3D character models. The results presented in this paper demonstrate the effectiveness and advantages of our proposed approach and usefulness of our developed user interface. Copyright © 2015 John Wiley & Sons, Ltd.

KEYWORDS

detailed character creation; sketch-based modeling; 3D template models; skeleton-based deformation; mean value coordinates; Laplacian mesh editing

Supporting information may be found in the online version of this article.

***Correspondence**

Ismail Khalid Kazmi, National Center for Computer Animation, Bournemouth University, Poole, Dorset UK.
E-mail: ikazmi@bournemouth.ac.uk

1. INTRODUCTION

Sketch-based modeling (SBM) has a great potential in enhancing and improving conventional 3D modeling and sculpting workflows in the animation industry [1]. Over the past three decades, the research in SBM has seen substantial advances. The existing SBM approaches can be broadly divided into two categories: (1) direct mesh generation and (2) template-based character creation also called data-driven character modeling in this paper. Direct mesh generation aims at generating geometrical objects from user's drawn sketches directly such as Teddy by Igarashi *et al.* [2]. Its weakness is that it is not suitable for modeling detailed or complex character models. Template-based modeling uses a 3D template model to provide the information about details and complex shapes of character models and avoid the weakness of the direct mesh generation. It provides a promising solution to create detailed or complex 3D character models from a simple sketch.

The current state of the art in template-based modeling focuses on highly automatic extraction of occluding contours. Because of complex topology and shapes of character models, the automatic methods lead to incorrect results and have known to produce inconsistent contours [3]. Another problem is the inefficient computing of one-to-one correspondence between user's drawn sketches and the 3D template models and time-consuming mesh deformation algorithm to deform the 3D template models to fit user's drawn sketches. These problems make current template-based character modeling impractical in real-time interactive modeling applications.

In order to tackle the aforementioned problems, this paper proposes a novel approach to efficiently create detailed and high quality 3D character models from user's drawn

sketches. We argue that, for a sketch based modeling system to be valuable, combining human's perception and necessary interactions with computer's powerful computing capacity are vitally important to aid the computer in quickly finding accurate correspondences. This hypothesis has served as one of the basic ideas behind our proposed approach. The second idea is to tackle time-consuming iterative calculations caused by using mesh editing algorithms by incorporating skeleton-based large deformation and mesh editing's small deformation to efficiently deform a 3D template model to exactly fit user's drawn sketches. The third idea is to develop a user interface to integrate all the functions into a complete system to speed up creation of character models. As shown in Figure 1, with our developed user interface, users first interactively create user's drawn sketches with or without a 2D reference image. Then, a 3D template model whose occluding contours best match user's drawn sketches is automatically retrieved from our developed dataset. Next, preliminary occluding contours of the 3D template model are automatically generated to guide users to create correct and consistent occluding contours on the mesh surface of the 3D template model. After that, with help of very few anchor points specified by user input, automatic subdivision is performed to help determine correct correspondences between user's drawn sketches and the 3D template model. Finally, skeleton-based deformation and mesh editing are combined to determine large and small mesh deformations, respectively, and generate new character models from the 3D template model.

The following are the main contributions of our work presented in this paper:

(1) Combining human's perception and necessary interactions with computer's powerful computing capacity to develop a fast and accurate method of determining correct correspondences between user's drawn sketches and the 3D template model.
(2) Proposing a hybrid mesh deformation technique that employs skeleton-based deformation to tackle large mesh displacements or deformations and mesh editing to deal with small mesh deformations for quickly deforming the 3D template model into a new model exactly matching user's drawn sketches.
(3) Developing a user interface with complete functions to facilitate and speed up creation of new character models from user's drawn sketches and a template model.

The remaining parts of this paper are organized in the succeeding paragraphs. In Section 2, we discuss the work related to our approach. Section 3 explains our approach in detail and the steps involved in creation of new character models from user's drawn sketches. Section 4 presents experimental results and comparison. Finally, Section 5 concludes the work described in this paper.

2. RELATED WORK

Existing sketch-based modeling systems can be used to model organic and inorganic models from a single-view [2] or multiple-view [4] sketches. Cook and Agah [5], and Olsen *et al.* [1] have presented thorough surveys on sketch-based modeling techniques. Some notable systems for modeling sketch-based inorganic models are [6], [7], [4], and [8]. In [7], the authors have introduced a new mathematical framework for inferring 3D curve networks from 2D drawings that are drawn from a perspective view, inspired by design, vision, and perception literature. However, their technique does not seem to work well for sketches that are drawn from orthographic views and is not suitable for modeling organic characters. Chen *et al.* [6] have demonstrated a 3D model reconstruction technique from single photos by automatically fitting the boundaries of generalized cylinder to the boundaries of the subject in the image. Rivers *et al.* [4] have proposed a new and simple algorithm toward computing the silhouette cylinders to compute the 3D constructive solid geometry by leveraging the special properties of silhouette cylinders. Bae *et al.* [8] have developed a robust and feature-rich system for 3D sketching. The basic "feel" of the system borrows from that of a physical paper sketchbook.

It has been observed that modeling of organic human characters is a highly challenging task for artists, and some SBM systems have tried to overcome these challenges such as [9] and [10]; but these systems possess several limitations, for example, these techniques do not cater for producing highly realistic models. Moreover, the modeling of body parts that are hidden in input sketches is highly challenging. To overcome these limitations, some researchers have proposed template-based or data-driven

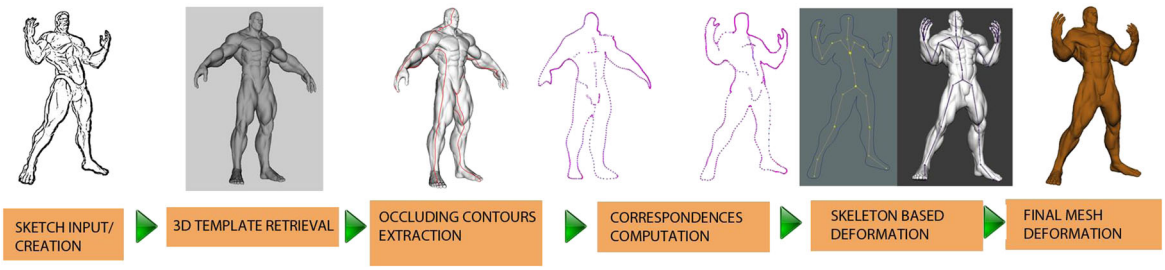


Figure 1. Process flow of our proposed approach.

Comp. Anim. Virtual Worlds 2015; **26**:469–481 © 2015 John Wiley & Sons, Ltd.
DOI: 10.1002/cav

systems that use a 3D human character as a template model and deform this model by either morph it or use cage based deformation techniques [11]. The area of template-based modeling for generating organic or human characters particularly from input sketches has been visited very rarely by researchers in the past. Some good examples of template based SBM systems are [12] and [13]. Shtof *et al.* [12] have introduced a snapping technique that automatically reshapes and snaps simple 3D geometric primitives to 2D sketch primitives and then improves the model globally by inferring geosemantic constraints that link the different parts. This technique leads to incorrect topology or tedious modeling when it comes to modeling human characters.

