Life-Sketch - A Framework for Sketch-Based Modelling and Animation of 3D Objects

Rong Yang **Burkhard C. Wünsche**

Graphics Group, Department of Computer Science
University of Auckland,
Private Bag 92019, Auckland 1142, New Zealand,
Email: wizard.yang@gmail.com, burkhard@cs.auckland.ac.nz

Abstract

The design and animation of digital 3D models is an essential task for many applications in science, engineering, education, medicine and arts. In many instances only an approximate representation is required and a simple and intuitive modelling and animation process, suitable for untrained users, is more important than realism and extensive features. Sketch-based modelling has been shown to be a suitable interface because the underlying pen-and-paper metaphor is intuitive and effective.

In this paper we present *LifeSketch*, a framework for sketched-based modelling and animation. Three-dimensional models are created with a variation of the popular "Teddy" algorithm. The models are analysed and skeletons with joints are extracted fully automatically. The surface mesh is bound to the curved skeletons using skinning techniques and the resulting model can be animated using skeletal animation methods.

The results of our evaluation and user study suggest that modelling and animation tasks are considerable more efficient than with traditional tools. The learning curve is very flat and a half page document was sufficient to familiarise users with the tools functionality. Users were satisfied with the automatically extracted joints, but some users struggled selecting the appropriate rotation axes and angles for animating the resulting 3D objects. A more intuitive, preferable automatic or sketch-based approach for animations is needed. Overall users were satisfied with the modelling capabilities of the tool, found most of its functionality natural and intuitive, and they enjoyed using it.

Keywords: sketch-based modelling, skeletal animation, skinning, human-computer interfaces

1 Introduction

The design and animation of 3D computer models is essential for many applications in science, engineering, education, medicine and arts. In many cases rough prototypes of a model are sufficient. For example, in the early production stages of professional animations storyboards are used to translate the story into images and organize scenes (Hart 1999). Improvements in computer technology have led to digital storyboards and sketch-based design tools (Landay & Myers 2001) and simple sketch-like interfaces for human character animation (Mao et al. 2006). In medical imaging and scientific computing rough prototypes (frequently termed *default models*) are used for segmentation, feature recognition and object tracking (Cootes et al. 1995, Lepetit & Fua 2005). In such applications the topology of the model is more important than its exact geometry. Approximate models are also useful for demonstrating basic concepts, e.g., in education.

Creating models and animations with sketches is particularly attractive since it encourages creativity (Gross & Do 1996) and enables user to concentrate on the overall problems rather than details (Wong 1992). The past decade has seen a tremendous increase in the design and use of sketch-based interfaces.

In this paper we present LifeSketch, a prototype of a sketch-based modelling and animation system. The modelling process is based on Igarashi et al.'s famous "Teddy" system. The main contribution of our work is an animation system which automatically detects movable parts and enables skeletal animation based on curved bones. A subsequent user study demonstrates that the system is sufficient for creating rough 3D models and the automatic joint detection is intuitive.

Section 2 reviews previous work on sketch-based modelling and animation systems. Section 3 presents the design of our system including the modelling, animation and rendering steps. Section 4 discusses the user study we use to evaluate our system and section 5 summarizes result of the user study and our own evaluation. We conclude the paper in section 6 and discuss important future work.

2 Literature Review

2.1 Sketched-Based Modelling Systems

Our analysis of successful sketch-based modelling systems for 3D objects showed that virtually all applications use the following three steps:

1. Features characterising the 3D object to be modelled are sketched in 2D using as few strokes as possible
2. The sketched 2D strokes are mapped to 3D shapes according to application specific constraints, which reflect assumptions about the shape of the 3D object to be modelled
3. Ambiguities are resolved and more detailed features added by using *modifier strokes*. Frequently these strokes are directly applied to the 3D shape resulting from the previous step

These steps do not include 2D sketch processing and recognition, which is a non-trivial task. Igarashi et al. (1997) evaluate each stroke input for potential

geometric relations such as connections, alignments, horizontal and vertical strokes, and symmetry. Interactive beautification is performed after identifying the most suitable geometry relation. Sezgin et al. (2001) eliminate noise from free-hand drawings by combining average based filtering and scale space filtering. The method uses curvature information and pen speed data in order to differentiate between shape features of a curve and unintended wriggles.

Different approaches have been suggested to classify sketch-based modelling algorithms. Eggli et al. (1997) categorize sketch-based modeling systems by the different types of drawing the user wants to create: technical, symbol and free-form. McCord et al. additionally differentiate free-form shapes into blobby and domain specific shapes (McCord et al. 2008). Olsen et al. (2009) present an excellent survey of sketch-based systems and classify them according to the type of modeling operation considered, i.e., creating full models from sketches, augment existing models with sketches, and deform existing models using sketch input.

We suggest to classify sketch-based modelling techniques according to the sketched primitives used to characterise the desired 3D model: silhouettes, contours, cross-sections and skeletons. In addition 3D objects can be created by deforming more basic shapes using sketch-input.

This classification takes into account that the most natural description of objects depends not on their type (e.g., man-made vs. natural), but on their structure and the required level of abstraction. This in turn can vary for different users and different applications. In many instances several of these concepts must be combined. For example, the stem and branches of a flower can be represented by a skeleton and its leaves by silhouettes and shape modifying strokes (Ijiri et al. 2005). Note that the type of captured primitives together with assumptions made about the shape of the desired 3D object determines the algorithms used to reconstruct the 3D surface from the sketch input.

2.1.1 Silhouette-Based Methods

The arguably most popular class of sketch-based 3D modelling techniques uses sketch input to represent the silhouette (outline) of a 3D object. The outline, usually referred to as contour, is expanded to a 3D object by making the assumption that the object is "blobby", i.e., the cross section of each component of the sketched contour is circular.

The arguably best known system in this class is Igarishi et al.'s Teddy application (Igarashia et al. 1999). A 3D object is computed by sampling the contour (outline), triangulating the sample points, computing a skeleton from the mid-points of all internal edges of the triangles, and then fitting circular cross-sections around the skeleton. Additional functionalities for cutting and combining objects allow the creation of complex, inflated (blobby) shapes. Various modifications have been suggested, e.g., for smoothing the resulting 3D surface (Igarashi & Hughes 2003), smoothing the underlying skeleton (Levet & Granier 2007), or for modifying the shape by sketching contours of local features (Zimmermann et al. 2008).

Karpenko et al. use implicit surfaces to "inflate" contours to 3D bodies. As a result different sketched components can be easily blended together (Karpenko et al. 2002). Similar ideas are employed in ShapeShop (Schmidt et al. 2006) and MIBlob which uses implicit surfaces to inflate contours traced in medical images (de Araújo et al. 2004). Other authors have shown that complex 3D objects can be edited using stylus strokes that retrace an object's silhouette (Cheutet et al. 2005, Hua & Qin 2003). The modification of a models silhouette subsequently rescales it so that it remaps itself to the new silhouette.

