

Efficient Sketch-Based Creation of Detailed Character Models through Data-Driven Mesh Deformations

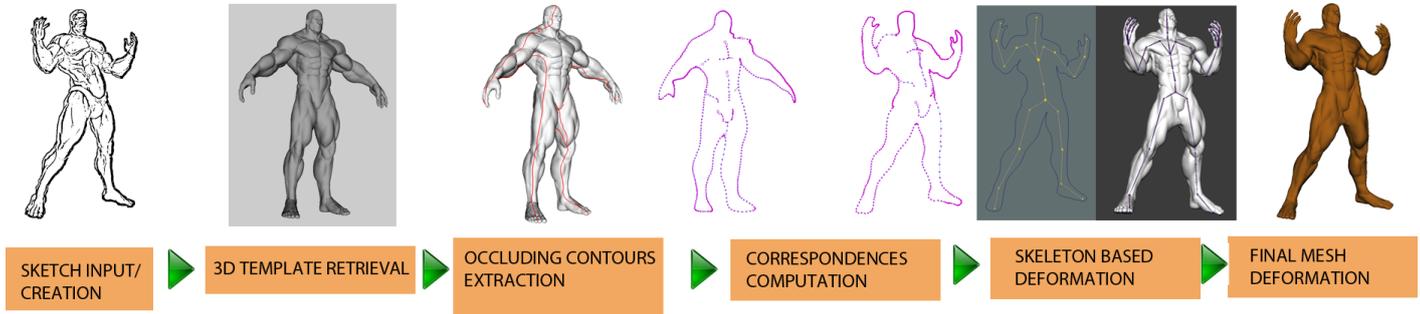


Figure 1: Process flow of our proposed approach.

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 modelling 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, 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, and combining 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.

Keywords: Detailed character creation, sketch based modelling, 3D template models, skeleton based deformation, Mean Value Coordinates, Laplacian Mesh Editing

1. Introduction

Sketch based modelling (SBM) has a great potential in enhancing and improving conventional 3D modelling and

sculpting workflows in the animation industry [15]. 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 modelling in this paper. Direct mesh generation aims at generating geometrical objects from user’s drawn sketches directly such as Teddy by Igarashi et al [27]. Its weakness is that it is not suitable for modeling detailed or complex character models. Template based modelling 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 modelling focuses on highly automatic extraction of occluding contours. Due to complex topology and shapes of character models, the automatic methods lead to incorrect results and have known to produce inconsistent contours [7]. 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 modelling impractical in real-time interactive modelling applications.

In order to tackle the above 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 modelling 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 basic ideas behind our proposed approach. The second basic 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 basic 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 which 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 below. 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 [27] or multiple views [18] sketches. Cook and Agah [14], and Olsen et al [15] have presented thorough surveys on sketch based modeling techniques. Some notable systems for modeling sketch based inorganic models are [16], [17], [18], and [19]. In [17] 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 [16] 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 [18] have proposed a new and simple algorithm towards computing the silhouette cylinders to compute the 3D Constructive Solid Geometry (CSG) by leveraging the special properties of silhouette cylinders. Bae et al [19] 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 artist, and some SBM systems have tried to overcome these challenges but these systems possess several limitations such as [23] and [28]. To overcome these limitations some researchers have proposed template based or data-driven 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 [20]. The area of template based modelling 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 [21] and [1]. Shtof et al [21] have introduced a snapping technique which automatically reshapes and snaps simple 3D geometric primitives to 2D sketch primitives, 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 modelling has been proposed in [1] which involves finding precise correspondences and mesh deformation. For finding one-to-one correspondences, the Hidden Markov Model (HMM) has been harnessed [2] 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 [11]. 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 20K polygons, HMM combined with MVC incurs an overall delay of two to ten minutes. 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 [26]. A powerful system has been developed in [4] 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 [7] has attempted to solve a long occurring problem of finding inconsistent contours using the method presented by [8] and [4]. However as mentioned by the authors, their algorithm is computationally expensive and does not guarantee 100% contour consistency. Due to 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 [5] have discussed and compared several useful deformation algorithms including Mean Value Coordinates, Harmonic functions and Green coordinates in their survey and concluded that Mean Value Coordinates is the best performing algorithm for deformation from the perspective of efficiency. Several techniques exist for Skeleton based deformation. Chen et al [22] 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 [23] have demonstrated a system which 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 above, users drawn sketches can be divided into three different types: silhouette contours, occluding contours, and suggestive contours. A comparison [26] among the three different types of contours is illustrated in Figure 2.



