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 detailed 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). The actual deformation is done using automatically setup Free Form Deformation cages. 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

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. From several shape-proportion guides in art and psychology literature [1, 2, 3], 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. Since shape-vector targets can also be procedurally generated, character-attribute centric deformations can be done for online generation of virtual characters. In this paper, we describe the relevant details that allow shape vector deconstruction of 3D meshes, and shape fitting of input strokes, as well as automatic construction of Free Form Deformation (FFD) lattices to implement the deformation.

Though we use FFD to implement the deformation, we could theoretically use any other method like skeletal or wire de Former. We choose FFD here because it is a model-free space transforming method, as opposed to parametric methods like Linear Blend Skinning. Our system works under no assumption on the geometry topology and 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:

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

b) 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.

c) 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.

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. Lastly, we present results, accompanied with analysis of potentials and limitations.

2 Related Work

Shape Signature: Shape representation is a well-studied field because of its tremendous importance in pattern recognition and computer vision [4,5,6,7,8]. 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 [4]. 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 [9]. 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.

Deformation: 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. In Free Form Deformation methods [12], the displacement of a cage control-point influences the entire space inside the lattice. However, specifying mesh deformations this way is both cumbersome and counterintuitive. Griessmair and Purgathofer [13] extended this technique to employ a trivariate B-spline basis. Though these methods are simple, efficient and popular in use, they 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. EFFD [14] is an improvement as it allows user-specified base-shapes, but manual lattice creation and deformation are still cumbersome [15]. MacCracken and Joy [16] use 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 [17].

Sketching: Schmidt et al [18] 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 [19] proposed the concept of sketching for character animation, but do not include shape modeling. Orzan et al [20] 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 [21] 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.

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 [22]. Integrated bone and cage deformation systems avoid potential artifacts that may arise in case of independent localized cages [19].

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. [23, 24, 25], Gingold et al. [26], and Fiore et al. [27]. 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 [1,2,3] as well as shape perception literature [28]. 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 [9,29,30].

3 Supporting Algorithms

We briefly describe our prior work on shape representation [9] and parameterization [31] 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.

\[
p_j' = \sum_{i=1}^{3} (w_i \cdot p_{i,j}) \quad (1)
\]

\[
m_j' = \sum_{i=1}^{3} (w_i \cdot m_{i,j}) \quad (2)
\]

where \( j \in \{1,2,3,4,5,6,7,8\} \) and \( \sum_{i=1}^{3} w_i = 1 \)
In the above equations, \( p' \) and \( m' \) represent the \( j \)-th blended segment boundary and midpoints respectively, while \( p_{ij} \) and \( m_{ij} \) represent the corresponding control points in the \( i \)-th primitive shape (circle, triangle, square). \( W_i \) is the weight contribution from the \( i \)-th primitive shape. Results of some blend operations are shown in Fig. 1. The cross hair under the shapes indicates the shape weights.

![Fig. 1: Consistent interpolation of circle, triangle, and square [9]](image)

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) [9]](image)

### 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. Each pixel in Cartesian coordinates \( \{x,y\} \) can be easily converted into polar shape coordinates \( \{s,t\} \) and vice versa.

![Fig. 3: Polar Coordinate Parameterization of a Cage [31]](image-url)
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 [32]. 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 (see Sec. 3.2) to generate the corresponding shape vectors. Note that accurate shape fitting of the 3D mesh body parts is only useful for arbitrary topology FFD [14, 16] implementations. For basic parallelepiped FFD implementations [12], doing an Oriented Bounding Box computation of the above projected convex-hull points is sufficient.

5 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 [19]; 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. 4. Using the template of a skinny girl image, a rough sketch of body parts (omitted in Figure 4a 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. 4a), to extract shape vectors for each individual body part (Fig. 4b), using the vector fitting technique in Sec. 3.2. Its corresponding 2D FFD lattice deformer setup (Fig. 4c), serves as the deformation target. Similarly, automatic body part shape analysis is performed on the source character model (Fig. 4d), to produce a set of vectors (Fig. 4e) corresponding to those of the 2D drawings. A set of FFD lattice deformers (Fig. 4f) can then be constructed from these vectors, which completes the process by deforming the source model (from Fig. 4d to Fig. 4g).
Lattice deformation proceeds in a standard manner, as described by Sederberg and Parry [12], once the source and target lattices are set up from the sketch and 3D undeformed model, respectively. In practice, arbitrary topology FFD [14, 16] yields better results than the original parallelepiped deformer base configuration in [12].

The rest of this section explains the automatic full body FFD lattice system construction from step (e) to (f) in Fig. 4. 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 5x5x2 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 5x5x2 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. 5, 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: $P_i = \sum S_i v_s + T_i v_t + C_i v_c$, 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. 5: Computing lattice control points from shape vectors](image)

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. 5.

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 Results and Analysis

Fig. 6 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 [32], takes a few minutes on lightweight meshes (1-100K triangles), so we do this as an offline step.

There are a few limitations to our current work. 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. 7, 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.
7 Conclusion and Future Work

We have successfully 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. We believe this can help significantly in the brainstorming of new characters, as well as in the procedural re-purposing of existing 3D meshes. 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 include: use of our novel parametric deformation [31] to replace FFD deformation, for better volume preservation, multi-view and pose aware adjustments to foreshortened body-parts, and support for multi-stroked silhouette inputs instead of body-part scaffold drawings) to cater to more experienced artists.

Acknowledgement

This research is funded by MDA GAMBIT fund (WBS: R-252-000-357-490), sponsored by Media Development Authority of Singapore.
References

3. Hart C., Cartoon Cool: How to Draw New Retro-Style Characters