Two interesting application of silhouette-based algorithms are garment and tree modelling. For tree modelling the user sketches the outline of the crown of the tree and the algorithm computes a fitting branching structure based on existing templates and a probabilistic distribution (Chen et al. 2008). Garments can be modelled by sketching their outline and the algorithm automatically fits them to the body shape (Turquin et al. 2007).

2.1.2 Contour-Based Methods

Contour-based methods are related to silhouette-based techniques. We use the popular definition that silhouettes represent the outline of shape (e.g., the shape of its shadow), but contours include all visible lines and divisions of a shape, e.g., discontinuities in the surface gradient. Karpenko & Hughes (2006) extend the closed shape outline of "Teddy" with cusps, T-junctions, and overlapping sketch lines in order to represent local details. Nealen et al. (2007) allow the user to draw arbitrary silhouette lines and use free form deformations to adapt the underlying 3D shape.

Gain et al. (2009) enable users to model 3D terrains by drawing the silhouette, spine and bounding curves of both extruding (hills and mountains) and embedding landforms (river courses and canyons).

A popular application of contour-based methods is the sketching of technical drawings and 3D CAD models. Computer designed items are often characterized by a blocky shape, flat or arced surfaces, sharp or evenly rounded edges and corners, many parallel and orthogonal edges and faces, and symmetrical features. These features can be captured using silhouettes which can then be interpret using application specific constraints, e.g., that surfaces of CAD objects frequently form 90 degree angles. Examples include "SKETCH" (Zeleznik et al. 1996), where complex objects are constructed using a combination of sketched 3D primitives. "Quick-sketch" allowed users to sketch 3D solid objects and B-spline surfaces (Eggli et al. 1997). A subsequent system, "3D Sketch", creates an edge graph from a user sketch and matches it with existing topologies. After identifying planar faces and determining a view point the edge graph can be projected into a 3D model (Mitani et al. 2002). Recently active contour models were proposed to allow both curve creation and modification from totally arbitrary view points (Kara & Shimada 2007). The depth coordinates are computed by minimizing the spatial deviation from the original target curve.

2.1.3 Skeleton-Based Methods

The third group of sketch-based modelling techniques are skeleton-based. These methods can be considered the dual to the silhouette-based techniques which frequently use the sketched outline to compute a skeleton (as done in "Teddy").

Skeleton-based techniques are most popular for modelling complex fibrous and branching structures. Ijiri et al. (2005) use sketch input to model the stem and branches of flowers. The 3D shape of a stem is computed by solving a differential equation such that the curvature and appearance of the resulting 3D shape is identical to the 2D sketch. User input can be reduced by combining the technique with L-systems. The L-systems reproduction rules simulate a growth pattern which can result in arbitrary complex shapes and the sketch input controls the overall appearance and the depth of the recursion. The techniques has been used for tree modelling (Ijiri et al.

2006) and modeling Purkinje fibres in the heart (Ijiri et al. 2008). Takayama et al. (2008) present a tool for modelling the myocardial fibre structure of the heart muscle. The user first creates a depth field representing layers of the muscle tissue and then indicates the fibre orientation within each layers using sketches.

A more general system is "Thor" (Arcila et al. 2008). The user draws a skeleton using a serious of sketches. The initial sketch defines the main shape and subsequent sketches modify it. The user can draw a radius for each skeleton segment and a 3D surface is generated by fitting a generalised cylinder to it. Grimm & Hughes (1998) take a similar approach based on implicit surfaces.

2.1.4 Cross Section-Based Methods

A less common approach for creating 3D models is to sketch 2D cross-sections. McCord et al. (2008) model orchids by allowing users to sketch the cross section of the labellum of an orchid. The cross section is then expanded into a 3D surface by fitting ellipsoidal contours around it and modifying the surface with a noise function which depends on the distance to the cross section and the apex of the cross section.

Xie & Wünsche (2010) model 3D terrains by sketching and labelling contour lines which are sampled, triangulated and then interpolated to compute a digital elevation mesh representing the terrain. Cross sections are used in most professional modelling tools, but are usually not sketched but represented by parametric curves. 3D surfaces are obtained by extrusion or by computing the tensor product with a second parametric shape. Google SketchUp employs some of these principles using an interface mixing sketch and CAD elements (Google 2009).

2.1.5 Deformation-Based Methods

The last category of modelling systems uses sketch input to deform (usually sketched) 3D models. In the easiest case surfaces are simply "pulled" to fit a modifier stroke, e.g., when bending sketched leaf shapes (Ijiri et al. 2005). A more complicated approach is to use a free form deformation which either applies directly to the surface of the input object (Nealen et al. 2007) or to the underlying volumetric representation if the shape is implicitly defined (Sugihara et al. 2008)

2.2 Sketched-Based Animation Systems

Relatively few techniques exist for the sketch-based animation of objects. The majority of methods either allow users to define motion paths or to adjust rigid characters to compose different poses.

Steger (2004) presents a visual language for creating simple 2D animations within a sketch-based interface. Motions are represented by motion paths, which are drawn as arrows. Disparate motions are synchronized using *events* which are indicated by time stamps along the motions paths. Motion Doodles (Thorne et al. 2004) allow the user to sketch a motion path for a sketched character which can consist of up to seven components with predefined functionalities (head, body, arms, etc.). The system parses the motion path and maps it to a parameterized set of 18 different output motions. Motion paths have also been used for robot navigation (Sakamoto et al. 2009). Additional control is achieved by stroke gestures and sketching operation areas.

Davis et al. (2003) achieve animations of articulated 3D characters by creating 2D sketches of the character in key frame poses. The authors use a set of constraints reflecting assumptions about the intentions of the user. For example, the system assumes that all objects are rigid. If a bone has different lengths in different key frames then this indicates a rotation. When the bone is longest it is assumed to be parallel to the image plane. Mao et al. (2007) use a similar approach. Users can sketch skeletons and body outlines at key frame poses. The body contour is used to estimate the fat distribution which influences the animation.

There are several papers about animating sketched objects. In all of these applications the animation itself does not use sketch-input. Igarashi et al. (2005*b*) use spatial key framing where key frames are not determined by points in the temporal domain, but by key poses of the 3D object. A different approach is used for the "As-Rigid-As-Possible Shape Manipulation" (Igarashi et al. 2005*a*). The user can animate a shape by selecting arbitrary points within it and moving them. This is achieved by triangulating the 2D shape and computing a configuration containing the moved points (triangles) such that distortions of all other triangles are minimised. The system is very intuitive and easy to use when employing a multi-touch interface. However, it does not extend to 3D shapes and the animations would be difficult to control using sketch-input (mouse, tablet and pen).

