SKETCH-BASED CHARACTER PROTOTYPING BY DEFORMATION

MASTER (RESEARCH) THESIS

Li Mo (HT090511Y)

Supervised by

Dr. Golam Ashraf

School of Computing

National University of Singapore

January, 2011

Table of Contents

Abstract

Most 3D character editing tools are complex and non-intuitive. It takes lot of skill and labor from the artists to create even a draft 3D humanoid model. This paper proposes an intuitive 2D sketch-driven drafting tool that allows users to quickly shape and proportion existing 3D models. We leverage on our existing vector shape representation to describe character body-part segments as affine-transformed circle-triangle-square shape blends. This is done for both the input 2D doodle as well as for the extracted point clouds from 3D library mesh. The simplified body part vector shapes help describe the relative deformation between the source (3D library mesh) and the target (2D frontal sketch). We design and implement two different approaches to achieve the actual deformation and compare the results. One approach is based on automatically setup Free Form Deformation cages, while the other is via shape-based analysis and geometry warping of corresponding body parts. To perform body-part shape analysis, we first segment the mesh with Baran and Popovic's algorithm for automatic fitting of an input skeleton to a given 3D mesh, followed by our existing 2D shape vector fitting process. There are several promising character design applications of this paper; e.g. accelerated personality pre-visualization in movie production houses, intuitive customization of avatars in games and interactive media, and procedural character generation.

Keywords: Deformation, sketch interface, vector art, 3D humanoid character

1. Introduction

While designing a humanoid character, artists typically use shape, size, pose and proportion as the first design layer to express role, physicality and personality traits of a character. The establishment of these traits in character design is one of the most important factors in the process of successful storytelling in any animated feature. Recent advancement in digital multimedia technologies has triggered widespread creation of aesthetic digital character art in the form of videos and images with textual labels or descriptions. But the process of creating humanoid characters with aesthetics matching the desired art style, role, physicality or personality traits still requires tedious labor and can only be done by experienced artists. A rapid visualization tool can be quite valuable in facilitating the character design brainstorming process by allowing the artists to prototype their rough ideas before spending the effort with the actual implementation of the detail 3D character design. From several shape-proportion guides in art and psychology literature [31, 32, 33], we find that typically artists use primitive shaped body parts, skeletons and motion arcs to draft characters. Promising creations are then layered with more details like color, attire, facial expression, and accessories. We take inspiration from this workflow to drive 3D character deformation with a sketch-like interface. The input doodle is constructed as a sum of coarsely sketched body part shapes. Each body part is estimated by the system as a combination of circle-triangle-square primitives. This also motivates us to de-construct existing 3D meshes into similar body-part primitive vectors, and thus implement consistent deformation of 3D characters in response to the sketches. In this paper, we describe the relevant details that allow shape vector deconstruction of 3D meshes, shape fitting of input strokes, automatic construction of Free Form Deformation (FFD) lattices, as well as shape-based geometry warping to implement the visualization and prototyping pipeline.

In this study, we explore both lattice-based and our novel shape-based geometry warping method to implement the deformation; however, we could theoretically use any other method like skeletal or

wire deformers. Our system works under no assumption on the resolution of the given character model. However, we assume a generic humanoid structure, where all models and drawings have similar number of body parts and semantic linkage between different body parts. The proposed system can be summarized as follows:

1. The 2D input character doodle is processed to extract the vector shape information of each body part (Sec. 4).

2. The 3D library character mesh is segmented into different body parts with an existing skeleton fitting algorithm, and then each set of body part vertices are projected and fitted into vector shapes (Sec. 4).

3. FFD lattices are setup around the body parts of the 3D character model according to the vectors extracted in step 2. These lattices are then deformed according to the vectors extracted from the 2D character drawing in step 1, which will in turn deform the 3D model (Sec. 5). Alternatively, the deformation can also be achieved via shape-based geometry warping, where each vertex on the original mesh is analyzed for relevant vector shapes obtained in Step 2 and corresponding influence weights. This information, in conjunction with the $\{s,t\}$ parameterization of the vertex's Cartesian coordinates, is then used to compute the new position for said vertex to achieve the deformation of the character (Sec. 6).

We organize this paper as follows. We first present a literature review of relevant techniques. Next we include a brief description of our existing supporting algorithms on shape representation, fitting and parameterization, for completeness. We then present details on 3D mesh body part segmentation and fitting, automatic FFD lattice construction, and deformation. We continue to illustrate the algorithms to approach the same deformation problem using shape-based analysis and geometry warping. Lastly, we present results using comparison of the both approaches and other popular deformation methods, accompanied with analysis of potentials and limitations.

2. Literature Review

2.1. Spatial Deformation

Spatial deformations are a group of techniques that allow indirect reshaping of geometry by transforming its surrounding space, thus eliminating any restriction imposed by the object's local geometry structure. With the benefit of being computationally efficient and having various abstraction layers for user control, spatial deformations are suitable for a wide range of application scenarios [1]. First introduced by Bechmann in 1998 [2], related techniques can be classified into four categories based on the dimensions of user control manipulators, namely point, curve, surface and volume. Gain et al [1] adopted a similar classification scheme and evaluated the techniques from a user-centric perspective using versatility, ease of use, efficiency and correctness as the evaluation criteria. Although deformation is not directly manipulated by users in our proposed application, the listed evaluation criteria are, however, very relevant since the interactive nature of our system requires the deformation procedure to be efficient, robust and easy for integration.

Volume-based deformation methods employ a lattice of control points that encloses the target object in the 3D space. The most important technique that falls under this category is the Free-Form Deformation (FFD) method first introduced by Sederberg et al [3], where the displacement of a cage control-point influences the entire space inside the lattice. This method benefits the users with a way of performing model-free spatial transformation with a high level of control for deforming detailed geometry. While the complex control lattices provide precise control over the resulting deformation, specifying and editing mesh deformations is an unintuitive and time consuming process. In addition, the method uses cuboid shaped initial lattice as the control volume to simplify the process of embedding objects within the lattice, which imposes serious limitation on the deformation boundary. Despite these shortcomings, FFD remains as a very popular approach due to its extremely simplistic mathematics. Griessmair and Purgathofer [4] extended this technique to employ a trivariate B-spline basis so that, unlike the original FFD method, local control is possible

when deformation is being specified. Though these methods are simple, efficient and popular in use, they still suffer from the drawback of a restrictive original volume shape. Parallelepiped volumes rarely bear any visual correlation to the objects they deform and typically have a globally uniform lattice point structure that is larger than is required for the deformations to which they are applied. Extended Free-Form Deformation (eFFD) [5] is an improvement as it allows noncuboid user-specified base shapes via vertex level editing. However, lattice creation is tedious and specifying deformation remains a time consuming process, rendering the technique as inefficient and difficult to use [1]. MacCracken and Joy [6] used a volume equivalent of the Catmull-Clark subdivision scheme for surfaces to iteratively define a volume of space based on a control point structure of arbitrary topology. This is a significant step in increasing the admissible set of control lattice shapes. The technique is powerful and its only real shortcoming is the potential continuity problems of the mapping function (a combination of subdivision and interpolation) of points within the volume. The approach also suffers from the same discontinuity problems as Catmull-Clark surfaces at extraordinary vertices in the surface-oriented FFD approach proposed by Singh et al [7] which builds a low resolution polygonal mesh near the high-level detailed skin to deform the nearby space. Despite the previously mentioned drawback on continuity, this method has the virtue of offering a similar type of control that one gets from high order surfaces (NRUBS subdivision surfaces) without any topological constraints.

Curve-based deformation techniques approach the problem by using axial or parametric curves to deform geometry. A notable example is the Wires technique [8] where an object is bound to a set of wires that act as manipulators for deforming the object, thus providing an additional layer of abstraction that hides the actual geometry complexity from users. Similar to the workflow in the actual sculpture art creation process, the Wires deformation method provides a natural means of capturing the structure of surfaces, which proved to be particularly suitable for applications that perform large scale deformation via surface editing [9]. As classified as a type of curve-based deformation technique from a user's perspective [1], skeletal deformation or character skinning

works by embedding a skeleton structure within a character mesh and binding each vertex to an appropriate set of joints based on anatomy knowledge of that particular character. Each joint transforms its attached vertexes as if they were rigidly binded to the joint. The deformed position for mesh vertices are a weighted result averaged over all of the joints that the vertex is attached to. While this technique offers an intuitive approach to achieve physically plausible deformation to essentially create various postures of the same character, it suffers from unrealistic volume loss at joints, which causes artifacts such as “collapsing elbow” and “candy wrapper” effects.