Figure 2: Comparison among a) Silhouette contour, b) Occluding contour, and c) Suggestive contour of a face model. (Image source: Wuhner and Wu [26])

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).

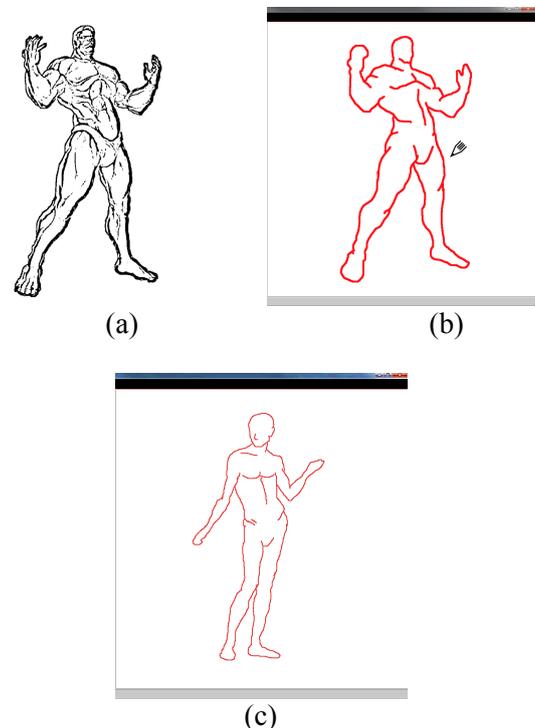


Figure 3: (a) input 2D image (b) user drawn occluding contours with reference to the input image. (c) user drawn occluding contours without using reference image

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 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 25K 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 (NPR) algorithm proposed by DeCarlo et al [4]. It generates suggestive contours and silhouette contours of organic 3D models in real time (Figure 4).

User drawn sketches described can be from any view angle. In order to obtain 2D sketched renders of 3D models in our dataset which 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 towards the centre of the model as shown in Figure 5.

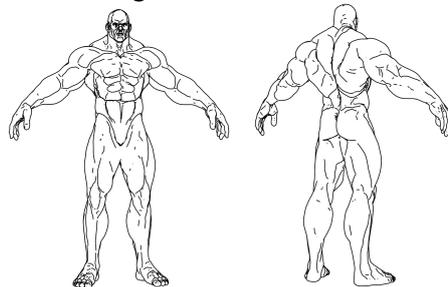


Figure 4: suggestive contours and silhouette contours generated using [4] showing the model as a line drawing.

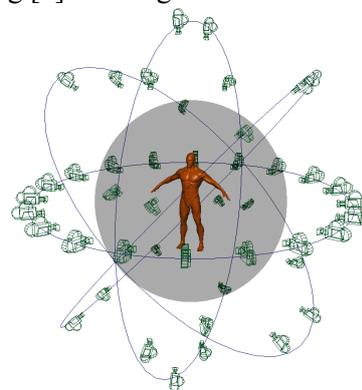
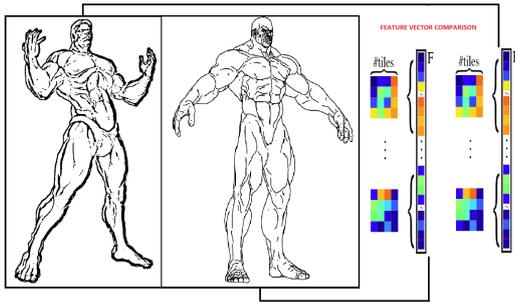


Figure 5: Camera setup for extracting 2D sketched render views of a 3D model in the dataset.