3 System Design

We want to have a system which is easy to use and intuitive, but at the same time flexible and extendable so that it can be integrated into more complex modelling environments. Our analysis of the literature shows that the Teddy algorithm is one of the most popular modelling tools, that many improvements exist, and that it has been well tested with even a commercial game based on it. For animation purposes we would like to use skeletal animation since it is widely used, supported by many graphics APIs and graphics engines, and because of the possibility to use existing motion capture and animation data. Baran and Popović demonstrated that it is possible to animate a wide variety of shapes with the same skeleton (Baran & Popović 2007).

Our system can be divided into five modules which are explained subsequently:

1. Model and spine generation
2. Skeleton generation - bones and joints
3. Sketelal animation and skinning setup
4. Rendering
5. Animation

3.1 Model and Spine Generation

The sketch-based model is created using the Teddy algorithm (Igarashia et al. 1999). The user sketches a 2D outline of the shape which is sampled. The sample point are triangulated using a Constrained Delauney Triangulation and classified according to the number of internal edges. Triangles with one internal edge are termed *terminal triangles*, triangles with two internal edges *split triangles*, and triangles with three internal triangles (i.e., no edge on the contour) are called *junction triangles*.

A skeleton, the so-called *chordal axis*, is defined by connecting all the mid points of internal edges and centroids of junction triangles. Small side branches of the skeleton are eliminated by using a pruning operation (Igarashia et al. 1999). Starting from the terminal triangle, triangles are merged until the merged

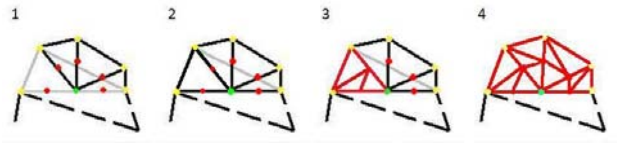

Figure 1: Left image: A sleeve triangle indicated by light grey lines. Second image from the left: The triangle fan after performing a pruning operation. The green dot represents an elevated spine node (ESNs), the yellow dots are vertices on the sketched contour and the red dots are vertices on the quarter circles connecting contour vertices to ESNs. The two images on the right show how these vertices are triangulated by connecting the mid points of the quarter circles and by subdividing the resulting quadrilaterals.

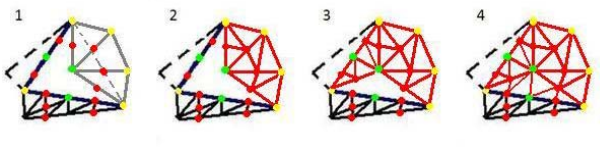

Figure 2: Left image: A junction triangle with a pruned edge resulting in a triangle fan (grey lines). The green dots represent elevated spine nodes (ESNs), the yellow dots are vertices on the sketched contour and the red dots are vertices on the quarter ovals connecting contour vertices to ESNs. Second image from the left: After elevating the spine nodes the 3D vertices corresponding to the triangle fan are triangulated using the same algorithm as for a pruned sleeve triangle. Two images on the right: The remaining part of the junction triangle can be divided into four spherical triangles by connecting the ESNs of the pruned edge to the other ESNs and the contour point opposite to it. The four spherical triangles can be triangulated by connecting the vertices along the quarter circle in a zig-zag pattern similar as for a normal junction triangle.

triangle is larger than the half circle around its internal edge. The merged triangle is then triangulated again using a triangle fan originating at the mid-point of its internal edge.

A 3D shape is formed by elevating all nodes of the chordal axis orthogonally to the sketch plane. The height is equal to the distance of a node to the next sample point on the sketched contour. Note that the nodes are either the mid-points of internal edges or the centroids of junction triangles, i.e., the distance is easily computed. The sample points on the contour and the elevated nodes are then connected by quarter circles which are sampled and triangulated to create a surface mesh.

Surface construction for pruned sections of the chordal axis requires special considerations which are not explained in the Teddy paper and subsequent papers. Two examples are indicated in figure 1 and 2. The complete details are given in (Yang 2009).

3.2 Skeleton Generation - Bones and Joints

In order to animate the object we have to define a skeleton which is a hierarchical structure consisting of bones and joints. The chordal axis is an ideal candidate for this since it lies approximately in the centre of the sketched contour and it has a branched structure. We define a *spine tree* by first finding the largest junction triangle. The largest junction triangle usually represents the widest component of the final shape. For example, for a human shape that would be the body. We therefore make the centre of this triangle the root of the skeleton. We then traverse the chordal axis graph in pre-order starting with this node. The resulting spine tree has the following properties:

- All centres of junction triangles are called branch nodes and are internal nodes of the spine tree. The root of the tree is the centre of the largest junction triangle. If no branch node (i.e., no junction triangle) exist then the resulting object is rigid.
- All branch nodes, which are directly connected to a branch node detected earlier in the pre-order traversal of the tree, are children of that node. The branch nodes are connected by sections of the chordal axis.
- The leaves of the tree are spine nodes belonging to terminal triangles. The leaves are the children of the branch node to which they are directly connected by sections of the chordal axis.

The tree represents a hierarchical skeleton, but the branch nodes (centres of the junction triangles) do not represent suitable joints of the skeleton as illustrated in the images on the left of figure 3 and 4.

Marr & Nishihara (1978) noted that the concave parts of a silhouette define the subparts of an object. We observed that the edges of junction triangles isolate the subparts of the sketched shape and hence define candidate joints. In a previous paper we presented an algorithm for selecting folding axes of a 2D sketched contour by merging edges of junction triangles and determining "bendable" sections (McCord et al. 2008). The results approximate how a piece of paper of the sketched shape can be bend. Applying this algorithm to 3D shapes obtained by the Teddy algorithm does not lead to satisfactory results as demonstrated in the images on the right of figure 3 and 4. Whereas the folding axis in the middle of the torso of figure 3 is still acceptable, the one separating the right shoulder from body is unintuitive. The same problem occurs for the folding axis separating groups of two and three fingers in figure 4.

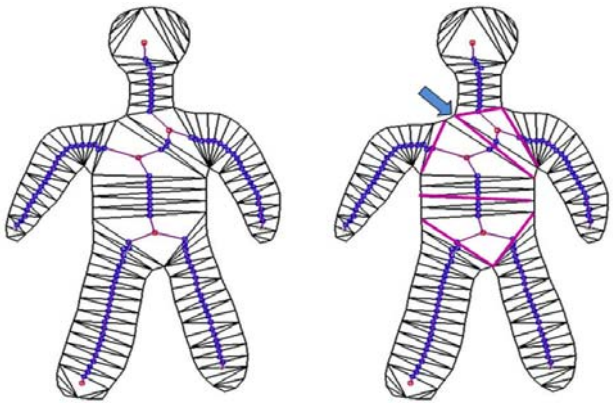

Figure 3: The chordal axis of a sketched contour of a doll (left) and folding axes constructed from it (bold lines) using a paper metaphor (right).