Laplacian deformation allows user-specified tweaks to one or a few points on the deformable surface, to be smoothly propagated to the vicinity. The tweaks are treated as hard constraints and the aim is to find an optimal deformation to satisfy them [10, 11]. Igarashi et al [11] first proposed an interactive system that lets user deform a two-dimensional shape using a variant of constrained Laplacian deformation. Laplacian deformation is good for quickly resizing a given part with a few vertex-edits, but it is still fairly tedious to control the body part shape.

2.2. Sketch-Based Modeling

Schmidt et al [12] explain the importance of the scaffolding technique in their review of sketching and inking techniques used by artists. In this method, artists construct characters from basic blocks representing different body parts. Our paper addresses this need for rapid abstraction of these basic blocks from rough strokes. Thorne et al [13] proposed the concept of sketching for character animation, but do not include shape modeling. Orzan et al [14] propose "Diffusion Curve" primitives for the creation of soft color-gradients from input strokes, along with an image analysis method to automatically extract Diffusion Curves from photographs. Schmidt et al [15] propose “ShapeShop”, a 3D sketch authoring system generating implicit surfaces, with non-linear editing via a construction history tree. Although these curve-based methods are intuitive, they require a fair amount of detailing. Thus they are inappropriate for rapid drafting. Our primitive blocks are a lossy abstraction of detailed convex shapes, and thus are easier to represent, construct and perceive.

Due to the intuitive and interactive nature of sketching, there has been substantial interest in adopting sketch interaction in systems for improved usability. Igarashi et al [16] proposed a 3D freeform design environment called Teddy that allows users to create and edit objects by sketching strokes. Hua et al [17] also developed a sketch-based user interface for Scalar-field based Free-Form Deformation (SFFD) technique, where scalar field embedded in 3-D space is manipulated by strokes to specify deformation. Along similar lines, Kraevoy et al [18] proposed a framework that creates models from multi-stroke contour drawings by iteratively establishing sketch-mesh correspondence and deforming geometry. A novel contribution is the adoption of hidden Markov model (HMM) as a representation for finding correspondence between the contour drawing and the base mesh geometry. The system is able to produce fairly decent results across a wide range of models at the expense of considerable computational time and unpredictable necessity for manual tweaking. The Sketch interface is adopted in our work in a similar spirit to make the process of specifying desired character physique traits efficient and straight forward.

2.3. Shape Signature

Shape representation is a well-studied field because of its tremendous importance in pattern recognition and computer vision [19, 20, 21, 22, 23]. These methods can be classified according to several criteria. The first classification is based on the use of shape boundary points as opposed to the interior of the shape. Another classification can be made according to whether the result is numeric or non-numeric. The scalar transform techniques map the image into an attribute vector description, while the space-domain techniques transform the input image into an alternative spatial domain representation. The third classification can be made on the basis of whether a transformation is information preserving or information losing. There is also an approach called mathematical morphology that is a geometrical based approach for image analysis [19]. It provides a potential tool for extracting geometrical structures and representing shapes in many applications.

Inspired by all these developments and from the fact that primitive shapes like circle, triangle and squares play a central role in human perception we developed the shape descriptor with a scaled/rotated/blended combination of these three primitive shapes [24]. Our descriptor can approximate any convex shape with a mixture of these three primitives. Every arbitrary shape is represented as a vector of height, width, rotation, centroid-position and three weight values for circle, triangle, and rectangle.

2.4. GUI

Exposing mathematical parameters for indirect manipulation via a GUI interface has two major disadvantages. Firstly, there is no intuitive connection between these parameters and the user-desired manipulation. Secondly, deformations defined using the handles of a specific representation cannot be trivially applied to other shape representations or even different instances of the same shape representation [35]. Integrated bone and cage deformation systems avoid potential artifacts that may arise in case of independent localized cages [36].

Our work focuses on creating 3D models of humanoid characters from a rough 2D sketch input from the user. It is trying to solve the character-drafting problem in the same spirit as Sykora et al [25, 26, 27], Gingold et al [28], and Fiore et al [24]. However, none of these works factor in the role of psychology in primitive shape scaffolding of characters. We derive inspiration from the use of primitive shapes outlined in art books [31, 32, 33] as well as shape perception literature [34]. Since primitive shapes like circle, triangle and rectangles play a central role in human perception, our underlying shape abstraction is closer to artists' creative intentions. We have recently proved this computationally through data mining techniques on perception feedback collected implicitly through online character puzzle games [29, 30, 37].

3. Supporting Algorithms

We briefly describe our prior work on shape representation [30] and parameterization [38] for completeness, as we will develop on it to implement scaffold drawing driven FFD deformation of character meshes.

3.1. Vector Shape Representation

As shown in Fig. 1, we store each of the three normalized primitive shapes as a set of eight quadratic Bezier curves. The solid points represent segment boundaries and the ragged blotches represent mid-segment control points. Note how a null segment (1-2) had to be created for the apex of the triangle. The reason why our piecewise curve segments work so well is that we were able to carefully identify the corresponding segments for the diverse topologies of circle, triangle and square. As a result, even under simple linear interpolation, we do not notice any tears or inconsistent shapes. The normalized shapes can be affine transformed to any location, scale and rotation. Finally, the shape weights are applied to blend the corresponding Bezier control points, to yield an in-between shape. Note that start-end-mid control points of only corresponding segments are interpolated, as shown in Eqns. 1 and 2.

$$p_j = \sum_{i=1}^{3} (w_i . p_{i,j}) \quad \text{.....(1)}$$

$$m_j = \sum_{i=1}^{3} (w_i . m_{i,j}) \quad \text{.....(2)}$$

where $j \in \{1,2,3,4,5,6,7,8\}$ and $\sum_{i=1}^{3} w_i = 1$.

In the above equations, p'_j and m'_j represent the *j-th* blended segment boundary and midpoints respectively, while $p_{i,j}$, and $m_{i,j}$ represent the corresponding control points in the *i-th* primitive shape (square, triangle,). w_i is the weight contribution from the *i-th* primitive shape.

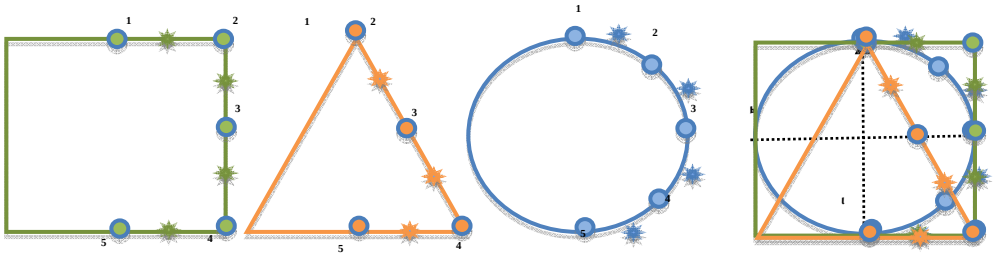

Fig. 1: Consistent interpolation of circle, triangle, and square [30].

Results of some blend operations are shown in Fig. 2. The cross hairs under the shapes indicate the shape weights. With this background information about our primitive representation, we are now ready to describe vector fitting of stroked body-part line drawings. We assume that the input shapes are roughly symmetric about their medial axis, and generally convex.