An elegant technique for sketch-based modeling has been proposed in [13] that involves finding precise correspondences and mesh deformation. For finding one-to-one correspondences, the hidden Markov model (HMM) has been harnessed [14], which is a computationally expensive algorithm. For deforming the mesh, the mean value coordinates (MVC) has been used, which incurs a huge bottleneck in the overall algorithm [15]. The method is iterative in nature and solves the energy minimization problem to obtain a deformed mesh in which each iteration includes heavy computations for finding correspondences and mesh deformation. On a mesh of more than 20,000 polygons, HMM combined with MVC incurs an overall delay of 2–10 min. This limitation renders their algorithm impractical in an interactive modeling environment.

Template-based modeling is heavily dependent on the extraction of feature curves. Feature curves are classified as silhouette contours, occluding contours, and suggestive contours as shown in [16]. A powerful system has been developed in [17] for extracting feature curves in real time from 3D models. However, the occluding contours extracted by their technique produces inconsistent (broken) and non-sequential contours. Benard *et al.* [3] has attempted to solve a long occurring problem of finding inconsistent contours using the method presented by [18] and [17]. However, as mentioned by the authors, their algorithm is computationally expensive and does not guarantee 100% contour consistency. Because of this reason, their method is infeasible to use within the context of our approach. In this paper, we only focus on extracting occluding contours as a sketching guide for the user so that the user can sketch a curve on the 3D mesh as close as possible to the occluding contours.

Several techniques have been proposed for mesh deformation. Nieto and Susin [19] have discussed and compared several useful deformation algorithms including MVC, harmonic functions, and green coordinates in their survey and concluded that MVC is the best performing algorithm for deformation from the perspective of efficiency. Several techniques exist for skeleton-based deformation. Chen *et al.* [20] have proposed a user friendly interface to generate a control skeleton of a model automatically, and a skeleton segment can be adjusted by drawing a new curve. Buchanan *et al.* [9] have demonstrated a system that generates low-polygon models directly from input character sketches without any interactivity by the artist.

3. SYSTEM OVERVIEW

As indicated in Figure 1, our approach consists of the following steps: sketch input/creation, template model retrieval, occluding contours extraction, correspondence computation, skeleton-based deformation, and mesh editing. We elaborate them in the sub-sections below.

3.1. Sketch Input/ Creation

As discussed earlier, users drawn sketches can be divided into three different types: silhouette contours, occluding contours, and suggestive contours. A comparison [16] among the three different types of contours is illustrated in Figure 2.

Suggestive contours and occluding contours are both vitally important for our proposed approach to work. We need suggestive contours to compute the shape descriptor and retrieve the 3D templates model from dataset (Section 3.2). However, for template model retrieval, we provide the user the freedom to use any type of contours to draw the input sketch. Because of the robustness of 3D model retrieval algorithm, we incorporated in our approach (Section 3.2.3). Occluding contours are also needed by our system to find correspondences (Section 3.4).

In our system, the user can use a reference image as the input sketch (Figure 3(a)). Our system also allows the user to trace occluding contours over the input sketch (Figure 3(b)). Alternatively, if a reference image is unavailable, he/she can draw the sketch with the help of our prototypical interface. Figure 3(c) depicts the occluding

Figure 2. Comparison among (a) silhouette contour, (b) occluding contour, and (c) suggestive contour of a face model. (Image source: Wuhrer and Wu [16]).

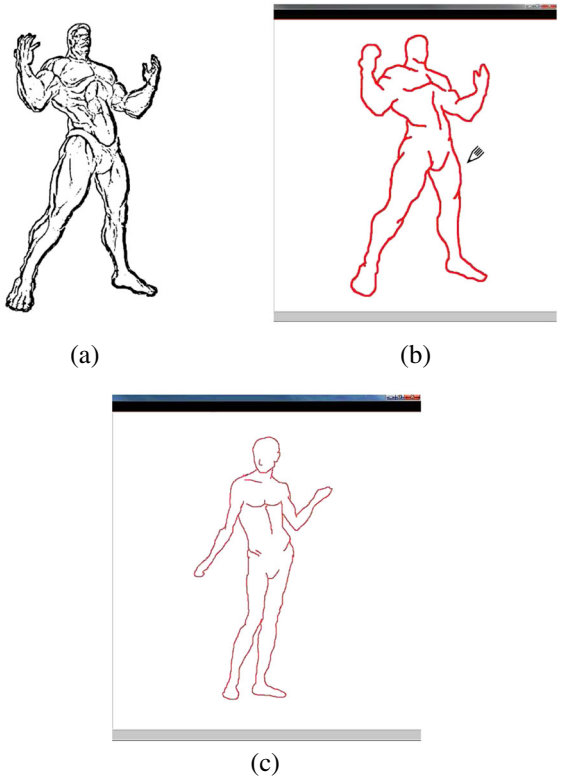


Figure 3. (a) Input 2D image (b) user-drawn occluding contours with reference to the input image, and (c) user-drawn occluding contours without using reference image.

contour created without referring to any 2D images. The occluding contours drawn by the user are called the created occluding contours and represented within our system as polylines rendered on the XY-plane and the value of z-coordinate is set to 0.

3.2. Template Model Retrieval

In this step, we will use the input sketch as an input query to retrieve a template model from our 3D dataset directly. To achieve this goal, the following algorithms are conducted: creation of a dataset of 3D template models, extraction of suggestive contours, and retrieval of 3D template models.

3.2.1. Dataset of 3D Template Models

We have setup a dataset of 3D template models through (1) collecting 3D human character models from the Internet including available 3D databases and (2) generating them using MakeHuman, which is an open source tool for making 3D character models (www.makehuman.org/). The created dataset has been included in our developed system. In the dataset, each model is generally composed of a high polygon count (more than 25,000 polygons). All the models are in either T-pose or A-pose. To enrich the variety of our dataset we keep adding more models. In addition to the template models, our dataset also contains 2D sketched renders (rendered using suggestive contours, occluding contours, and silhouette contours), as well as 2D anchor point specified on each 2D sketched render rendered with occluding contours. The creation of our dataset is an offline process, and the process of generating 2D sketched renders is explained in the following sections.

3.2.2. Extraction of Suggestive Contours

Our system automatically generates 2D sketched contour representations of the models in the dataset. To acquire 2D sketched renders from 3D models, we have used a non-photorealistic rendering algorithm proposed by DeCarlo *et al.* [17]. It generates suggestive contours and silhouette contours of organic 3D models in real time (Figure 4).