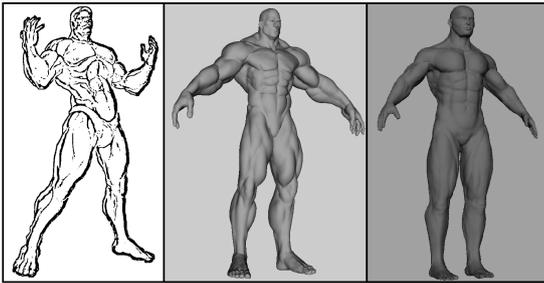
3.2.3 Retrieval of 3D template models

After extracting 2D sketched renders, our system computes the feature descriptor vector of each 2D view. We have drawn inspiration from [6] 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 [6]. 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.

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.



(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 model.

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. Firstly, we investigate how to generate occluding contours of the 3D template model which will be used to provide guidance for the user to sketch the curve on the surface of the template model in this subsection. Secondly, 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 which 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 template model and renders them on the mesh as shown in Figure 8. With the method developed in [4], 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:

$$n(p) \cdot v(p) = 0 \quad (1)$$

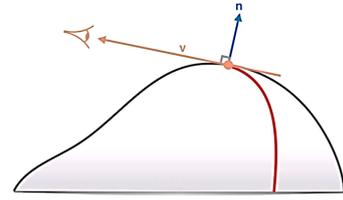


Figure 7: Occluding contour (red curve) on a smooth surface

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), i.e., $-0.3 \leq n(p) \cdot v(p) \leq 0.3$. 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.

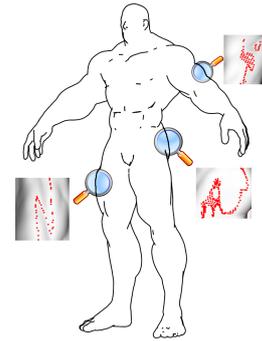


Figure 8: Inconsistent occluding contours magnified on the template model.

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 below.

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 which 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 closest to the points on the B-spline curve which solve the correspondence problem between the extracted 3D template model and the user’s created sketch. These three basic ideas will be elaborated below.

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 to 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 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 which contains fewer points, and then evenly distributes the points in the two curves using the even rule (Figure 9).

$$p_{2i+1}^{j+1} = 1/8(4p_i^j + 4p_{i+1}^j) \quad (2)$$

$$p_{2i}^{j+1} = 1/8(p_{i-1}^j + 6p_i^j + p_{i+1}^j) \quad (3)$$

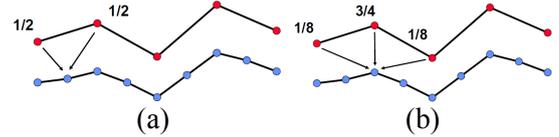


Figure 9: (a) Odd rule inserts new points in the curve, (b) Even rule repositions the old points

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).

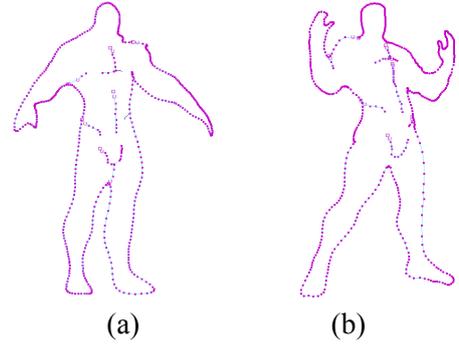


Figure 10: The contours on the left show guided occluding contours that will be corresponded with the input sketch contours on the right.