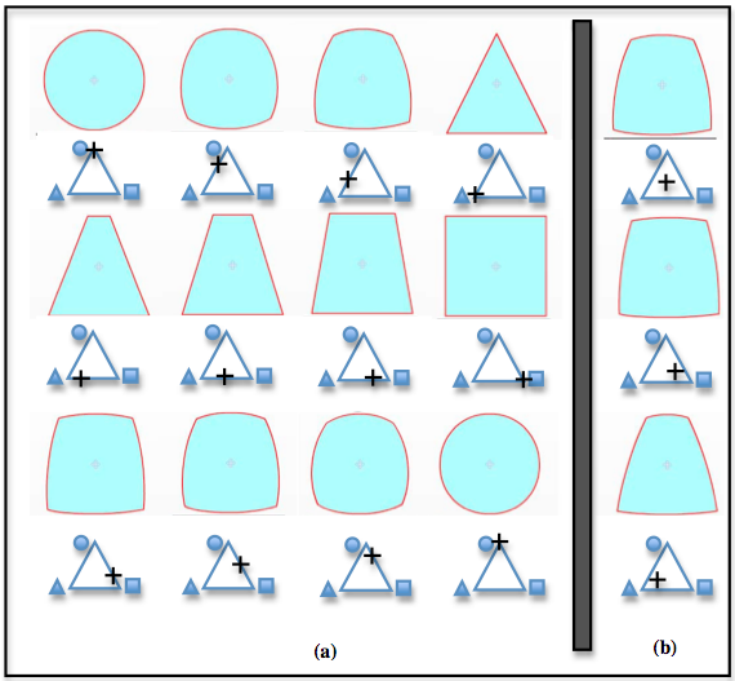

Fig. 2: Blended shapes after consistent interpolation (shape weights indicated by cursor positions) [30]

3.2. Vector Fitting

A closed input stroke can be treated as a set of connected points, where the first and last points are fairly close to each other. We first resample the stroke at fixed angular intervals about the centroid of the input points. This helps avoid any bias due to variances in stylus pressure and stroke timing. A standard projection variance maximization algorithm, commonly employed to compute Oriented Bounding Boxes, is used to find the medial axis. In this algorithm, a ray is cast through the centroid, then all the boundary points are projected onto the ray, and the variance of the projected point distances from the centroid is noted. The ray that produces maximum variance is estimated to be the medial axis. Once the medial axis is noted, the axial-length and lateral-breadth of the shape can be easily calculated. We then perform a normalization affine transform to align the input shape to the Y-axis and scale it into a unit square. This simplifies shape error checking while ensuring rotation/translation/scale invariance during the fitting process. Lastly, we compute the best primitive shape combination, by minimizing boundary distance errors between our template shape combinations and the input points. In practice, this is a simple 2-level for-loop, incrementing shape weights by a fixed small value, and measuring the accumulated shape error. The shape error is calculated by accumulating slice-width errors over 40 lateral segments (along the medial axis). We have achieved decent fitting results for most cases. However, there are some cases where shapes computed with boundary distance errors do not match with human perception. We are currently working to improve the qualitative results through a perception regression model.

3.3. Space Parameterization

As shown in Fig. 3, we use a tuple $\{s,t\}$ for parameterizing the cage and correctly positioning corresponding lattice points in the source (mesh) and target (sketch) FFD lattices for each body part. Parameter t is a floating point number whose integral part holds the Bezier segment number of the curve and parameter s is the measurement of distance along the line joining the center of a cage and the point on the Bezier-segment-curve. We include the following details for completeness.

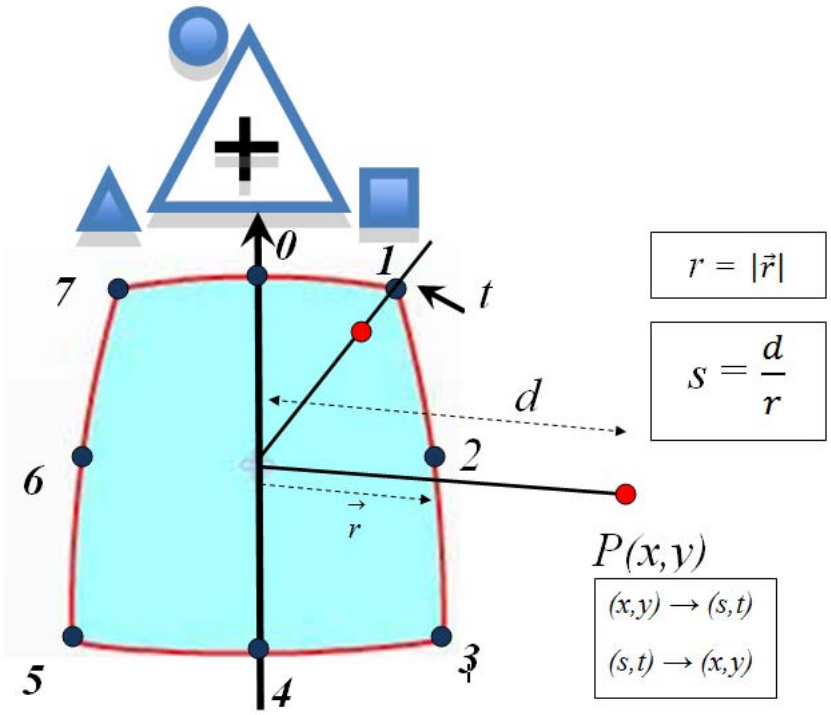

Fig. 3: Polar Coordinate parameterization of a cage [38]

To elaborate, we parameterize a cage with $\{s,t\}$ polar coordinates [29], where s represents a scaled distance from the center of the primitive, and t represents the curve segment parameter, where the extended direction vector $\vec{s}$ intersects the boundary as shown in Fig. 3. To avoid repeated curve intersection calculations, we cache the t parameter as well as the center-to-boundary distance r (360 rows @ 1° increments) for every cage in the scene graph. It is thus quite efficient to convert between Cartesian pixel coordinates $\{x,y\}$ and polar shape coordinates $\{s,t\}$. To convert a pixel position into polar coordinates we first lookup the closest t parameter with an angle index. The angle is computed as a dot product between the relative vector from the cage center to the pixel, and the medial axis of the cage. The parameter s is then computed as the pixel distance to the cage center, divided by the r distance cached along with the t parameter. Similarly, the reverse transformation from polar to Cartesian proceeds with scaling of the direction vector returned by the boundary position computed from t (using $\mathrm{d} = \mathrm{s}\vec{r}$).

Further to the above polar coordinate parameterization, performing point inclusion and cage overlap tests become very easy. A Cartesian point p is considered to be inside a cage, if its polar coordinate $s \leq 1$.

4. Vector Segmentation of 3D Mesh

Similar to the 2D drawings, the 3D character model needs to be analyzed for body part shape vector extraction. The details of the process are as follows:

a) For each vertex on the mesh, its body part membership information is computed, which specifies the body part this vertex belongs to. To accomplish this, a standard humanoid skeleton is created to fit the humanoid mesh using Baran and Popovic's automatic skeleton fitting algorithm [39]. Their skinning algorithm returns a set of influence weights and active bone indices for every vertex. We use this information to partition the vertices into body part sets, using influence weight thresholds and identity of the most influential bone. Since the segmentation algorithm uses many iterative calculations to search for the optimal skeleton (matching the input skeleton structure), this step is performed offline on all the 3D meshes in the library, to allow for efficient deformation during the sketching process.

b) All the vertices that belong to the same body part are then grouped and projected onto the XY plane as both the source 2D character drawings and the 3D character model are posed in the front profile. The convex hull for each of these groups is then computed, giving the exact contour for that body part in the front profile. To extract the shape vectors of the convex hulls, 2D points are sampled at regular intervals along hull outline, and then fed into our vector fitting routine to generate the corresponding shape vectors.

5. Lattice-based Character Mesh Deformation

Character deformation is achieved by first segmenting the source (sketched body parts) and target (3D mesh) figures, and then, generating shape vectors from them. Since the sketch consists of a set of body part outlines, the segmentation is simply the process of auto-identifying the body parts using a set of heuristics similar to [13]; e.g. head appears top-most, under which appears neck and/or torso, etc. The overall sketch driven mesh deformation idea is illustrated in Fig. 5. Using the template of a skinny girl image, a rough sketch of body parts (omitted in Fig. 5a for clarity) is fed into the system, which then deforms a pre-segmented mesh from the 3D library. The process starts with 2D character drawings being processed (Fig. 5a), to extract shape vectors for each individual body part (Fig. 5b), using the vector fitting technique in Sec. 3.2. Its corresponding 2D FFD lattice deformer setup (Fig. 5c), serves as the deformation target. Similarly, automatic body part shape analysis is performed on the source character model (Fig. 5d), to produce a set of vectors (Fig. 5e) corresponding to those of the 2D drawings. A set of FFD lattice deformers (Fig. 5f) can then be constructed from these vectors, which completes the process by deforming the source model (from Fig. 5d to Fig. 5g).