Described user drawn sketches can be from any view angle. In order to obtain 2D sketched renders of 3D models in our dataset that is from the view angle same as or close to that of the created sketches, we position a camera at different points for each of models in our dataset to take the images of the model. These points are on four circumferences circling a bounding sphere of the model and the camera points toward the center of the model as shown in Figure 5.

3.2.3. Retrieval of 3D Template Models

After extracting 2D sketched renders, our system computes the feature descriptor vector of each 2D view. We have drawn inspiration from [21] and used a powerful shape descriptor based on Gabor Filters to compute the shape descriptors. We divide the input sketch and 2D sketched render into $n \times n$ tiles and create a feature vector from these tiles (Figure 6(a)). We compare these two feature vectors using a simple distance metric [21]. Based on this metric, we retrieve a model that most closely match the user's created sketches and present that to the user. Figure 6(a) gives such an example. In the figure using a user's created sketch (left), our system performs a comparison between feature vectors of the user's sketch and a 2D sketched render of a model in the dataset. In Figure 6(b), we show the two most similar models found in the dataset where the middle model is the most similar model.

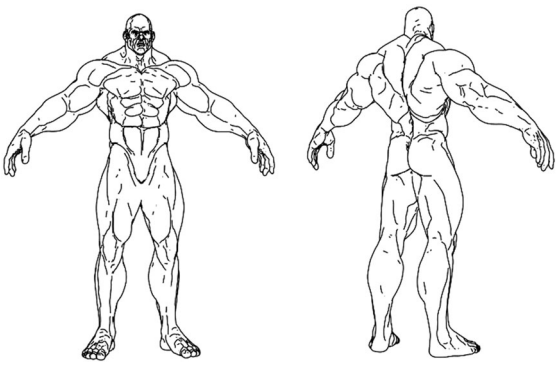


Figure 4. Suggestive contours and silhouette contours generated using [17] showing the model as a line drawing.

DOI: 10.1002/cav

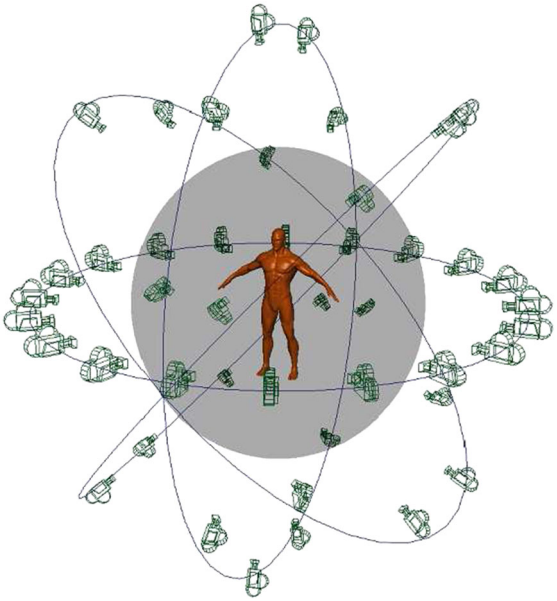


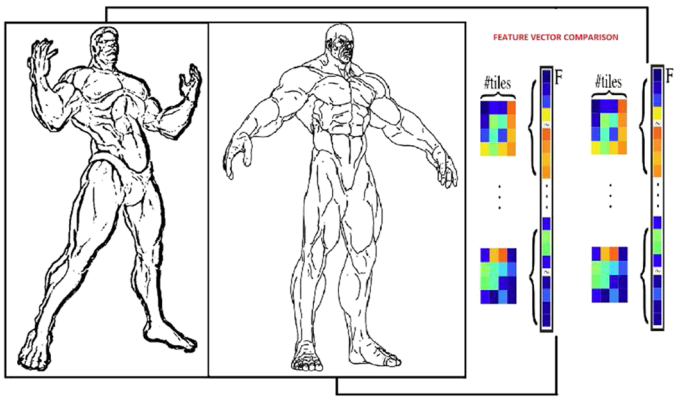
Figure 5. Camera setup for extracting 2D sketched render views of a 3D model in the dataset.

From the retrieved template model, we store the coordinates and view direction of the camera angle that will be used to compute the occluding contours and correspondences in the subsequent steps.

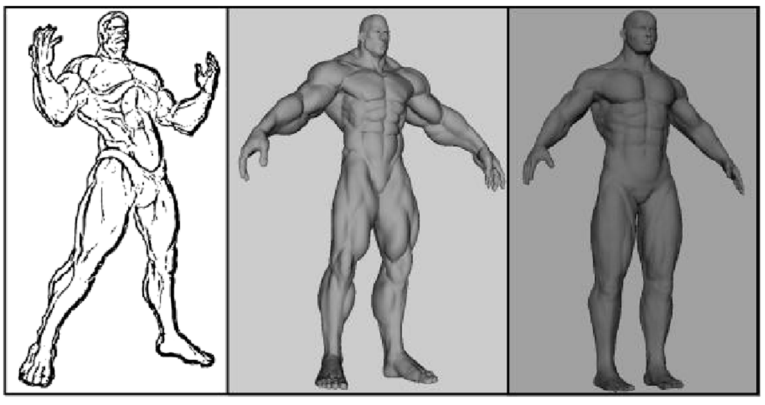
3.3. Occluding Contours Extraction

In order to deform the extracted 3D template model to make it exactly match the user's created sketch, we carry out the following research studies. First, we investigate how to generate occluding contours of the 3D template model that will be used to provide guidance for the user to sketch the curve on the surface of the template model in this subsection. Second, we examine the correspondence between the user's created sketch and the sketched curve on the template model.

We first require the user to draw the occluding contours of the input sketch, which can be easily drawn by simply tracing over the silhouette contours and drawing some internal contours that are enough to convey the inner shape. We only require this in case the user has not already provided occluding contours at the start of the pipeline. Our system then computes the occluding contours of the



(a)



(b)

Figure 6. (a) Comparison between feature vectors of the user's sketch and a 2D sketched render of a model in the dataset. (b) Models similar to input sketch are ranked here. The middle model is the most similar. The right model is the second most similar.

template model and renders them on the mesh as shown in Figure 8. With the method developed in [17], occluding contours are extracted from the 3D model using the expression defined in Equation (1). Let n be the normal vector at a point p and v be the vector pointing from the point p to the camera. On a smooth mesh, the occluding contours are made up of vertices where the dot product of the view vector and the normal vector is zero (Figure 7), which is represented as

$$n(p).\, v(p) = 0 \tag{1}$$

To satisfy Equation (1) exactly, the extracted occluding contours contain too few vertices to be used in the subsequent steps in our case. In order to acquire enough vertices to serve guidance for the user, we chose to select vertices with the modified Equation (1), that is, $-0.3 \leq n(p) \cdot v(p) \leq 0.3$. Such a treatment extracts sufficient vertices. However, as shown in Figure 8, the extracted vertices do not lead to desired occluding contours, which cannot be used to determine correspondences effectively.