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 which correspond to the points on created occluding contours. This can be achieved with the following 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. Since 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 neighbour searching. It was first proposed by John Louis Bentley [9] and is designed to handle special data in a simple way. Utilizing KD-tree algorithm makes our approach very fast as compared to the heavy computations performed by the Hidden Markov Model (HMM) in Kraevoy’s method [1]. 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).

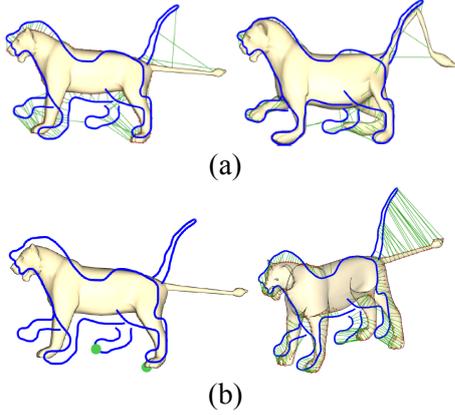


Figure 11: (a) Incorrect correspondence and deformation without specifying anchor vertices. (b) Correct correspondence and deformation with specifying anchor points for correspondences. (Image source: Kraevoy et al [1])

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

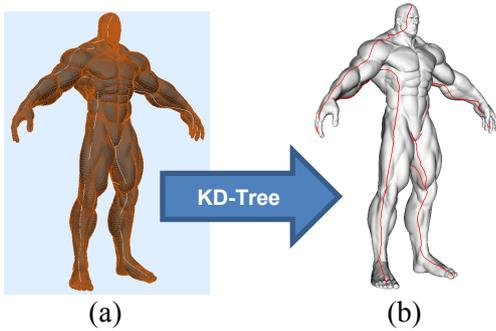


Figure 12: (a) guided occluding contours drawn by the user. (b) Accurate occluding contours achieved by our approach after applying KD-tree algorithm

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 Figures 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 does not exist in our dataset, the pose of the user's created sketches may 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 [1], when using the Mean Value Coordinates, an iterative match-and-deform algorithm is involved. This typically requires a total of ten to twenty outer iterations of matching and deformation to obtain the final result leading to an entire process of two to ten minutes, which makes the algorithm unsuitable in an interactive modelling 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, and 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 [25], Harmonics coordinates [24], Mean Value Coordinates [11-13], and Laplacian Mesh Editing [3]. 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 Mean Value Coordinates (MVC) and Laplace coordinates have been used 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 above, 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 as translation, rotation and scale to align the skeleton of the 3D template model exactly with the skeleton of the created occluding contours, 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) which 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 [29] 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) below. At a particular skeletal configuration c , a deformed vertex V_c , can be computed by

$$V_c = \sum_{i=1}^n w_i M_{i,c} M_{i,d}^{-1} V_d \quad (4)$$

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.

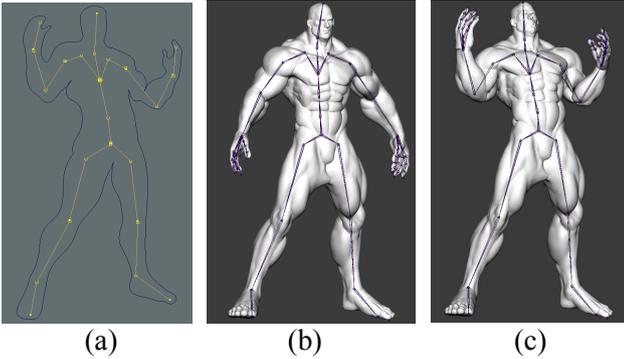


Figure 13: Skeleton based deformation: a) Skeleton of the input sketch, b) Skeleton of the 3D template model, c) The new mesh shape of the 3D template model after geometric transformations

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 [11] to compute the final mesh deformation. For a mesh with vertices V and edges E the mean-value Coordinates for each vertex $v_i \in V$ is computed from the Euclidean coordinates of the vertex and its m neighbour vertices v_j , where $(i, j) \in E$.