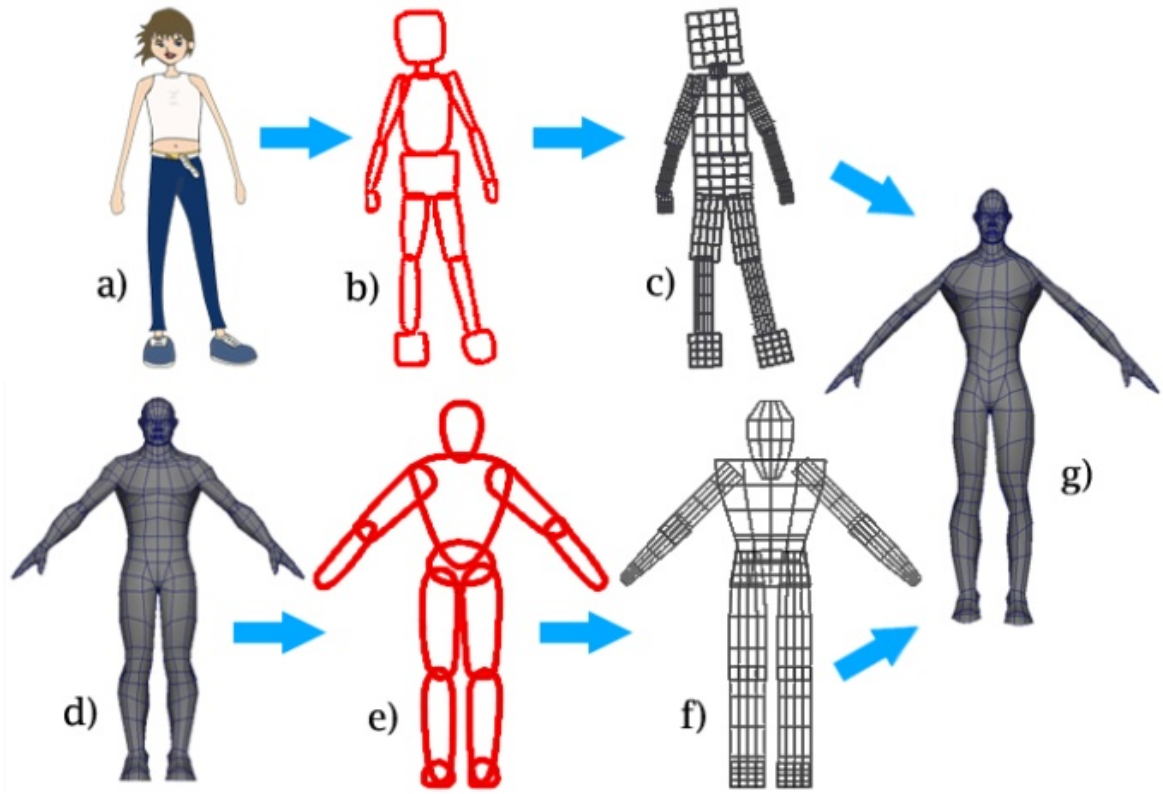

Fig. 5: Deformation pipeline. a) 2D input drawing/sketch. b) Shape vectors of 2D drawing. c) Full body lattice construction for 2D drawing. d) Source 3D character model. e) Shape vectors of the 3D model body parts. f) Full body lattice construction for 3D model. g) Final deformed 3D model.

Lattice deformation proceeds in a standard manner, as described by Sederberg and Parry [3], once the source and target lattices are set up from the sketch and 3D un-deformed model, respectively. In practice, arbitrary topology FFD [5, 6] yields better results than the original parallelepiped deformer base configuration in [3].

The rest of this section explains the automatic full body FFD lattice system construction from step (e) to (f) in Fig. 5. Given a shape vector, v, a lattice, l, needs to be created such that its shape and affine transformation match that of the body part front profile. This ensures the subsequent lattice deformation is accurate. The following steps illustrate the details of the algorithm in the case of a 5×5×2 3D lattice deformer. However, it should be noted that the same algorithm also applies to any lattice subdivision configuration, though this particular configuration proves to be capable of

producing satisfactory deformation results without adding much complexity to the real-time deformation calculations. The steps are as follows:

a) A unit sized 5×5×2 3D lattice deformer, L, is created at the world origin. Before any global affine transformation is applied to match L to its corresponding body part in terms of rotation, scaling/size and position, L is deformed into a linear combination of the three primitive shapes according to the weights indicated in the body part vector, V. As shown in Fig. 6, for every boundary lattice control point (P_i) along the outline of L (in clockwise direction), three position values are calculated: i) square with shape weights (1, 0, 0); ii) triangle with shape weights (0, 1, 0); iii) circle with shape weights (0, 0, 1). Denoted by S_i, T_i and C_i, these values represent the corresponding positions of lattice control point, P_i, on the respective primitive shape. The final interpolated position P_i is then given by a linear combination of the position values: $P_i = S_i v_s + T_i v_t + C_i v_c$, where (v_s, v_t, v_c) denote the shape weights of vector shape V. Based on the number of sub-divisions, we can easily assign regular {s, t} intervals (see Sec. 3.3) to the lattice points, and accurately extract Cartesian coordinates for both source and target FFD cages.

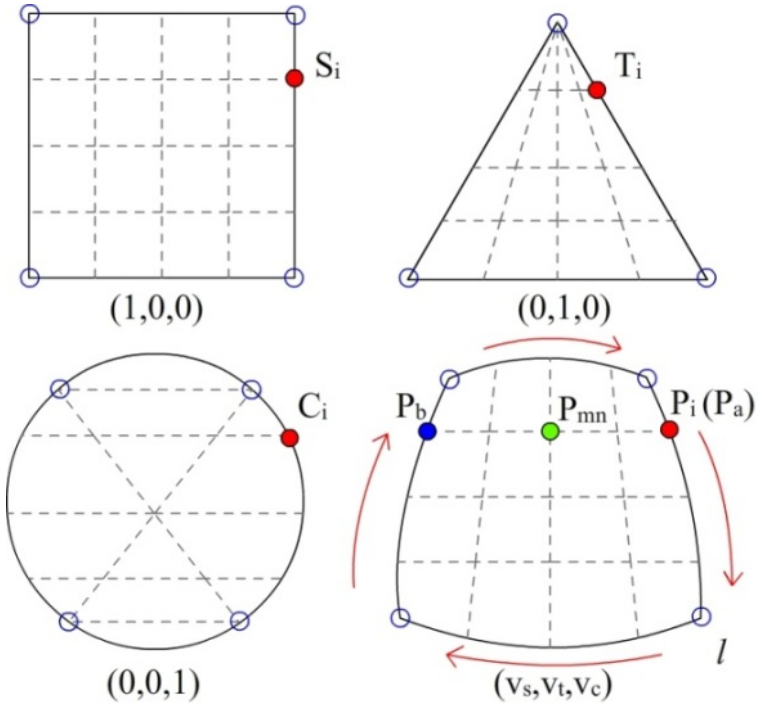

Fig. 6: Computing lattice control points from shape vectors

b) The positions for the internal lattice control points, P_{mn}, are computed by interpolating the positions of the boundary control points, P_i. We traverse through these points in top→down and left→right order. Each internal lattice point is computed as a distance-weighted sum of the two boundary lattice points, P_a and P_b, on the same lattice row m, as shown in Fig. 6.

c) With the shape defined, L is now scaled along X and Y-axis, rotated and finally translated according to V to complete the construction of a lattice deformer, L. In order to prevent unwanted distortion to the geometry of the source model, L is set to influence the mesh geometry only after the entire construction process is completed.

d) Finally, the depth of L is set to be a fixed value, which should exceed the girth of the model along the Z direction. Since our system concentrates on the front profile of the prototyping process, the exact value of this parameter is not that significant.

6. Shape-based Character Mesh Deformation

In order to deform a base character mesh to a target character, we adopt the aforementioned shape vector scheme to extract and represent the shapes of both the base mesh and the target mesh (Sec. 3.1 and 3.2). The preparation for shape vectors is done on a per body part basis, which means exactly one vector corresponds to a body part (Sec. 4). The actual deformation takes place in the $\{s,t\}$ parameterization space local to each individual body part where the shape vector represents the cage for the parameterization (Sec. 3.3). The $\{s,t\}$ space essentially establishes a common ground to allow the relative spatial relationship among vertices to be preserved during the deformation. Finally, before the model is deformed, procedural constraints are defined for all body parts so that parts affect each other, therefore their vertices, in an anatomically correct manner. The following sections contain further details on each of the above aspects.