To solve this problem, our approach proposes a user-friendly method to generate correct occluding contours in a semi-automatic fashion thus making our system to compute correspondences very quickly. We request the reader to see the accompanying video for a demonstration of this step. Such a method uses the extracted occluding contours to guide the user to draw smooth and sequential occluding contours on the template model surface. The user's drawn sketch is represented with a B-spline curve called a *guided occluding contour*, which will be used in the correspondence computation discussed in the subsection in the succeeding paragraphs.

3.4. Correspondence Computation

Our proposed correspondence computation is based on three basic ideas. The first is to manually select very few anchor points on the guided occluding contour (B-spline curve) and the corresponding anchor points on the user's created 2D silhouette contour. The second is to generate the same number of points between the adjacent anchor points on both B-spline curve and the user's created sketch

$$n(p).\, v(p) = 0 \tag{1}$$

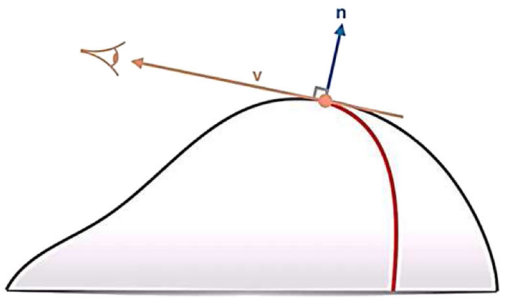


Figure 7. Occluding contour (red curve) on a smooth surface.

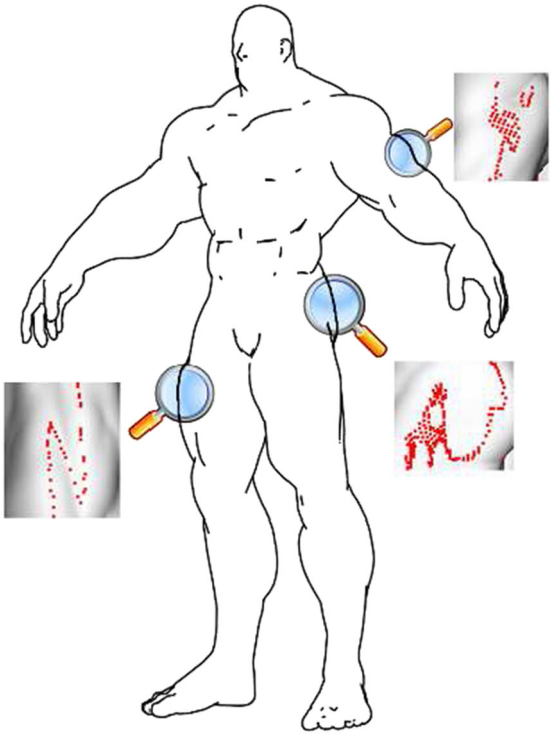


Figure 8. Inconsistent occluding contours magnified on the template model.

that solves the correspondence problem between the B-spline curve and the user's created sketch. The third is to determine the correspondence vertices on the 3D template model by using the KD-tree algorithm to find the vertices closer to the points on the B-spline curve that solve the correspondence problem between the extracted 3D template model and the user's created sketch. These three basic ideas will be elaborated in the succeeding paragraphs.

3.4.1. Selection of Anchor Points

After the user has obtained the guided occluding contour (B-spline curve), he/she can select very few anchor points on the guided silhouette contour and their corresponding anchor points on the 2D user's created sketch to aid our algorithm in finding correct correspondences. This step is very easy for a human to perform, as humans are very quick at identifying these apparent correspondences, as compared with a computer. A maximum of 10 anchor points is sufficient. However, finding the final one-to-one correspondence is a cumbersome and error prone task for a human. In the next subsection, we explain our proposed algorithm of finding the final correspondences accurately and quickly.

3.4.2. Determining Correspondence Between Created and Guided Occluding Contours

To establish a one-to-one correspondence between the user input sketch (created occluding contours) and guided

DOI: 10.1002/cav

occluding contours, the number of points in the 2D curves must be equal to the number of vertices in the traced 3D occluding contours. Our algorithm has achieved this in two steps. First, we decompose the input and guided occluding contours into a set of sub-curves using the anchor points selected by the user in the previous step, as boundaries of the sub-curves. We then subdivide the sub-curves to insert new curve points thus resizing two corresponding sub-curves to contain equal number of points. Our algorithm adds new curve points between the original curve points and updates positions of the original curve points thus making it smoother. We define two rules for this subdivision: (1) odd rule (Equation (2)) and (2) even rule (Equation (3)). Our system first resizes the curves using the odd rule by inserting more points in the curve that contains fewer points and then evenly distributes the points in the two curves using the even rule (Figure 9).

$$p_{2i+1}^{j+1} = 1/8\left(4p_{2i+1}^{j+1}\right) \tag{2}$$

$$p_{2i}^{j+1} = 1/8\left(p_{i-1}^{j} + 6p_{i}^{j} + p_{i+1}^{j}\right) \tag{3}$$

Once the two curves are resized to contain equal number of points, the system computes a one-to-one correspondence between these guided occluding contours (B-spline curve) and the input sketched contours (Figure 10).

3.4.3. Determining Correspondence Between Created Occluding Contours and the 3D Template Model

The last work of correspondence computation is to find the vertices on the 3D template model that correspond to the points on created occluding contours. This can be achieved with the following K-dimensional tree (KD-tree) algorithm. The KD-tree algorithm finds the vertices on the 3D mesh that are closest to points on the guided occluding contour. The obtained vertices are naturally in sequence and give us the accurate correspondences with the points on the created occluding contours. An alternative to KD-tree algorithm is the BSP (binary space partitioning). BSP is a generic process of recursively dividing a scene into two until the partitioning satisfies one or more requirements. Because BSP algorithm has not been used in existing research in the context of finding nearest vertices on a 3D mesh, we use the KD-tree algorithm. The KD-tree is an elegant data structure for range searching and nearest neighbor searching. It was first proposed by John Louis Bentley [22] and is designed to handle special data in a simple way. Utilizing KD-tree algorithm makes our approach very fast as compared with

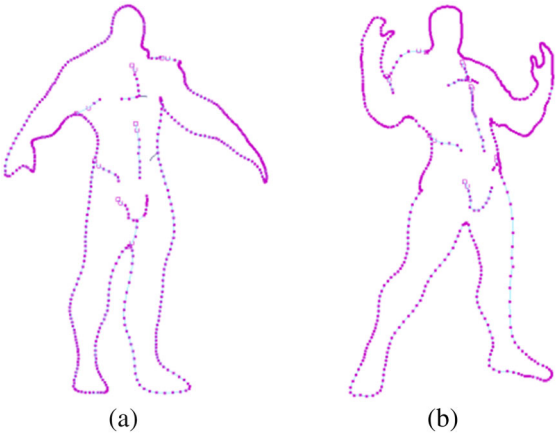


Figure 10. The contours on the left show guided occluding contours that will be corresponded with the input sketch contours on the right.

the heavy computations performed by the HMM in Kraevoy's method [13]. In an HMM, the input is a sequential series of observed states and the goal is to infer the corresponding sequence of hidden states that is most likely to have generated these observations. Their method for computing correspondences is a semi-automatic method as it requires the user to select a few anchor points to assist the algorithm to establish the right correspondences such as the case in which the two rear legs of the lioness are incorrectly deformed unless the user manually selects some corresponding vertices to accomplish correct correspondences (Figure 11).