The results indicate that in order to make the tool intuitive we must find a clearly separated subset of these folding axes. We achieve this by clustering branch nodes. All spine nodes lying on edges of junction triangles are potential joints. For each branch node we compute the circumcircle of the corresponding junction triangle. If the circumcircles of two adjacent branch nodes intersect they belong to the same component. If they do not intersect the spine node on

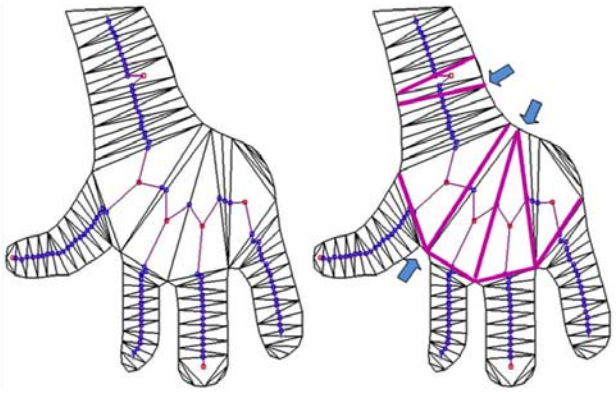

Figure 4: The chordal axis of a sketched contour of a doll (left) and the folding axes constructed from it (bold lines) using a paper metaphor (right).

the edge nearest to the parent spine node is a joint. The algorithm is illustrated in figure 5. The five red dots in the image on the bottom right are the final joints of the skeleton.

3.3 Sketelal Animation and Skinning Setup

An object is animated by moving its bones around joints. In order to get a smooth deformation of the surface mesh the mesh vertices must be associated with bone movements. We use the popular Linear Blend Skinning algorithm which is also frequently called Skeleton Subspace Deformation (SSD). The algorithm is unpublished in the literature but an excellent description is found in (Lewis et al. 2000). A linear blend skin is created by beginning with a static model of the character. We use the 3D model and hierarchical skeleton (spine tree) explained in the previous subsections. Note that the bones of the skeleton are the sections of the spine tree, which connect two joints or a joint and a leaf. Hence the bones are usually curved. We now define for each bone a curvilinear coordinate system by parameterising the corresponding curved section of the spine tree. At each point of the bone two orthogonal vectors are created by the normal vector of the sketch plane and cross product of the normal and tangent of the bone at that point. Using this coordinate system we can now compute the coordinates of a mesh vertex with respect to the bone. This is done by first projecting it onto the sketch plane and then finding the closest point to it on the curved bone.

If vertices are bound to only one bone then linear blend skinning results in gaps or overlaps in the surface. This is avoided by binding a vertex to several bones. We do this by determining for each vertex its distance to a joint and computing appropriate vertex weights. If a vertex has an equal distance to two bones then its weights are 0.5 for each bone. In our case distance is defined by the distance of a sample point to a junction triangle in the original 2D sketch. Since all 3D vertices and the skeleton result from this triangulation it is an appropriate and easy method to determine vertex weights.

In order to animate the mesh vertices we need to rotate bones around their parent joints (i.e., the joint connecting the bone to its parent in the hierarchical skeleton). Let $\mathbf{v}_d^k$ be the coordinates of a vertex in a dress pose with respect to bone k. The position v of the deformed vertex is computed by:

$$\bar{\mathbf{v}} = \sum_{i=0}^{n} w_i \mathbf{M}_i \mathbf{L}_i^{-1} \mathbf{L}_\mathbf{v} \mathbf{v}_d^i \qquad (1)$$

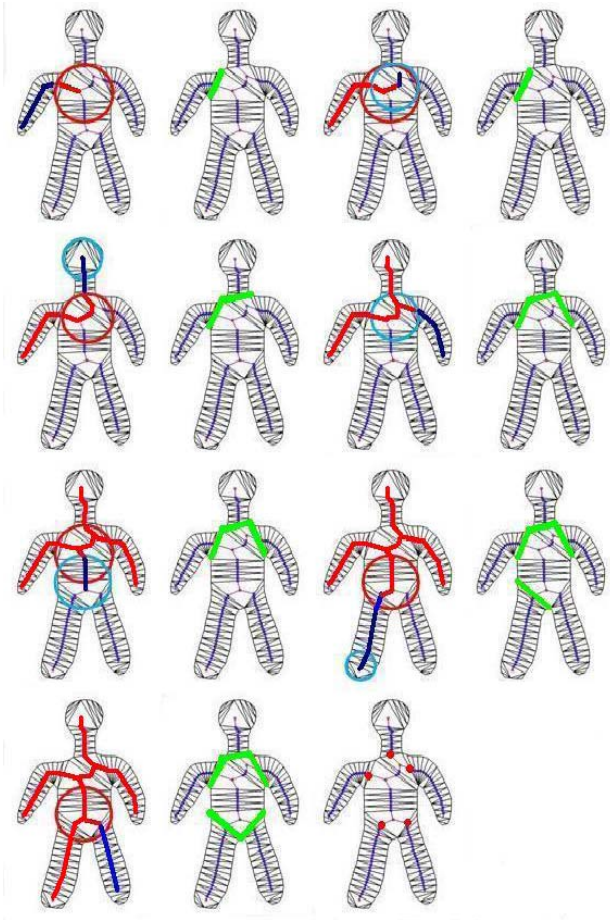

Figure 5: The joint detection algorithm for finding spine nodes which divide the sketched contour into separate parts for animation. The algorithm traverses branches (red and blue lines) in the spine tree. Each branch connects two branch nodes or one branch node to a leaf. Each step tests the intersection (red and light blue cycle) between two adjacent branch nodes' boundary circles. If there is no intersection between two boundary circles or the branch connects a branch node to a leaf, a boundary edge is found (green lines). The spline nodes on these boundary edges are the joints of our skeleton.

where w_i is the weight of the vertex with respect to bone i, $\mathbf{L}_\mathbf{v}$ is the matrix transforming the vertex $\mathbf{v}$ from its surface representation to the world coordinate system, $\mathbf{L}_i^{-1}$ transforms the vertex from the world coordinate system into the static i-th coordinate frame (dress pose), and $\mathbf{M}_i$ expresses the motion of the i-th coordinate frame. The effect of this transformation is that we can express a rotation around an axis of the coordinate frame of the parent joint as rotation around the x-, y- or z-axes. Note that since we have a hierarchical skeleton the matrix $\mathbf{L}_i$ is in fact the product of the matrices representing each bone in the coordinate system of its parent bone.

3.4 Rendering

Two rendering algorithms were implemented: Gouraud shading is the most popular algorithm for polygon rendering and provides a smooth shaded representation of the surface. However, the surfaces produced by the Teddy algorithm frequently suffers from shading artifacts caused by irregularities in the triangulation. This can be remedied using a mesh smoothing algorithm (Igarashi & Hughes 2003), which we plan to implement in future. For the

current model we found that using a toon shader gives visually acceptable results. We implemented the approach presented in (Villar 2007–2009). The cartoon-like appearance corresponds well with the intended usage, i.e., creating rough prototypes of 3D objects and scenes. Figure 6 illustrates the different effects achieved with these rendering algorithms.