6.1. Geometry Warping

Geometry warping is achieved essentially through the translation of every existing vertex to their new position to form a differently shaped character as a whole. In general, each vertex needs to go through a series of steps as illustrated in Fig. 7. First, we identify the body parts on the base character model that influence this vertex (a). For example, the vertex in interest, denoted by v in the figure, is anatomically influenced by body part c_1 and c_2, but not c_4 despite the fact that v is located not far from c_4. The corresponding target cages, denoted by c_1' and c_2', respectively, are then used to compute the new location for vertex v (b). As the method for computing the new Cartesian coordinate preserves the relative spatial relationship between v and its influencing cages, the new coordinates, denoted by v_1 and v_2, are expected to differ from each other as they each represents where the new vertex should be located with respect to c_1' and c_2', respectively. As a final step, we address this difference by blending the results using the influencing weights from the involved body

parts to compute v_3, which is then used as the final Cartesian position for v after the deformation (c). The followings contain further details regarding each of the steps involved in the geometry warping process.

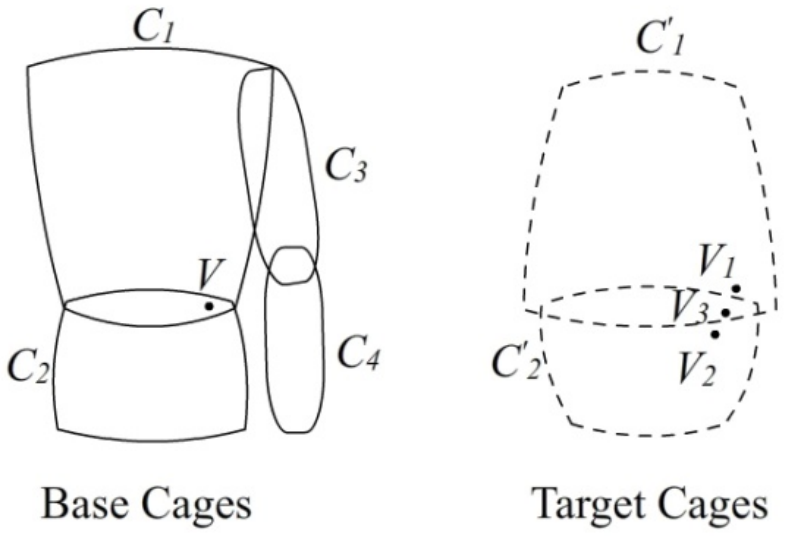

Fig. 7: Geometry Warping

a) *Determine influencing cages*: To determine the influencing body parts for a given vertex, we compute the $\{s,t\}$ coordinates of the vertex with respect to each of the 16 body parts of the original model. If the value of s does not exceed 1, i.e. the vertex is inside the current cage, this $\{s,t\}$ pair will be then used to look up heuristics-based rules for exactly which other cage(s) it should be influenced. The corresponding influencing weights are then calculated procedurally. Without breaking the flow, the details on the definition of the constraints and the computation of weights are elaborated in the next section (Sec. 6.2).

b) *Compute new Cartesian coordinate*: Given a vertex, v, cage c and the target cage c', we use $\{s,t\}$ space as the common channel to preserve the locality of v while computing its new Cartesian coordinate, denoted by v'. Specifically, we first convert v into $\{s,t\}$ coordinates with respect to c according to the definition of $\{s,t\}$ parameterization explained in Sec. 3.3. We then use this $\{s,t\}$ value to convert the position back into the Cartesian space with respect to c' by performing a simple

reverse calculation which essentially interpolates a point location along a quadratic Bézier curve segment. Since the reverse computation depends on c', it is conceivable that the obtained Cartesian coordinate is expected to be different for different body parts.

c) *Blend new positions*: As a final step in the warping process, each new Cartesian coordinates obtained from the previous steps are blended based on how much *v* is affected by the corresponding cage. We measure the relevance of a cage based on the {*s,t*} expression of *v* in the cage. As explained previously, *s* is defined to be the ratio that reflects the distance between *v* and the centroid of *c*, where the distance becomes larger as *s* becomes smaller, thus implying a smaller influence over *v* from *c*. Therefore, in the scenario where a point *v* is influenced by n cages, c_1, c_2,…c_n, the weights for c_i, denoted by w_i, can be calculated as shown in Eqn. 3.

$$w_i = \frac{\frac{1}{s_i}}{\frac{1}{s_1} + \frac{1}{s_2} + \cdots + \frac{1}{s_n}} \quad \text{.....(3)}$$

$$\text{where} \quad s_1, s_2, \cdots s_n \neq 0$$

In the case where *v* is only affected by exactly one cage, it is easy to conclude that *w* is equal to 1, reflecting the fact that *v* is under the influence of that cage only.

6.2. Procedural Constraints

From previous description of the geometry warping process, it can be seen that it is crucial to be able to identify the correct influencing cages for a given vertex, *v*, in order to avoid artifacts and achieve optimal deformation results. To ensure that the vertex-cage relationship is defined in an anatomically correct manner, our general approach is to specify non-overlapping regions in {*s,t*} space for each body part, which are called zones. If *v* is located within a particular zone, we can then determine which other body part is related to that specific zone based on heuristics derived from human body structure. As illustrated in Fig. 8, *c* is the cage representing the shape of the upper

body portion of a full body human character. For this particular body part, four zones can be defined such that they correspond to the neck (Zone 0), the left upper arm (Zone 1), the abdomen (Zone 2) and the right upper arm (Zone 3), respectively. Therefore, v_1 is influenced by the left upper arm in addition to the upper body since it is located in Zone 1. Similarly, c is the only influencing cage for v_2 since v_2 is located inside the cage but outside all the defined zones. Complications occur when a vertex does not technically belong inside any of the 16 body part cages. In this case, we find the nearest cage by measuring the value of s calculated against all the cages and choosing the smallest one. As shown in Fig. 8, let us assume that c has been identified as the nearest cage for outside vertices v_3 and v_4. Our approach is to categorize them into the cases of v_1 and v_2 by checking if they belong to extended zones. For example, v_3 is within the coverage of the extended region of Zone 1; therefore, v_3 is treated in the same way as v_1, which maps the left upper arm as its second influencing cage. In contrast, v_4 does not belong to any of the extended zone definitions. Since c is the nearest cage, it is therefore considered to be the only influencing cage in the cage of v_4.

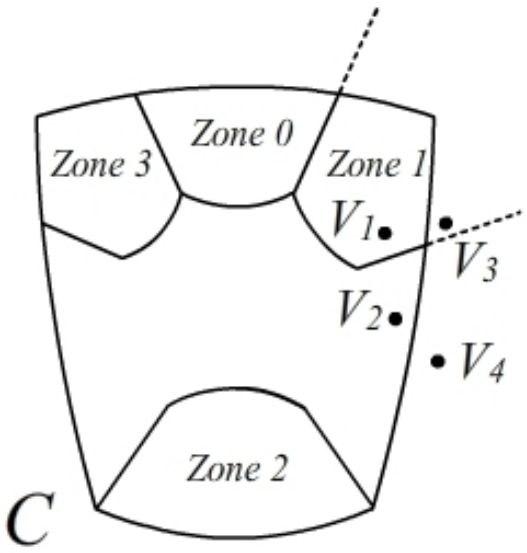

Fig. 8: Zone Definition for Upper Body

Due to the polarized nature of $\{s,t\}$ space, it is very intuitive to express a zone definition using $\{s,t\}$ based constraints. For example, Zone 1 in Fig. 8 can be approximated using the following conditions:

$$\begin{cases} s \in [0.5,1] \\ t \in [7.5,8) \cup [0,0.5] \end{cases}$$

It should be noted that the exact shape of the zone defined this way is different from what is shown in the figure. However, the difference is insignificant and does not seem to have negative impacts on the deformation results. In addition, all the zones within the same cage are defined such that no overlapping between zones occurs. Therefore, for any give vertex, the number of influencing cages won't be greater than two, which allows us to accelerate the process of determining the relevant cages for a given vertex.

Extended zones can be defined in a similar manner by eliminating constraints on s since they are only applicable to outside vertices. The following example defines the extended zone region for Zone 1.

$$t \in [7.5,8) \cup [0,0.5]$$

One unique advantage of segmenting a cage into zones is that it allows us to specify vertex-cage relationship local to a cage, without imposing assumptions on the pose of the character model. By tweaking the zone definitions, we can quickly adapt the system to cater for non-standard character models with body proportions different from standard human structures.