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

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 Mean Value coordinates. 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:

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

The computation of MVE based deformation is iterative and it takes a maximum of 5 iterations to achieve reasonable deformations with each iteration taking between 0.01 to 0.03 seconds. We have set the value of h_i as 0.01. The mesh shape of the 3D template model after applying Mean Value Coordinates 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 neighbour vertices and the actual 3D position of the vertex. Using such Coordinates, editing of meshes is highly efficient and fast, since Laplacian Mesh Editing only requires solving a simple linear system [3].

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

$$d_i = L(x_i) = x_i - \frac{1}{n} \sum_{j \in N_i} x_j \quad (7)$$

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 $X = \{x_i\}$), i.e., $D = LX$.

The selection of curve points $x_i, i = \{1, \dots, n\}$ in the region of interest (ROI) 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).

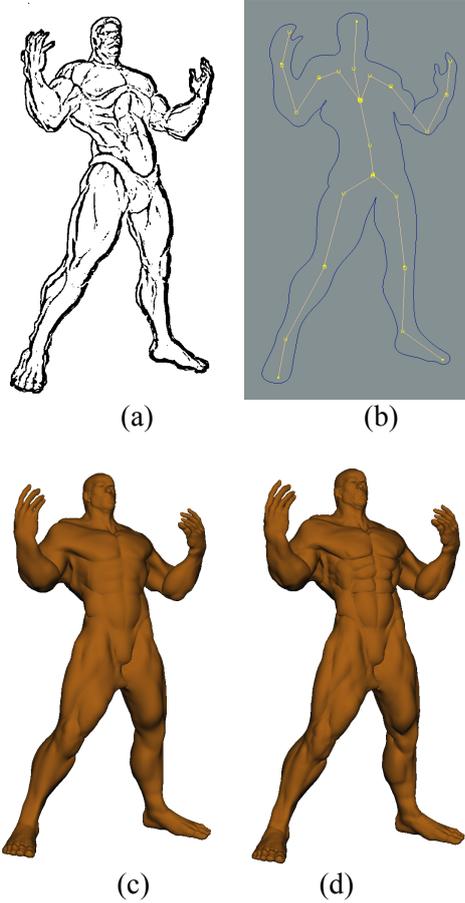


Figure 14: (a) input sketch, (b) skeleton of the input sketch (c) final deformed model using Mean Value Coordinates showing exact matching (d) final deformed mesh using Laplacian Mesh Editing showing exact matching.

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 LME on a mesh of more than 30K polygons (Figure 14). We have also compared the total time durations of computing the final deformed mesh using Kraevoy’s method [1] and our approach. Table 1 shows this comparison.

As mentioned by the authors in [1], the HMM computation takes up to two seconds on a mesh of more

than 20K polygons to compute correspondences. Our method takes much less than a second to compute the correspondences on a mesh of more than 30K polygons. The bottleneck of [1] method is the actual deformation which incurs a total delay of two to ten minutes on a full-resolution mesh. Our method on the other hand is considerably faster in deforming the mesh as demonstrated in Table 1.

Method/Approach	Polygons	Maximum time taken (in milliseconds)
Kraevoy’s method [1]	>20K	600000
Our approach	>30K	2000 (correspondence+ deformation using MVC)

Table 1: Performance comparison between our proposed approach and Kraevoy’s method [1]

We have demonstrated the difference between the final deformation results obtained from using Mean Value Coordinates and Laplacian Mesh Editing [3] in Figure 15. We found that Laplacian Mesh Editing preserves the local geometry of the model while Mean Value Coordinates makes the slightly re-arranges the topology and reduces the muscle bulge on the surface. While both algorithms computed final deformation at interactive rates, we found Mean Value Coordinates slightly faster in computation than Laplacian Mesh Editing. Figure 15 shows a comparison of the mesh obtained after applying LME and MVC respectively.