Figure 6: Two sketch-based models rendered using Gouraud shading (left) and toon shading (right).

3.5 Animation

An efficient animation of sketched objects can be achieved in three ways: The first possibility is an automatic animation, i.e., without requiring any user inputs. This can be achieved using physically-based modelling, but usually requires some domain knowledge. For example, when we detect that an object does not have joints it can be considered rigid and animated using a physics library such as ODE (Smith 2007). This way the object could collide with other rigid objects or tumble down a (sketched) slope.

If the object does have joints it can still be animated automatically, e.g., using an evolved locomotion controller (Sims 1994, Sanders et al. 2003). In this case naturally placed joints are essential and appropriate constraints (hinge joint, ball joint, etc.) must be known. A sketched object could also be animated by rigging it with an existing animated skeleton using the algorithm presented in (Baran & Popović 2007)

Finally a sketched object can be animated efficiently using sketch input as explained in subsection 2.2. In this case the joints must be intuitively placed and the user must be able to understand how a desired shape can be achieved by rotating a section of the shape around a joint.

In order to test the user's understanding of joint locations and rotations we implemented a very simple key-based interactive animation tool. In this instance keyboard input is preferable over sketch input since it avoids ambiguous responses due to confusion about how to select a joint or about how sketch a desired behaviour.

After the user has drawn a closed contour our system immediately creates the corresponding 3D model and indicates movable parts by thick dotted blue lines as illustrated in figure 7. The user can rotate movable parts using four keys: The first key provides a toggling through all possible rotation axes. The currently active joint is indicated by a yellow dotted line. A second key is used for toggling through the three possible rotation axes represented by the coordinate system of the parent bone at the joint. The last two keys are used for rotating the selected component forwards and backwards around the selected rotation axis.

4 User Study

We performed a user study in order to evaluate the effectiveness of our "LifeSketch" framework. The participants had to do five tasks of various difficulty. The tasks required increasingly complex interactions and had an increasing demand on 3D perception and mental modeling. Our main objectives were to determine whether users are able to model simple shapes, under which circumstances the modelling is most successful, whether the automatically detected joints are natural and intuitive, and whether users can use the joints correctly in order to deform a simple shape.

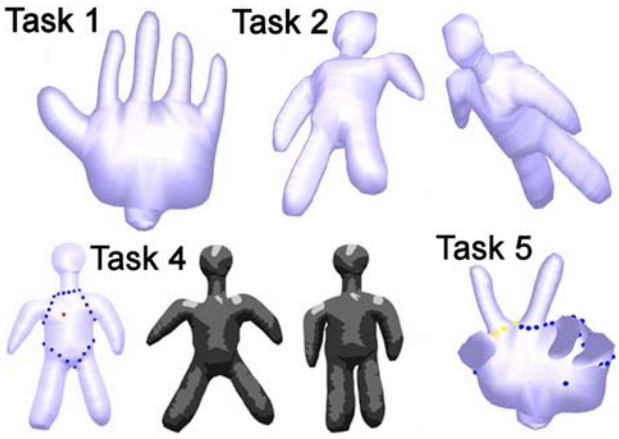

Figure 8: The four pictures shown to the users for the tasks 1,2,4 and 5.

4.1 Hypotheses

Based on our experience with teaching students and based on results from cognitive science we formulated the following hypotheses which formed the bases for our experiments:

- It is easy to create a 3D model from an existing model if the view plane is the sketch plane. In this case the 3D model's contour is identical to the sketch contour and the user only has to copy the contour.
- It is harder to create an appropriate contour sketch for a 3D model shown from a different perspective, since in this case the user has to create a mental model of the object and rotate it to find the correct sketch contour.
- It is hardest to create a sketch for a shape which is only described in words (i.e., without visual template).
- Animating a 3D model is relatively easy if the shape's components move inside the image plane, i.e., the rotation axis is orthogonal to the image plane.
- It is more difficult to animate an object if rotations around axes in all three dimensions are required.

4.2 Tasks

In order to test the hypotheses above, we designed the following five experiments:

1. The user is shown the 3D model of a hand in figure 8 "Task 1" and is asked to copy it. Note that the view direction is orthogonal to the sketch plane, i.e., the user only needs to draw the contour of the 3D model.
2. The user is shown the 3D model of a doll in figure 8 "Task 2" and is asked to copy it. Note that the two views do not contain the contour, i.e., the user must reconstruct it mentally.

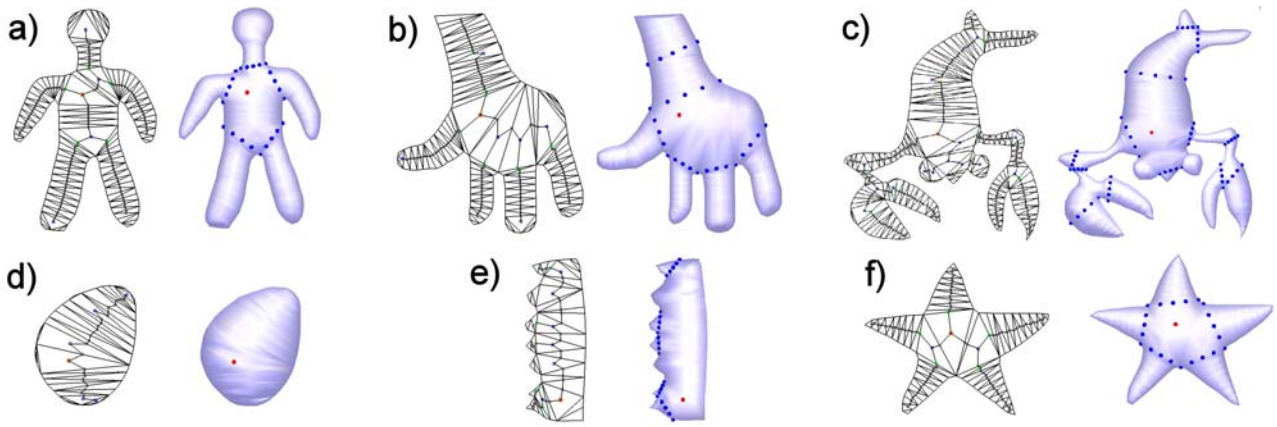

Figure 7: Examples of sketched-based models: (a) doll, (b) hand, (c) lobster, (d) egg, (e) blobby saw, (f) sea star (star fish). Each image shows the sketched contour and resulting skeleton (left), and the Gouraud shaded 3D model (right). The joints separating movable components are indicated by thick blue dotted lines. The root of the hierarchical skeleton for skeletal animation is indicated by a red dot.