7. Results and Analysis

7.1. Lattice-based Deformation Results

Fig. 9 illustrates completely automatic results of body-part shape analysis on a 2D input sketch (body parts traced over an existing "skinny-girl" stock image), as well as two different (muscular and fat) 3D humanoid meshes. As can be seen, the deformed models inherit the dominant shape traits from the corresponding body parts of the input character drawing while still preserving the smoothness at the joined area between body parts. The deformation input was sketched within 20 seconds, and the FFD mesh deformation result was achieved within 1-2 seconds. The un-optimized 3D mesh segmentation code, *Pinocchio* [24], takes a few minutes on lightweight meshes (1-100K triangles), so we do this as an offline step.

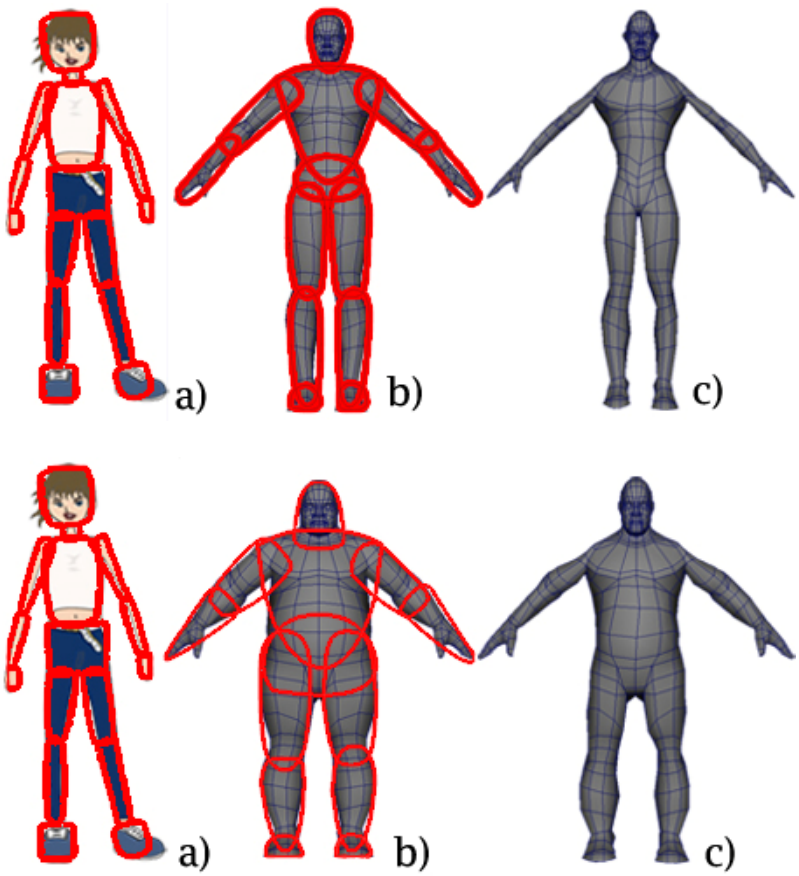

Fig. 9: Lattice-based Mesh deformation results. a) Input 2D sketch with body part vector analysis. b) Source 3D character model with body part vector shape analysis. c) Final deformed 3D model.

There are a few limitations to this approach. Firstly, we notice that the degree of compliance with the source sketch shapes varies with different models. This is expected, as we implement the shape transfer as a relative shape deformation operation, rather than a hard boundary constrained optimization problem. Secondly, foreshortening of the input drawing is inevitable in the general case as our body part shape analysis is currently limited to the front profile only. Lastly, some vertex collapsing artifacts are produced for vertices in overlapping FFD influence regions. As shown in Fig. 10, limited control over overlapping lattice deformers at joint areas like shoulders tend to create geometry artifacts such as shrinking. Such problems can be addressed by setting up better procedural fall-off of influence, as well as controlled smoothing of influence between neighboring FFD lattices.

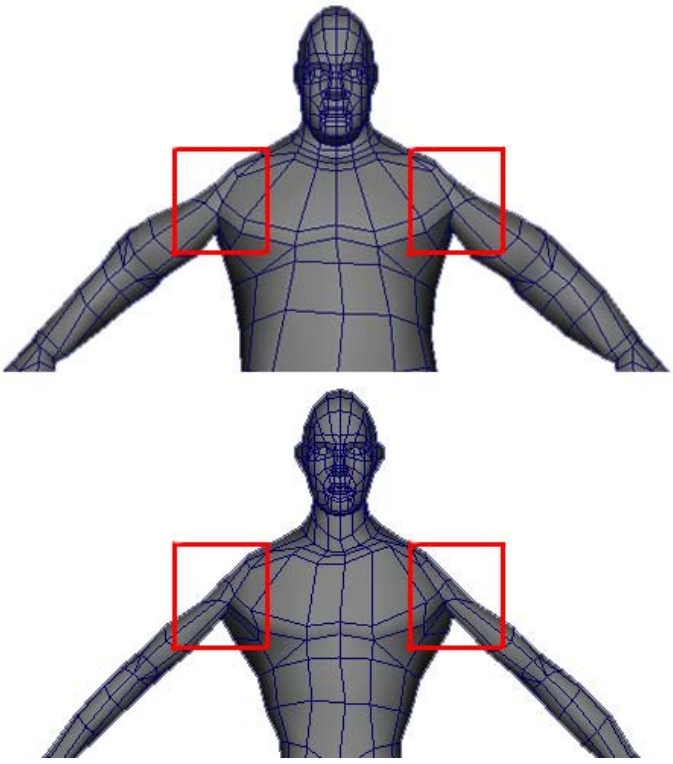

Fig. 10: Deformation artifacts at joints with overlapping lattices

7.2. Shape-based Deformation Results

We use the same generic male model as the one used in the lattice-based deformation study (i.e. Fig 9b) to gather results for the shape-based deformation method. The same shape analysis performed previously is also used in this case. The vectors extracted from the previous 2D input sketch are modified to generate different characters used as targets for deformation.