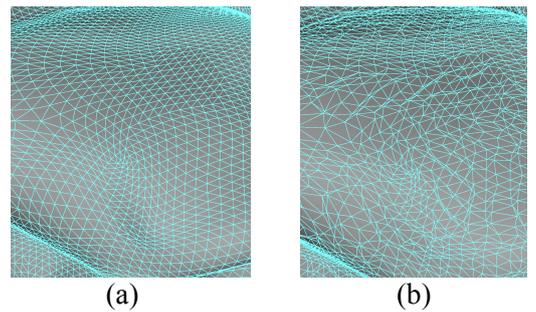


Figure 15: (a) mesh close-up from LME shows it preserves the mesh topology, (b) mesh close-up from MVE showing re-arranged topology

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 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, hybrid mesh deformation. We have presented some examples to demonstrate efficiency and quality of our proposed approach.

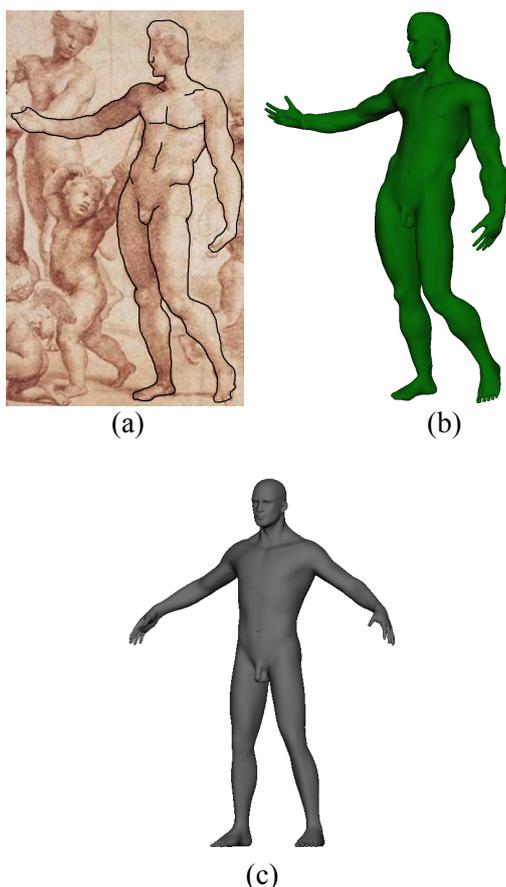


Figure 16: (a) input sketch, (b) final deformed model using Laplacian Mesh Editing (c) template model

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

(to be provided after review)

References

- [1] V. Kraevoy, A. Sheffer, and M. van de Panne, "Modeling from contour drawings," in Proceedings of the 6th Eurographics Symposium on Sketch-Based interfaces and Modeling, 2009, pp. 37-44.
- [2] Rabiner L. R.: A tutorial on hidden Markov models and selected applications inspeech recognition. Proceedings of the IEEE 77, 2 (1989), 257–286.
- [3] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel, "Laplacian surface editing," in Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, 2004, pp. 175-184.
- [4] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella, "Suggestive contours for conveying shape," ACM Transactions on Graphics (TOG), vol. 22, pp. 848-855, 2003.
- [5] J. R. Nieto and A. Susin, "Cage based deformations: a survey," in Deformation Models, ed: Springer, 2013, pp. 75-99.
- [6] M. Eitz, R. Richter, T. Boubekeur, K. Hildebrand, and M. Alexa, "Sketch-based shape retrieval," ACM Transactions on Graphics (TOG), vol. 31, p. 31, 2012.
- [7] P. Bénard, A. Hertzmann, and M. Kass, "Computing smooth surface contours with accurate topology," ACM Transactions on Graphics (TOG), vol. 33, p. 19, 2014.
- [8] J. J. Koenderink, "What does the occluding contour tell us about solid shape," Perception, vol. 13, pp. 321-330, 1984.
- [9] J. L. Bentley, "Multidimensional binary search trees used for associative searching," Communications of the ACM, vol. 18, pp. 509-517, 1975.
- [10] A. Mohr and M. Gleicher, "Building efficient, accurate character skins from examples," in ACM Transactions on Graphics (TOG), 2003, pp. 562-568.
- [11] M. S. Floater, "Mean value coordinates," Computer Aided Geometric Design, vol. 20, pp. 19-27, 2003.