3. The user is asked to create a 3D model of a duck. No images are shown and no instructions are given.
4. The user is shown the images of the doll model in figure 8 "Task 4" and is asked to copy the poses by moving the model's legs and arms. Note that the rotations are within the sketch (view) plane.
5. The user is shown the images of the hand model in figure 8 "Task 5" and is asked to copy the pose by moving the hand's fingers. Note that this requires rotations around several axes (for the thumb) and that the fingers move out of the image plane.

5 Results

5.1 Efficiency

We first evaluated the efficiency and correctness of the algorithm. More than a dozen simple models were constructed and a selection of them is shown in figure 7. The most complex model, the lobster, took about 20-30 seconds to sketch. The contour has 243 sample points. The resulting 3D model has 2087 vertices, 4170 triangles, and took 0.493 seconds to compute on a machine with E7200 dual 2.53GHz CPUs with 4GB RAM and NVIDIA GeForce 9600GT graphics card with 512MB memory. The subsequent animations were all performed in real-time with no noticeable delays - the exact frame rate was not measured.

5.2 Correctness

All models we created were plausible with no major artifacts such as holes or non-manifold surfaces. Figure 7 demonstrates that most detected joints are meaningful. In particular note that the egg (stone) has no joints even though its triangulation contains junction triangles. In none of our examples were "natural" joints missing. In Figure 7 (a) it could be argued that the doll figure should also have an elbow joint, but when interviewing the participants in the subsequent user study, none of them commented on this. The algorithm produces some unexpected extra joints, e.g., the mouth and the left elbow of the lobster, the teeth of the saw, and a joint below the wrist of the hand model. The extra joint of the hand demonstrates a problem with Teddy's pruning algorithm which we plan to fix together with the problem of having to many extra joints. The extra joints could be avoided by enforcing a stricter size criteria for branches of the spine tree. However, it is not clear what size is most appropriate. We are planning to conduct an analysis of natural and man-made "blobby" objects in order to optimize the automatic joint selection. Note that some users might expect the eyes of the lobster to be movable.

5.3 User Study Results

We conducted a user study with 11 participants (8 male, 3 female). Eight of them were Computer Science students and six of them had previous experience with modelling tools. The users were asked to perform the five tasks described in subsection 4.2. A time limit of 5 minutes/task was set. The users were observed and unusual actions were recorded. Users were provided with a half page long summary of the functionality of the tool, e.g., the keys for performing rotations as explained in subsection 3.5. However, users were not told what functionality to use for a modeling task. The time for each task was measured. The user experience was evaluated by assessing the modeling results and by asking the user's level of agreement with the following statements:

1. The resulting 3D shape looks like the shape I wanted to model.
2. The resulting 3D shape looks like what I expected after sketching the 2D contour.
3. The rotation axes (joints) were at the positions where I wanted them (This question was asked only for task 4 and 5).
4. The tool is easy to use.
5. The tool is fun to use.

The response were recorded using a seven-level Likert scale ("strongly disagree" (-3) to "strongly agree" (3)) and are shown in table 1.

5.3.1 Task 1

Five users (two non-computer science students) felt that it is difficult to create a model by drawing a contour using a single stroke. Two of them suggested that the tool should have a function which can automatically connect multiple strokes. Two participants suggested that using a pen tablet as input device would be better. We agree that when drawing long curves it is quite cumbersome to keep the mouse

		1 Shape wanted	2 Shape expected	3 Rotation wanted	4 Easy	5 Fun
Task 1	X	1.55	1.73	n/a	1.73	2.18
($\bar{T} = 29s$)	σ	1.04	1.01	n/a	1.52	0.91
Task 2	X	2.00	2.09	n/a	1.90	2.18
($\bar{T} = 36s$)	σ	0.63	0.54	n/a	0.70	0.60
Task 3	X	0.63	1.45	n/a	1.73	2.09
($\bar{T} = 63s$)	σ	1.80	1.37	n/a	1.42	1.14
Task 4	X	1.91	2.00	2.27	1.82	2.55
($\bar{T} = 94s$)	σ	0.70	0.45	0.65	0.87	0.63
Task 5	X	1.18	1.45	0.91	1.18	2.00
($\bar{T} = 165s$)	σ	1.54	1.44	1.64	1.83	1.34

Table 1: Users' level of agreement (from -3 to 3) with the five statements about the five tasks in subsection 4.2. $\bar{X}$ indicates the average response and σ the standard deviation. Below each task the average time for completion is given.

button pressed and keep it moving without creating self intersections. One user in particular had problems with this and with the automatic closing of the contour and had to try four times to create a valid model. In future work we would like to perform the same experiments using a tablet PC with pen-based input.

The high level of agreement with statement 2 is surprising since the Teddy algorithm generates a hand model with a "blobby" palm and not a flat palm as one might expect. One reason for the lack of criticism is that most users didn't try to rotate the hand and hence didn't even notice that the hand's 3D shape was not realistic.

5.3.2 Task 2

Feedback for the second task was in general even better than for the first task. We suppose the reason is that users are more familiar with the tool after the first task. Note that the standard derivation is reduced and we noticed that especially the non-Computer Science students gave better scores. This indicates that the tool is very intuitive and easy to learn. We could not find any support for our second hypothesis that it is harder to draw a shape based on rotated views of it.

5.3.3 Task 3

For task three users had to form a mental image of a duck and then sketch an appropriate contour. Some users forgot the commands to control the tool (e.g., restart), apparently because they were preoccupied creating a mental model. We also found large differences in the level of complexity of the sketched models. We expected users to create a model resembling a rubber duck as shown in figure 9 on the left. However, almost half of the users tried to create a model with wings and/or legs. This resulted in frustration and a low level of satisfaction with the tool. Non-computer science students were overrepresented in that group.

We believe that a more interactive approach, e.g., a user can draw initial shapes and then refine them, would help the user to formulate a mental model and represent it by a sketch.

5.3.4 Task 4

Task 4 was easily accomplished by both computer science students and other users. We observed that users had no problems to rotate 3D shapes when the rotation axis is perpendicular to the image plane. Users

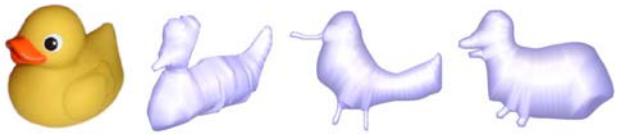

Figure 9: The mental model we expected users to employ when told to model a duck (left) and the actual results of the modelling process (right).

found it easy to select appropriate joints and they had no problem recognizing rotation axes and the associated behavior. Several users required assistance finding the correct key combinations for cycling through joints and performing the rotation - they did not want to read the half page long documentation. Note that this task resulted in the highest level of enjoyment.