With the above KD-tree algorithm, we obtained the vertices on the 3D template model that form the read curve in Figure 12(b) where the white curve in Figure 12(a) is the guided occluding contour.

We have found that with some necessary user interaction in our approach, the overall performance of finding correspondences has been boosted.

3.5. Mesh Deformation

When deforming a 3D template model to fit the input sketch contours, large mesh displacements or deformations may be involved as indicated in Figure 13. If there is a 3D template model in the dataset whose occluding contours are close to the user's drawn contours, the mesh editing algorithms described in Subsections 3.5.2 and 3.5.3 can be applied directly. However, if such a template model

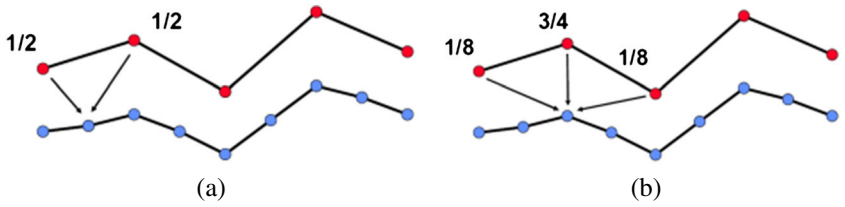


Figure 9. (a) Odd-rule inserts new points in the curve and (b) even-rule repositions the old points.

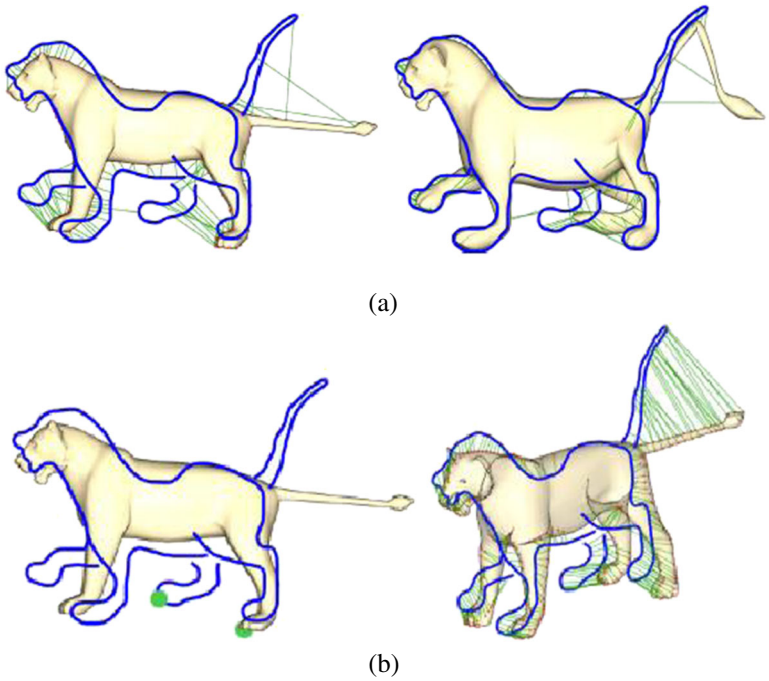


Figure 11. (a) Incorrect correspondence and deformation without specifying anchor vertices and (b) Correct correspondence and deformation with specifying anchor points for correspondences. (Image source: Kraevoy *et al.* [13]).

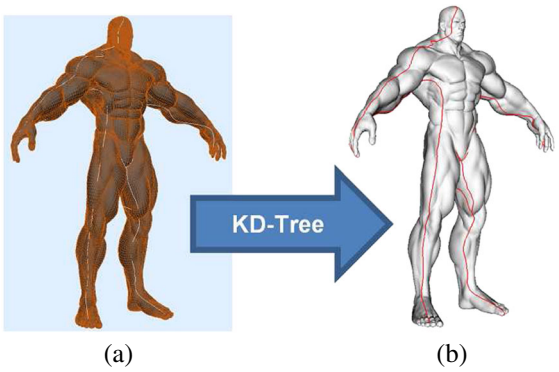


Figure 12. (a) Guided occluding contours drawn by the user and (b) Accurate occluding contours achieved by our approach after applying KD-tree algorithm.

does not exist in our dataset, the pose of the user's created sketches might be quite different from the pose of the 3D template model. One such example is to deform the 3D template model whose occluding contours shown in Figure 10(a) exactly match the user's created occluding contour shown in Figure 10(a). For this case, direct use of mesh editing algorithms will have to face large mesh displacements or deformations and involve time-consuming iterative calculations.

As reported in [13], when using the MVC, an iterative match-and-deform algorithm is involved. This typically requires a total of 10–20 outer iterations of matching and deformation to obtain the final result leading to an entire process of 2–10 min, which makes the algorithm unsuitable in an interactive modeling environment. In order to tackle this problem, we propose a hybrid skeleton-based deformation and mesh editing scheme in which skeleton-based deformation deals with large displacements or deformations; mesh editing removes the small deformation errors between the input sketch contours and the 3D template model after skeleton-based deformation. Such a treatment avoids iterative match-and-deform calculations and achieves efficient and accurate mesh deformation. In the existing references, mainly four mesh editing algorithms exist. They are: Poisson mesh editing [23], Harmonics coordinates [24], MVC [15,25,26], and Laplacian mesh editing [27]. Poisson mesh editing is mostly used to create a transition surface to smoothly connect two separate surfaces or models together. Harmonic coordinates have been known to be used in cage-based deformation. In contrast, both MVC and Laplace coordinates have been used in tackle mesh editing problems similar to the one in this paper. Here, we use both of them in Subsections 3.5.2 and 3.5.3 to remove the deformation errors between the created occluding contours and those of the 3D template model after skeleton-based deformations.