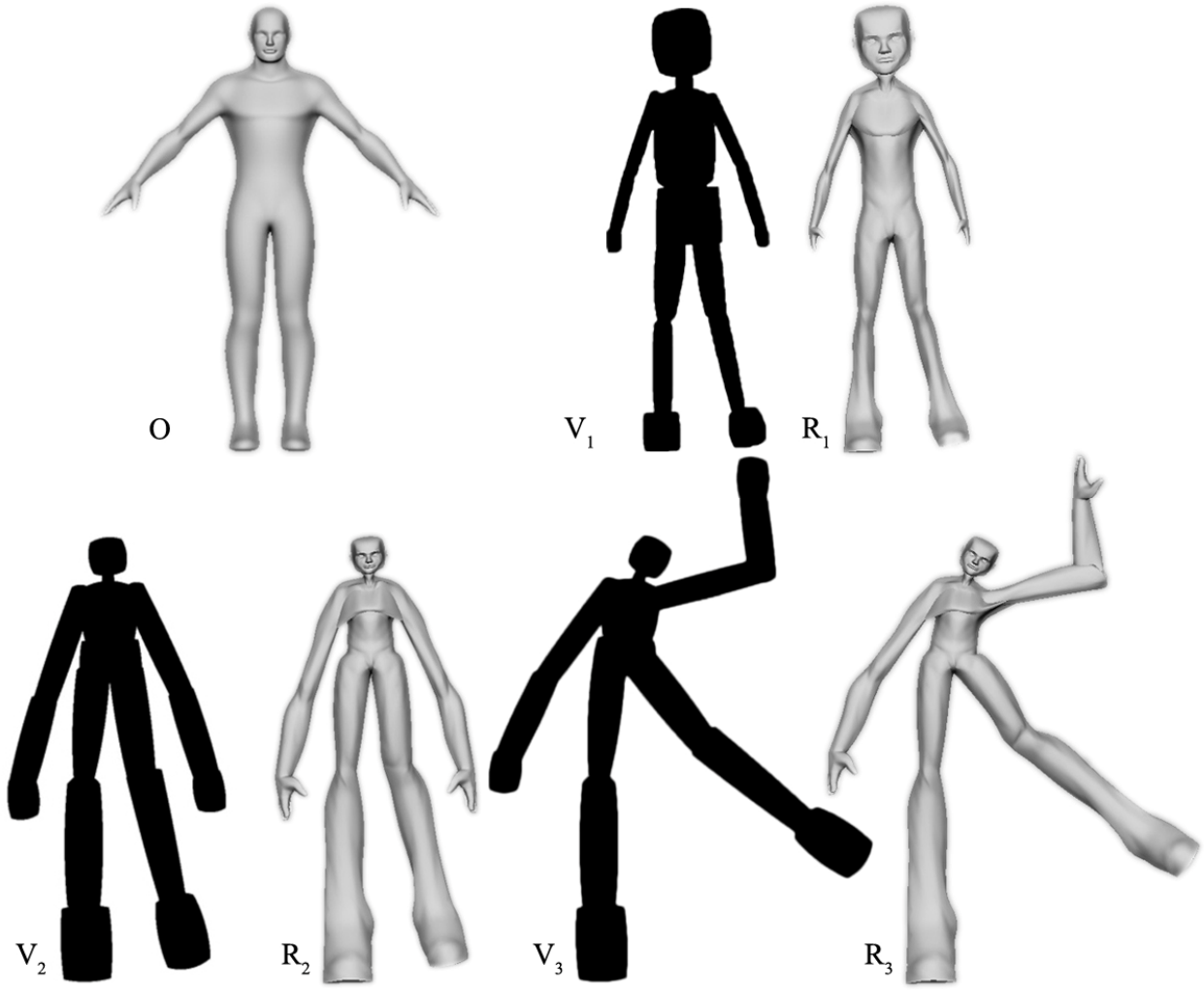

Fig. 11: Shape-based Mesh Deformation Results

Three deformation examples are shown in Fig. 11, where *O* is the original mesh (i.e. Fig. 9b), V_1, V_2 and V_3 are visualizations of the target vectors, and R_1, R_2 and R_3 are the corresponding deformation results. Compared with the original model, V_1 specifies a target character with very different body proportions, body part shapes and pose. Keeping the same pose as V_1, V_2 takes the proportion and

size variations to a more extreme extent (i.e. small head, huge limbs and small torso) while V_3 defines a new pose on top of the V_2 character (i.e. the body is bent to the left with the left arm and the left leg lifted). It can be seen that in all three cases, the resulted mesh follow the shape, the proportion and the pose of the target cages closely while preserving the geometry detail (e.g. face, hands, chest) of the original model. Vertex positions are interpolated properly to retain surface continuity and smoothness. Note that no unwanted geometry artifacts (e.g. tearing, discontinuity, bumps and dents) have been introduced as a result of deformation even at joined areas between body parts. In addition, it can be observed that only intended body parts are affected by the various pose and shape changes despite the fact that these body parts are often geometrically located close to other body parts. For example, the outer border of the upper body is fairly near the upper arm, however, only the upper arm is affected comparing R_2 and R_3, implying a precise deformation process. Finally, the ability to handle difficult poses suggests the possibility of repurposing the proposed mesh deformation method as an alternative to skeleton-based skinning approaches, which is further illustrated in Fig. 12, where the rectangles and the circles draw attention to the corresponding areas on various mesh for comparison. As can be seen from the figure, the skeletal approach is prone to artifacts caused due to bad weighting (e.g. rectangle area in Fig. 12b). Such problems can be rectified by refining the influence weights of relevant joints (e.g. rectangle area in Fig. 12c), but geometry loss still occurs at the joints (e.g. dents on the top of the shoulder and the concave side of the elbow as shown in the circle area in Fig. 12c). In comparison, the proposed shape-based deformation approach does not suffer from similar problems while being able to effectively reshape and re-pose the mesh (Fig. 12a).

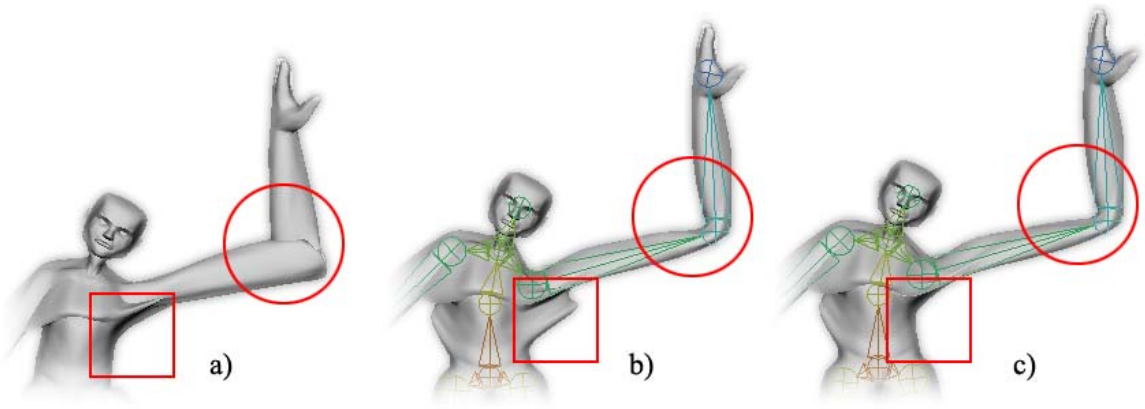

Fig. 12: Comparison with Skeletal Deformation

Here we compare these results with our previous attempt using the lattice-based approach. Fig. 13 shows the deformation results using the same base model and target cage set, where Fig. 13a is the result obtained using the shape-based method (i.e. R_1 in Fig. 11), Fig. 13c is the result from our earlier work (i.e. Fig. 9c-top), and Fig. 13b is the target cage visualization for reference purpose (i.e. Fig. 9b-top V_1 in Fig. 11). According to the illustration, it is obvious that various improvements have been made including better pose handling, more precise deformation, and better control over artifacts.

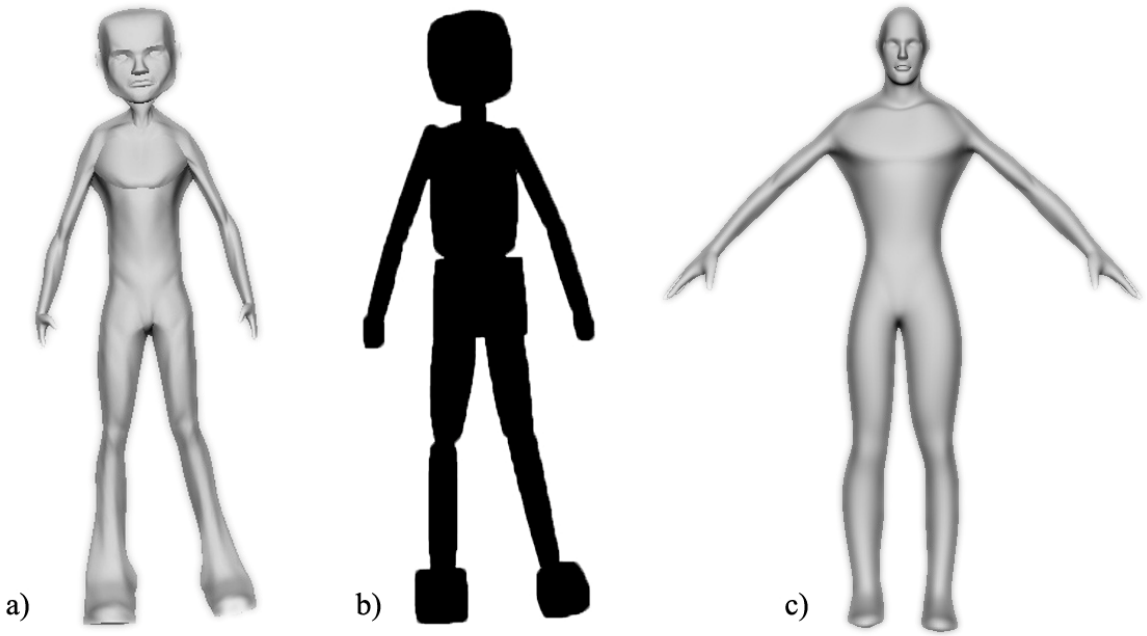

Fig. 13: Comparison with Lattice-based Approach

The proposed shape-based method has some limitations at the moment. For example, some target shapes can be reflected better on the deformed mesh with added geometry resolution. Fig. 14 illustrates such a scenario at the foot-ankle area. It can be noticed that the deformed foot does not possess the exact square-ish look from the target vector due to insufficient local geometry resolution on the original base mesh. The results can be improved by intelligently adding vertices and edges if needed to better retain the target shape feature.