5.3.5 Task 5

For the fifth task users had to draw a hand model and rotate its fingers in order to create the shape in figure 8 "Task 5". Table 1 shows that this task had the lowest level of satisfaction, especially among non-computer science students, and the largest variation in user responses. The subsequent interviews showed that joint positions were perceived as intuitive but six participants thought that the rotation of the mesh around joints is not intuitive. Five people felt frustrated when they were rotating the fingers and one participant could not finish the animation within the time limit of 5 minutes.

The main problem was that users had difficulty finding and recognizing the correct rotation axis. This was partially due to their representation as unshaded lines, but also due to that fact that none of the three coordinate axis of the thumb joint corresponded to the direction in which the thumb had to be rotated. Users had to rotate around two different rotation axes which seemed to create considerable confusion, especially for inexperienced users. Another surprising observation was that many users did not rotate the model with the trackball, which would have helped with the perception of the three different rotations possible (the users were made aware of this functionality in the beginning).

The results suggest that any implementation of sketch-based animations should not use rotation axes or at the very least allow arbitrary rotations. In the subsequent interviews users suggested that a more natural way to define rotations would be to indicate a motion path and goal configuration with a curved arrow. In this case the system must use the current configuration and the context to map the arrow into a feasible 3D motion (i.e., use inverse kinematics). More user testing is necessary to verify this hypothesis.

6 Conclusion and Future Work

Sketch-based modelling and animation is an exciting technology with a wide range of applications. We have reviewed the current state of the literature and suggested a novel classification of sketch-based modelling systems which we believe is useful for developing more general modelling frameworks.

We have presented a prototype of our own framework, called LifeSketch. The system is based on the "Teddy" modelling system, but uses novel algorithms to automatically extract a skeleton for skeletal animation. This makes it possible to integrate physical-based animation systems, map motion templates or develop evolutionary algorithms in order to achieve

an automatic animation of sketched objects. We have also presented some previously unpublished details for generating a surface mesh for various configurations resulting from pruning the skeleton.

Our user study demonstrated that the resulting models and joint positions are perceived as natural and intuitive. In general users enjoyed using the tool and were able to complete simple modelling and animation tasks in a short time. The main disadvantage is that the current user interface for specifying joint rotations is clumsy and unintuitive.

We are currently working on a sketch-based interface for animating sketched objects. Arrows drawn by the user are associated with bones and the arrow shape together with the orientation and position of the corresponding bone and joint are used to estimate a rotation axis and angle. We would also like to implement some of the various extensions of the Teddy algorithm in order to increase modelling power and visual attractiveness of the resulting shapes. Finally we are interested in the automatic animation of sketched objects. So far we have a prototype of a physical animation system based on ODE (Smith 2007).

References

Arcila, R., Levet, F. & Schlick, C. (2008), Thor: Sketch-based 3d modeling by skeletons, *in* 'Smart Graphics', pp. 232–238.

Baran, I. & Popović, J. (2007), 'Automatic rigging and animation of 3d characters', *ACM Trans. Graph.* **26**(3), 72. `http://www.mit.edu/~ibaran/autorig/`.

Chen, X., Neubert, B., Xu, Y.-Q., Deussen, O. & Kang, S. B. (2008), Sketch-based tree modeling using markov random field, *in* 'ACM SIGGRAPH Asia 2008 papers', ACM, pp. 1–9.

Cheutet, V., Catalano, C. E., Pernot, J.-P., Falcidieno, B., Giannini, F. & Léon, J.-C. (2005), '3d sketching for aesthetic design using fully free-form deformation features', *Computers & Graphics* **29**(6), 916–930.

Cootes, T. F., Taylor, C. J., Cooper, D. H. & Graham, J. (1995), 'Active shape models—their training and application', *Computer Vision and Image Understanding* **61**(1), 38–59. `http://www.isbe.man.ac.uk/~bim/Papers/cviu95.pdf`.

Davis, J., Agrawala, M., Chuang, E., Popović, Z. & Salesin, D. (2003), A sketching interface for articulated figure animation, *in* 'SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation', Eurographics Association, pp. 320–328.

de Araújo, B., Jorge, J., Sousa, M. C., Samavati, F. & Wyvill, B. (2004), MIBlob: a tool for medical visualization and modelling using sketches, *in* 'SIGGRAPH '04: Posters', ACM Press, p. 107.

Eggli, L., Ching-Yao, H., Bruderlin, B. D. & Elber, G. (1997), 'Inferring 3d models from freehand sketches and constraints', *Computer-Aided Design* **29**(2), 101–112.

Gain, J., Marais, P. & Strasser, W. (2009), Terrain sketching, *in* 'I3D '09: Proceedings of the 2009 symposium on Interactive 3D graphics and games', ACM, pp. 31–38. `http://people.cs.uct.ac.za/~jgain/publications/terrsketch.pdf`.

Google (2009), 'Google sketchup homepage'. `http://sketchup.google.com/`.

Grimm, C. & Hughes, J. (1998), Implicit generalized cylinders using profile curves, *in* 'Implicit Surfaces', pp. 33–41. Creating sweep surfaces using sketching.

Gross, M. D. & Do, E. Y.-L. (1996), Ambiguous intentions: a paper-like interface for creative design, *in* 'UIST '96: Proceedings of the 9th annual ACM symposium on User interface software and technology', ACM, pp. 183–192.

Hart, J. (1999), *The Art of Storyboard: Storyboarding for Film, TV, and Animation*, Focal Press.

Hua, J. & Qin, H. (2003), Free-form deformations via sketching and manipulating scalar fields, *in* 'Proceedings of the 8th Symposium on Solid modeling and Applications (SM 03)', ACM Press, pp. 328–333.

Igarashi, T. & Hughes, J. F. (2003), Smooth meshes for sketch-based freeform modeling, *in* 'I3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics', ACM, pp. 139–142. `http://www-ui.is.s.u-tokyo.ac.jp/~takeo/java/smoothteddy/index.html`.

Igarashi, T., Matsuoka, S., Kawachiya, S. & Tanaka, H. (1997), Interactive beautification: a technique for rapid geometric design, *in* 'Proceedings of the Symposium on User Interface Software and Technology', pp. 105–114.

Igarashi, T., Moscovich, T. & Hughes, J. F. (2005*a*), 'As-rigid-as-possible shape manipulation', *ACM Trans. Graph.* **24**(3), 1134–1141.

Igarashi, T., Moscovich, T. & Hughes, J. F. (2005*b*), Spatial keyframing for performance-driven animation, *in* 'SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation', ACM, pp. 107–115. `http://www-ui.is.s.u-tokyo.ac.jp/~takeo/research/squirrel/index.html`.

Igarashia, T., Matsuoka, S. & Tanaka, H. (1999), Teddy: a sketching interface for 3d freeform design, *in* 'Proceedings of SIGGRAPH '99', ACM Press, pp. 409–416.