3.5.1. Skeleton-Based Deformation

As discussed earlier, when the pose of the created occluding contours is quite different from the pose of the 3D template model, skeleton-based deformation will be applied first. This involves (1) creating skeleton of the created occluding contours and the 3D template model: (2) automatically applying geometric transformations such

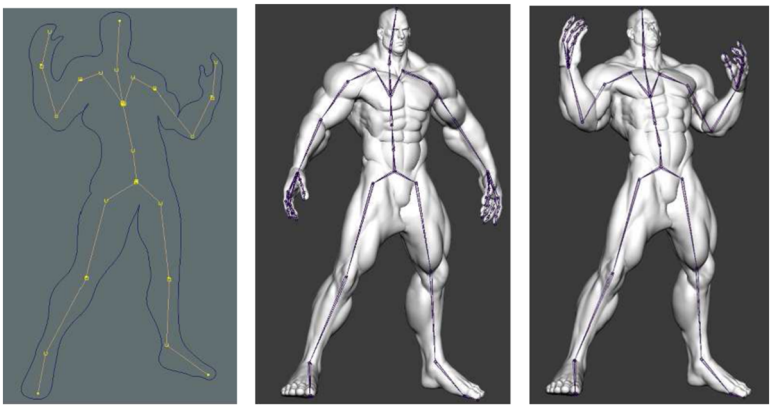


Figure 13. Skeleton-based deformation: (a) skeleton of the input sketch, (b) skeleton of the 3D template model, and (c) the new mesh shape of the 3D template model after geometric transformations.

as translation, rotation, and scale to align the skeleton of the 3D template model exactly with the skeleton of the created occluding contours; and (3) using skeleton-driven skin deformation algorithm to obtain a new mesh shape deformation caused by skeleton movements. For the created occluding contours shown in Figure 10(b) and the 3D template model indicated in the middle image of Figure 6b, the created skeletons are depicted in Figure 13. In the figure, (a) indicates the skeleton of the created occluding contours and (b) demonstrates the skeleton of the 3D template model. After automatically applying geometric transformations, the skeleton of the 3D template model in Figure 13(b) is changed into the skeleton in Figure 13(c) that is well aligned with the skeleton of the created occluding contours in Figure 13(a). We use a powerful skeleton-based skin deformation technique by Lewis *et al.* [28] to determine the new mesh shape of the 3D template model. With this algorithm, the new position of a deformed vertex is determined by transforming the vertex at the initial pose through Equation (4) in the succeeding paragraphs. At a particular skeletal configuration c, a deformed vertex V_c can be computed by

$$V_c = \sum_{i=1}^{n} W_i M_{i,c} M_{i,d}^{-1} V_d \tag{4}$$

where W_i are the weights, V_d is the location of a vertex at its initial pose, $M_{i,c}$ denotes the transformation matrix associated with the $i-th$ joint in configuration c, and $M_{i,d}^{-1}$ is the inverse of the transformation matrix associated with the *i-th* influencing joint.

Using Equation (4), we obtain the new mesh shape of the 3D template model after whose skeleton has aligned with the skeleton of the input sketch contours in Figure 13(a) and depicted it in Figure 13(c) as well. It can be seen that the new mesh shape of the 3D template model is very close to the input sketch contours, leaving very small deformation errors to be removed by mesh editing to be examined in the following two subsections.

3.5.2. Final Deformation Using Mean Value Coordinates

We have used the algorithm presented in [15] to compute the final mesh deformation. For a mesh with vertices V and edges E the mean-value Coordinates for each Vertex $v_i \in \mathrm{V}$ is computed from the Euclidean coordinates of the vertex and its m neighbor vertices v_j, where $(i,j) \in E$.

$$v_i = F_i(V) = \sum_{(i,j)\in E} w_{ij}\left[v_j - \left(d_i + v_j \bullet n_i\right)n_i\right] + h_i n_i \tag{5}$$

Here, h_i is the vertex offset above the projection plane. W_{ij} are the weight functions and normalizing each weight function by the sum of all weight functions gives us the MVC. d_i is the average distance from the origin. To achieve final deformation of the mesh, we solve the following energy minimization deformation functional using Gauss–Newton algorithm:

$$\arg\min_v G(V) = \frac{1}{2}\sum_{v_i \in V}(v_i - F_i(V))^2 \tag{6}$$

The computation of Mean Value Encoding (MVE)-based deformation is iterative and it takes a maximum of five iterations to achieve reasonable deformations with each iteration taking between 0.01 and 0.03 s. We have set the value of h_i as 0.01. The mesh shape of the 3D template model after applying MVC is given in Figure 14(c), where (a) is the input sketch and (b) indicates the skeleton of the input sketch.

3.5.3. Final Deformation Using Laplacian Mesh Editing

The Laplacian coordinates preserve the local geometry of the mesh after deforming it. The Laplacian coordinates of each vertex are defined as a displacement vector between the average of the neighbor vertices and the actual 3D position of the vertex. Using such coordinates, editing of meshes is highly efficient and fast because Laplacian mesh editing only requires solving a simple linear system [27].

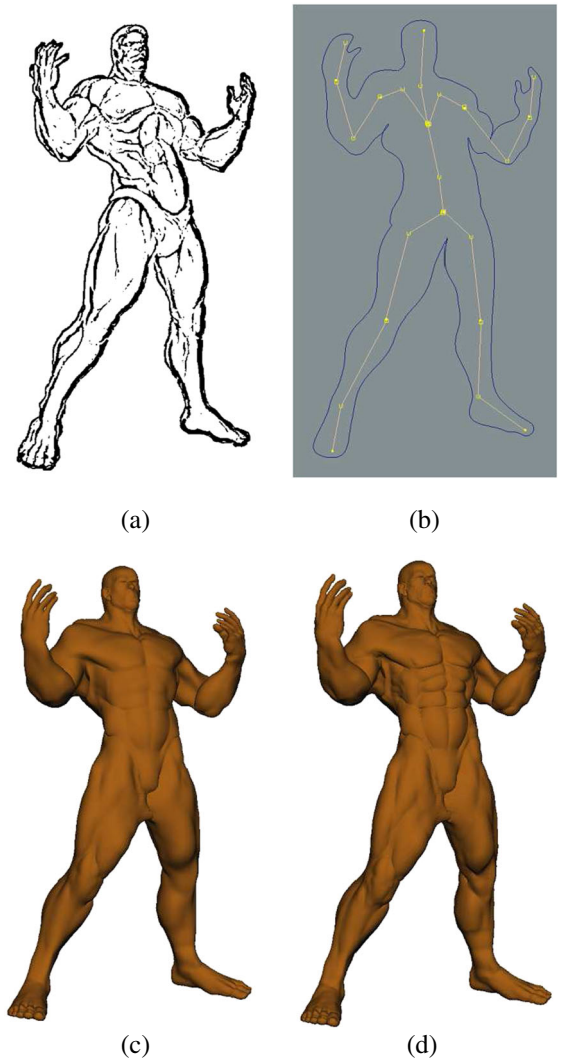


Figure 14. (a) Input sketch, (b) skeleton of the input sketch, (c) final deformed model using mean value coordinates showing exact matching, and (d) final deformed mesh using Laplacian mesh editing showing exact matching.

For each vertex, we represent its differential coordinate by the difference between its position x_i and the average position of its neighbors

$$d_i = L(\chi_i) = \chi_i - \frac{1}{n}\sum_{j\in N_i}\chi_j \tag{7}$$

A mesh can be described by a vector of differential coordinates of all its vertices $D=\{d_i\}$. D can be calculated by multiplying a constant coefficient sparse matrix L (each vertex only interacts with its local neighbors) with a position vector X (a vector of positions of all the vertices $X=\{x_i\}$), that is, $D=LX$.