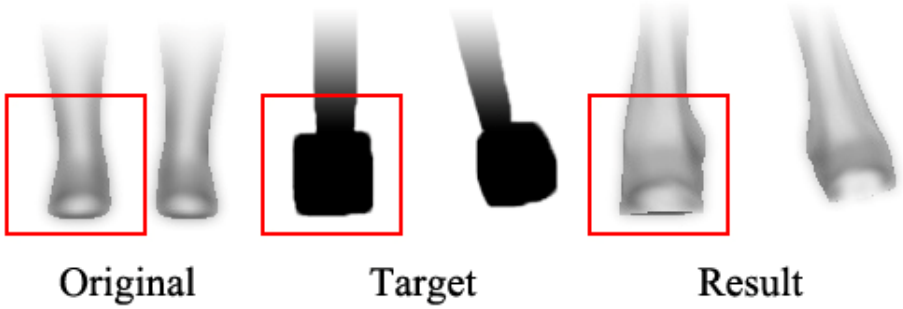

Fig. 14: Lack of Geometry Resolutions

8. Conclusion and Future Work

We have demonstrated a system that allows artists to intuitively reshape an existing detailed 3D character model using 2D character sketch inputs, in just a few seconds. This enables them to quickly visualize characters in the 3D space, without spending much effort in modeling/texturing/deformation/detailing. We believe this can help significantly in the brainstorming of new characters, as well as in the procedural re-purposing of existing 3D meshes. By exploring and comparing lattice-based approach and our novel parametric deformation method [38], we have illustrated decent quality results for deforming two characters models with significantly different builds. Our approach focuses on intuitiveness and automation, which makes it suitable as a quick 3D character visualization tool.

Improvements currently under development for the deformation pipeline include support for multi-stroked silhouette inputs instead of body-part scaffold drawings) to cater to more experienced artists. In addition, geometry transfer can be added to the pipeline to enable more flexible character visualization. Future direction along this direction can be to transfer posture, texture and colour together with the geometric features.

References

1. Gain J. and Bechmann, D. A survey of spatial deformation from a user-centered perspective. ACM Trans Graph, 27, 4, Article 107 (October 2008), 21 pages, 2008.
2. Bechmann, D. Multidimensional free-form deformation tools. In proceedings of EUROGRAPHICS'. State of the Art Reports(STARs), 1998.
3. Sederberg T. W. and Parry S. R. Free-form deformation of solid geometric models. Comput. Graph. 20, 4, 151-160.
4. Griessmair J. and Purgathofer W. Deformation of solids with trivariate B-splines. Eurographics 89, 137-148.
5. Coquillart S. Extended free-form deformation: A sculpturing tool for 3D geometric modeling. Comput. Graph. 24,4,187-196.
6. MacCracken R. and Joy K. Free-form deformations with lattices of arbitrary topology. In: SIGGRAPH 96 Conference Proceedings, pp. 181-188, 1996.
7. Singh K., and Kokkevis E. Skinning Characters using Surface-Oriented Free-Form Deformations. Graphics Interface 2000: 35-42. 2000.
8. Singh K. and Fiume, E. Wires: A Geometric Deformation Technique. Comput. Graph., 405-414, 1998.
9. Kho Y. and Garland M. Sketching Mesh Deformation. ACM Trans Graph, 24,3, 934-934, 2005.
10. Botsch M., Pauly M., Wicke M., and Gross M. H. Adaptive space deformation based on rigid cells. Computer Graphics Forum 26,3,339-37,2007.
11. Igarashi T., Moscovich T., and Hughes J. F. As-rigid-as-possible shape manipulation. ACM Trans. Graphics 24(3), 1134-1141, 2005.
12. Schmidt R., Isenberg T., Jepp P., Singh K., and Wyvill B. Sketching, Scaffolding, and Inking: A Visual History for Interactive. 3D Modeling, 2007.
13. Thorne M., Burke D., and van de Panne M. Motion Doodles: An Interface for Sketching Character. ACM Siggraph, 424-431, 2004.
14. Orzan A., Bousseau A., Winnemöller H., Barla P., Thollot J., and Salesin D. Diffusion Curves: A Vector Representation for Smooth-Shaded Images. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2008), Volume 27, 2008.
15. Schmidt R., Wyvill B., Sousa M.C., and Jorge J.A. ShapeShop: Sketch-Based Solid Modeling with BlobTrees. 2nd Eurographics Workshop on Sketch-Based Interfaces and Modeling, 53-62, 2005.

16. Igarashi, T., Matsuoka, S., and Tanaka, H. Teddy: a sketching interface for 3d freeform design. In Proceedings of the 26th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., 409–-416, 1999.
17. Hua, J., and Qin, H. Free-form deformations via sketching and manipulating scalar fields. In Proceedings of the eighth ACM symposium on Solid modeling and applications, ACM Press, 2003.
18. Kraevoy, V., Sheffer, A., and Panne, M, Modeling from contour drawings. Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling. ACM, 2009, pp. 37-44.
19. Bebis, G., Georgiopoulos, M., and Da Vitoria Lobo, N. Using self-organizing maps to learn geometric hash functions for model-based object recognition. IEEE Transactions on Neural Networ006Bs 9, 3, 560–570, 1998.
20. Ballard, D. H. Generalizing the hough transform to detect arbitary shapes. Pattern Recognition 13, 111–122, 1981.
21. Belongie S., Malik, J. and J., P. Shape matching and object recognition using shape context. IEEE Transactions on Visualization and Computer Graphics 24, 24, 2002.
22. Loncaric S. A survey of shape analysis techniques. Pattern Recognition 31, 983–1001, 1998.
23. Pavlidis T. A review of algorithms for shape analysis. Comput. Graphics Image Process. 7 (2), 243–258, 1978.
24. Fiore, F. D., Reeth, F. V., Patterson, J., and Willis, P. Highly stylised animation. The Visual Computer 24, 2, 105–123, 2008.
25. S´ykora, D., Buri´anek, J., and Z ´ara, J. Sketching cartoons by example. In Proceedings of Eurographics Workshop on Sketch-Based Interfaces and Modeling, 27–34, 2005.
26. S´ykora, D., Dingliana, J., and Collins, S. As rigid-as-possible image registration for hand-drawn cartoon animations. In Proceedings of International Symposium on Nonphotorealistic Animation and Rendering, 25–33, 2009.
27. S´ykora, D., Sedl´a ˇCek, D., Jinchao, S., Dingliana, J., and Collins, S. Adding depth to cartoons using sparse depth (in)equalities. Computer Graphics Forum 29, 2, 2010.
28. Gingold, Y., Igarashi, T., and Zorin, D. Structured annotations for 2D-to-3D modeling. ACM Transactions on Graphics (TOG) 28, 5, 148, 2009.
29. Ashraf G., Why Y. P., and Islam M. T. Mining human shapes perception with role playing games. In 3rd Annual International Conference on Computer Games, Multimedia and Allied Technology, Singapore, pp 58-64, 2010.
30. Islam, M. T., Nahiduzzaman K. M., Why Y. P., and Ashraf G. Learning from humanoid cartoon designs. In 10th Industrial Conference on Data Mining ICDM, 2010.

31. Beiman, N., Prepare to Board! : Creating Story and Characters for Animated feature
32. Camara, Sergi, All about techniques in drawing for animation production, 1st ed.2006, Barron's Education Series, Inc.
33. Hart, C., Cartoon Cool: How to Draw New Retro-Style Characters
34. Pizlo Z. 3D Shape: Its Unique Place in Visual Perception, MIT Press, Cambridge MA, ISBN: 978 0262162517, 2008.
35. Angelidis A., Singh K. Space deformations and their application to shape modeling. ACM SIGGRAPH 2006 Courses, July 30-August 03, 2006, Boston, Massachusetts, 2006.
36. Pizlo Z. 3D Shape: Its Unique Place in Visual Perception, MIT Press, Cambridge MA, ISBN: 978 0262162517, 2008.
37. Islam M. T., Why Y. P. and Ashraf G. Learning Shape-Proportion Relationships from Labeled Humanoid Cartoons. 6th International Conference on Digital Content, Multimedia Technology and its Applications, Seoul, 416-420, 2010.
38. Ashraf G., Nahiduzzaman K. M, Hai L. N. K., and Li M. Drafting 2D Characters with Primitive Scaffolds", Second International Conference on Creative Content Technologies, Computation World, Lisbon, Nov, 2010.
39. Baran I., and Popović J. Automatic rigging and animation of 3D characters. ACM SIGGRAPH, August 05-09, 2007, San Diego, California, 2007.