- [12] V. Kraevoy, A. Sheffer, and C. Gotsman, "Mean-value geometry Coordinates," *International Journal of Shape Modeling*, vol. 12, pp. 29-46, 2006.
- [13] T. Ju, S. Schaefer, and J. Warren, "Mean value coordinates for closed triangular meshes," in *ACM Transactions on Graphics (TOG)*, 2005, pp. 561-566.
- [14] M. T. Cook and A. Agah, "A survey of sketch-based 3-D modeling techniques," *Interacting with Computers*, vol. 21, pp. 201-211, 2009.
- [15] L. Olsen, F. F. Samavati, M. C. Sousa, and J. A. Jorge, "Sketch-based modeling: A survey," *Computers & Graphics*, vol. 33, pp. 85-103, 2009.
- [16] T. Chen, Z. Zhu, A. Shamir, S.-M. Hu, and D. Cohen-Or, "3-sweep: extracting editable objects from a single photo," *ACM Transactions on Graphics (TOG)*, vol. 32, p. 195, 2013.
- [17] B. Xu, W. Chang, A. Sheffer, A. Bousseau, J. Mccrae, and K. Singh, "True2form: 3d curve networks from 2d sketches via selective regularization," *ACM Transactions on Graphics*, vol. 33, 2014.
- [18] Rivers, Alec, Frédo Durand, and Takeo Igarashi. 3D modeling with silhouettes. Vol. 29, no. 4. ACM, 2010.
- [19] Bae, Seok-Hyung, Ravin Balakrishnan, and Karan Singh. "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*, pp. 151-160. ACM, 2008.
- [20] Lipman, Yaron, David Levin, and Daniel Cohen-Or. "Green coordinates." In *ACM Transactions on Graphics (TOG)*, vol. 27, no. 3, p. 78. ACM, 2008.
- [21] Shtof, Alex, Alexander Agathos, Yotam Gingold, Ariel Shamir, and Daniel Cohen-Or. "Geosemantic Snapping for Sketch-Based Modeling." In *Computer Graphics Forum*, vol. 32, no. 2pt2, pp. 245-253. Blackwell Publishing Ltd, 2013.
- [22] Chen, Bing-Yu, Fu-Che Wu, Tsung-Yi Lin, and Meng-Chang Su. "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*, pp. 1-7.
- [23] Buchanan, Philip, R. Mukundan, and Michael Doggett. "Automatic single-view character model reconstruction." In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*, pp. 5-14. ACM, 2013.
- [24] P. Joshi, M. Meyer, T. DeRose, B. Green, and T. Sanocki, "Harmonic coordinates for character articulation," in *ACM Transactions on Graphics (TOG)*, 2007, p. 71.
- [25] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum, "Mesh editing with poisson-based gradient field manipulation," in *ACM Transactions on Graphics (TOG)*, 2004, pp. 644-651.
- [26] S. Wuhere, and C. Shu, "Shape from suggestive contours using 3D priors." In *Proceedings of the Ninth Conference on Computer and Robot Vision*, pp. 236-243. IEE, 2012.
- [27] Takeo Igarashi, Satoshi Matsuoka, Hidehiko Tanaka: *Teddy: A Sketching Interface for 3D Freeform Design*. SIGGRAPH 1999: 409-416.
- [28] Y. Gingold, T. Igarashi, and D. Zorin, "Structured annotations for 2D-to-3D modeling," *ACM Transactions on Graphics (TOG)*, vol. 28, p. 148, 2009.
- [29] Lewis, John P., Matt Corder, and Nickson Fong. "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*, pp. 165-172. ACM Press/Addison-Wesley Publishing Co., 2000.