The selection of curve Points $x_i, i=\{1,\ldots,n\}$ in the region of interest is automatically done and the algorithm sets the target position p_i for each curve point. The mesh is then deformed according to this setup along with preserving its geometry. The mesh shape of the 3D template model after applying Laplacian mesh editing is shown in Figure 14(d).

Laplacian mesh editing will use the sketch points as the handle to deform the mesh.

4. RESULTS AND COMPARISON

In this section, we have shown some results obtained from using MVE and Laplacian Mesh Editing (LME) on a mesh of more than 30,000 polygons (Figure 14). We have also compared the total time durations of computing the final deformed mesh using Kraevoy's method [13] and our approach. Table I shows this comparison.

As mentioned by the authors in [13], the HMM computation takes up to 2 s on a mesh of more than 20,000 polygons to compute correspondences. Our method takes much less than a second to compute the correspondences on a mesh of more than 30,000 polygons. The bottleneck of [13] method is the actual deformation that incurs a total delay of 2–10 min on a full-resolution mesh. Our method on the other hand is considerably faster in deforming the mesh as demonstrated in Table I.

We have demonstrated the difference between the final deformation results obtained from using MVC and Laplacian mesh editing [27] in Figure 15. We found that Laplacian mesh editing preserves the local geometry of the model, while MVC makes the slight re-arranges in the topology and reduces the muscle bulge on the surface. While both algorithms computed final deformation at interactive rates, we found MVC slightly faster in computation than Laplacian mesh editing. Figure 15 shows a comparison of the mesh obtained after applying LME and MVC, respectively.

Figure 16 gives one more result, which deforms a 3D template model in (c) to exactly match the created 2D occluding contours, in (a). The deformed shape of the template model is depicted in Figure 16(b).

5. CONCLUSION & FUTURE WORK

In this paper, we have proposed a new approach and developed an interface to create new character models from user's drawn sketches and a 3D template model. Our proposed new approach combines human vision, intelligence, and interactions with computer's powerful computing

Table I. Performance comparison between our proposed approach and Kraevoy's method [13].

Method/approach	Polygons	Maximum time taken (in ms)
Kraevoy's method [13]	>20,000	600,000
Our approach	>30,000	2000 (correspondence + deformation using mean value coordinate)

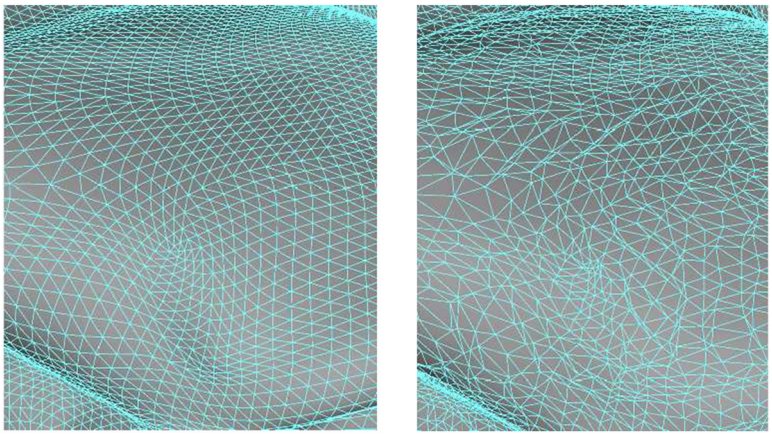


Figure 15. (a) Mesh close-up from LME shows it preserves the mesh topology and (b) mesh close-up from MVE showing re-arranged topology.

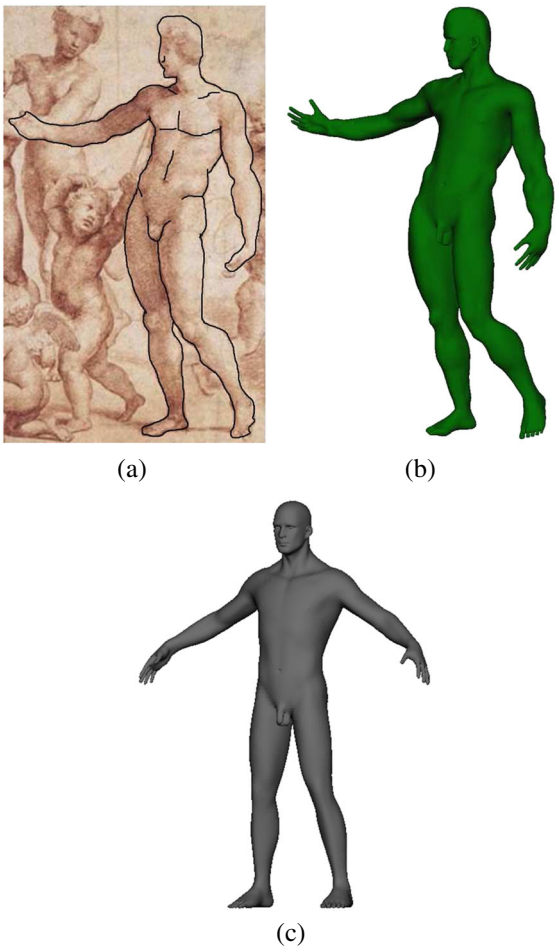


Figure 16. (a) Input sketch, (b) final deformed model using Laplacian mesh editing, (c) template model.

capacity to achieve accurate and quick correspondences between the 3D template model and user's drawn sketches, and presents a hybrid mesh deformation technique to change the 3D template model into new character models efficiently. Our presented hybrid mesh deformation technique maximizes the strength of skeleton based deformation in dealing with large mesh displacements and deformations globally and that of mesh editing algorithms in tackling small mesh deformations locally. The developed user interface integrates all the functions ranging from creating a character dataset, interactive sketch creation, retrieval of a most similar 3D template model, fast and accurate correspondences, and hybrid mesh deformation. We have presented some examples to demonstrate efficiency and quality of our proposed approach.

Our technique has some limitations. First, the current dataset only includes human character models. In future, we intend to evolve it into a large database involving various human and non-human character models. Second, manually drawing occluding contours is slightly difficult for the first time users. This can be avoided by developing an automatic algorithm to generate the occluding contours.

ACKNOWLEDGEMENTS

This research is supported by the grants of 2013 International Exchanges Scheme (Grant no. IE131367), the Royal Society, United Kingdom, and the Sino-UK Higher Education Research Partnership for Ph.D. Studies Project. Xiaogang Jin was supported by the National Natural Science Foundation of China (Grant nos. 61272298, 61472351), China. We would also like to acknowledge partial support from project Dr. Inventor (FP7-ICT-2013.8.1 611383).

REFERENCES

1. Olsen L, Samavati FF, Sousa MC, Jorge JA. Sketch-based modeling: a survey. *Computers & Graphics* 2009; **33**: 85–103.

2. Igarashi T, Matsuoka S, Tanaka H. Teddy: a sketching interface for 3D freeform design. *SIGGRAPH* 1999: 409–416.
3. Bénard P, Hertzmann A, Kass M. Computing smooth surface contours with accurate topology. *ACM Transactions on Graphics* 2014; **33**: 19.
4. Rivers A, Durand F, Igarashi T. 3D modeling with silhouettes. *ACM* 2010; **29**(4): 1–8.
5. Cook MT, Agah A. A survey of sketch-based 3-D modeling techniques. *Interacting with Computers* 2009; **21**: 201–211.
6. Chen T, Zhu Z, Shamir A, Hu S-M, Cohen-Or D. 3-sweep: extracting editable objects from a single photo. *ACM Transactions on Graphics* 2013; **32**: 195.
7. Xu B, Chang W, Sheffer A, Bousseau A, Mccrae J, Singh K. True2form: 3D curve networks from 2D sketches via selective regularization. *ACM Transactions on Graphics* 2014; **33**: 1–13.
8. Bae S-H, Balakrishnan R, Singh K. ILoveSketch: as-natural-as-possible sketching system for creating 3D curve models. In Proceedings of the 21st annual ACM symposium on User interface software and technology, ACM, 2008; 151–160.
9. Buchanan P, Mukundan R, Doggett M. Automatic single-view character model reconstruction. In Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling, ACM, 2013; 5–14.
10. Gingold Y, Igarashi T, Zorin D. Structured annotations for 2D-to-3D modeling. *ACM Transactions on Graphics* 2009; **28**: 148.
11. Lipman Y, Levin D, Cohen-Or D. Green coordinates. In *ACM Transactions on Graphics* 2008; **27**(3): 78. ACM.
12. Shtof A, Agathos A, Gingold Y, Shamir A, Cohen-Or D. Geosemantic snapping for sketch-based modeling. In *Computer Graphics Forum*, Vol. **32**, no. 2pt2. Blackwell Publishing Ltd, 2013; 245–253.
13. Kraevoy V, Sheffer A, van de Panne M. Modeling from contour drawings. In Proceedings of the 6th Eurographics Symposium on Sketch-Based interfaces and Modeling SBIM '09 2009; 37–44.
14. Rabiner LR. A tutorial on hidden Markov models and selected applications inspeech recognition. *Proceedings of the IEEE* 1989; **77**(2): 257–286.
15. FloaterMS. Mean value coordinates. *Computer Aided Geometric Design* 2003; **20**: 19–27.
16. Wuhere S, Shu C. Shape from suggestive contours using 3D priors. In Proceedings of the Ninth Conference on Computer and Robot Vision, IEE 2012; 236–243.
17. DeCarlo D, Finkelstein A, Rusinkiewicz S, Santella A. Suggestive contours for conveying shape. *ACM Transactions on Graphics* 2003; **22**: 848–855.
18. Koenderink JJ. What does the occluding contour tell us about solid shape. *Perception* 1984; **13**: 321–330.
19. Nieto JR, Susín A. Cage based deformations: a survey. In *Deformation Models*, Hidalgo MG, Torres AM, Gómez JV (eds). Lecture Notes in Computational Vision and Biomechanics. Springer: Netherlands, 2013; 75–99.
20. Chen B-Y, Fu-Che W, Lin T-Y, Meng-Chang S. Skeleton constrained dual-resolution modeling for sketch-based deformation. In Proceedings: APSIPA ASC 2009: Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference, 1–7.
21. Eitz M, Richter R, Boubekeur T, Hildebrand K, Alexa M. Sketch-based shape retrieval. *ACM Transactions on Graphics* 2012; **31**: 31.
22. Bentley JL. Multidimensional binary search trees used for associative searching. *Communications of the ACM* 1975; **18**: 509–517.
23. Yu Y, Zhou K, Xu D, *et al.* Mesh editing with Poisson-based gradient field manipulation. In *ACM Transactions on Graphics* 2004; 644–651.
24. Joshi P, Meyer M, DeRose T, Green B, Sanocki T. Harmonic coordinates for character articulation. In *ACM Transactions on Graphics* 2007; **26**: 71.
25. Kraevoy V, Sheffer A, Gotsman C. Mean-value geometry encoding. *International Journal of Shape Modeling* 2006; **12**: 29–46.
26. Ju T, Schaefer S, Warren J. Mean value coordinates for closed triangular meshes. In *ACM Transactions on Graphics* 2005; 561–566.
27. Sorkine O, Cohen-Or D, Lipman Y, Alexa M, Rössl C, Seidel H-P. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* 2004; 175–184.
28. Lewis John P, Cordner M, Fong N. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation." In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press/Addison-Wesley Publishing Co., 2000; 165–172.

SUPPORTING INFORMATION

Supporting information may be found in the online version of this article.

AUTHORS' BIOGRAPHIES

Ismail K. Kazmi is currently a PhD student at National Center for Computer Animation, Bournemouth University, UK. He received his MSc in Software Engineering from University of Glasgow, UK and BSc in Software Engineering from Bahria University, Pakistan. His current research interests include 3D character modeling, sketch-based character modeling, and computational geometry.

DOI: 10.1002/cav

Lihua You is currently an associate professor at the National Centre for Computer Animation, Bournemouth University, UK. He received his MSc degree and PhD degree from Chongqing University, China, and another PhD degree from Bournemouth University, UK. His current research interests are in computer graphics, computer animation, and geometric modeling.

Xiaosong Yang is a senior lecturer in the National Centre for Computer Animation, The Media School, Bournemouth University, UK. He received his bachelor (1993) and master's degree (1996) in Computer Science from ZJU, China, PhD (2000) in computing mechanics from Dalian University of Technology, China. He worked as postdoc (2000–2002) in the Department of Computer Science and Technology of Tsinghua University for 2 years and research assistant (2001–2002) at the 'Virtual Reality, Visualization and Imaging Research Centre' of Chinese University of Hong Kong. His research interests include 3D modeling, animation, real-time rendering, virtual reality, virtual surgery simulation, and computer-aided design.

Xiaogang Jin is a Professor of the State Key Lab of CAD&CG, Zhejiang University. He received his BSc degree in computer science in 1989, MSc and PhD degrees in applied mathematics in 1992 and 1995, all from Zhejiang University. His current research interests include implicit surface computing, special effects simulation, mesh fusion, texture synthesis, crowd animation, cloth animation and facial animation.

Jian J. Zhang is currently a professor of computer graphics at the National Centre for Computer Animation, Bournemouth University, UK and leads the Computer Animation Research Centre. He is also a cofounder of the UK's Centre for Digital Entertainment, funded by the Engineering and Physical Sciences Research Council. His research focuses on a number of topics relating to 3D virtual human modeling, animation, and simulation; including geometric modeling, rigging and skinning, motion synthesis, deformation and physics-based simulation.