Ijiri, T., Ashihara, T., Yamaguchi, T., Takayama, K., Igarashi, T., Shimada, T., Namba, T., Haraguchi, R. & Nakazawa, K. (2008), 'A procedural method for modeling the purkinje fibers of the heart', *The Journal of Physiological Science* **58**(7), 481–486.

Ijiri, T., Owada, S. & Igarashi, T. (2006), The sketch l-system: Global control of tree modeling using free-form strokes, *in* 'Proceedings of Smart Graphics 2006', pp. 138–146. `http://www-ui.is.s.u-tokyo.ac.jp/~ijiri/SketchLSystem/index.html`.

Ijiri, T., Owada, S., Okabe, M. & Igarashi, T. (2005), 'Floral diagrams and inflorescences: interactive flower modeling using botanical structural constraints', *ACM Trans. Graph.* **24**(3), 720–726. `http://www-ui.is.s.u-tokyo.ac.jp/~ijiri/FdAndIfs/index.html`.

Kara, L. B. & Shimada, K. (2007), 'Sketch-based 3d-shape creation for industrial styling design', *IEEE Computer Graphics and Applications* **27**(1), 60–71.

Karpenko, O. A. & Hughes, J. F. (2006), 'Smoothsketch: 3d free-form shapes from complex sketches', *ACM Transactions on Graphics* **25**(3), 589–598.

Karpenko, O., Hughes, J. & Raskar, R. (2002), 'Free-form sketching with variational implicit surfaces', *Computer Graphics Forum* **21**(3), 585–594. http://www.merl.com/papers/docs/TR2002-27.pdf.

Landay, J. A. & Myers, B. A. (2001), 'Sketching interfaces: Toward more human interface design', *Computer* **34**(3), 56–64. http://www.eecs.berkeley.edu/~landay/research/publications/silk-ieee-published.pdf.

Lepetit, V. & Fua, P. (2005), 'Monocular model-based 3d tracking of rigid objects: A survey', *Foundations and Trends in Computer Graphics and Vision* **1**(1), 1–89.

Levet, F. & Granier, X. (2007), Improved skeleton extraction and surface generation for sketch-based modeling, *in* 'GI '07: Proceedings of Graphics Interface 2007', ACM, pp. 27–33.

Lewis, J. P., Cordner, M. & Fong, N. (2000), Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation, *in* 'SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques', ACM Press/Addison-Wesley Publishing Co., pp. 165–172.

Mao, C., Qin, S. F. & Wright, D. (2007), Sketch-based virtual human modelling and animation, *in* 'SG '07: Proceedings of the 8th international symposium on Smart Graphics', Springer-Verlag, pp. 220–223.

Mao, C., Qin, S. F. & Wright, D. K. (2006), Sketching-out virtual humans: from 2d storyboarding to immediate 3d character animation, *in* 'ACE '06: Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology', ACM, p. 61.

Marr, D. & Nishihara, H. K. (1978), Representation and recognition of the spatial organization of three-dimensional images, *in* 'Proceedings of the Royal Society of London, Series B', Vol. 200, pp. 269–294.

McCord, G., Wünsche, B. C., Plimmer, B., Gilbert, G. & Hirsch, C. (2008), A pen and paper metaphor for orchid modeling, *in* 'Proceedings of the 3rd International Conference on Computer Graphics Theory and Applications (GRAPP 2008)', pp. 119–124.

Mitani, J., Suzuki, H. & Kimura, F. (2002), '3d sketch: sketch-based model reconstruction and rendering', pp. 85–98.

Nealen, A., Igarashi, T., Sorkine, O. & Alexa, M. (2007), Fibermesh: designing freeform surfaces with 3d curves, *in* 'SIGGRAPH '07: ACM SIGGRAPH 2007 papers', ACM, p. 41.

Olsen, L., Samavati, F. F., Sousa, M. C. & Jorge, J. A. (2009), 'Technical section: Sketch-based modeling: A survey', *Computers & Graphics* **33**(1), 85–103. http://pages.cpsc.ucalgary.ca/~olsenl/wiki/uploads/Papers/CnGsurvey.pdf.

Sakamoto, D., Honda, K., Inami, M. & Igarashi, T. (2009), Sketch and run: a stroke-based interface for home robots, *in* 'CHI '09: Proceedings of the 27th international conference on Human factors in computing systems', ACM, pp. 197–200.

Sanders, M., Lobb, R. & Riddle, P. (2003), Evolving controllers for virtual creature locomotion, *in* 'GRAPHITE '03: Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia', ACM, pp. 255–256.

Schmidt, R., Wyvill, B., Sousa, M. C. & Jorge, J. A. (2006), Shapeshop: sketch-based solid modeling with blobtrees, *in* 'SIGGRAPH '06: ACM SIGGRAPH 2006 Courses', ACM, p. 14.

Sezgin, T. M., Stahovich, T. & Davis, R. (2001), Sketch based interfaces: Early processing for sketch understanding, *in* 'Proceedings of the Workshop for Perceptive User Interfaces (PUI 01)', ACM Press, pp. 1–8.

Sims, K. (1994), Evolving virtual creatures, *in* 'SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques', ACM, pp. 15–22.

Smith, R. (2007), 'Open Dynamics Engine home page'. http://www.ode.org.

Steger, E. (2004), Sketch-based animation language, Technical report, Department of Computer Science, University of Toronto. URL: http://www.cs.toronto.edu/~esteger/sketchlang/index.html.

Sugihara, M., Groot, E., Wyvill, B. & Schmidt, R. (2008), A sketch-based method to control deformation in a skeletal implicit surface modeler, *in* 'Proceedings of the 5th Eurographics Workshop on Sketch-Based Interfaces and Modeling', pp. 65–72.

Takayama, K., Ashihara, T., Ijiri, T., Igarashi, T., Haraguchi, R. & Nakazawa, K. (2008), 'A sketch-based interface for modeling myocardial fiber orientation that considers the layered structure of the ventricles', *The Journal of Physiological Sciences* **58**(7), 487–492.

Thorne, M., Burke, D. & van de Panne, M. (2004), 'Motion doodles: an interface for sketching character motion', *ACM Trans. Graph.* **23**(3), 424–431. http://people.cs.ubc.ca/~van/papers/doodle.html.

Turquin, E., Wither, J., Boissieux, L., Cani, M.-P. & Hughes, J. F. (2007), 'A sketch-based interface for clothing virtual characters', *IEEE Comput. Graph. Appl.* **27**(1), 72–81.

Villar, J. R. (2007–2009), 'Typhoonlabs' opengl shading language tutorials'. Typhoonlabs Real Time Technologies [Online] www.opengl.org/sdk/docs/tutorials/TyphoonLabs/.

Wong, Y. Y. (1992), Rough and ready prototypes: lessons from graphic design, *in* 'CHI '92: Posters and short talks of the 1992 SIGCHI conference on Human factors in computing systems', ACM, pp. 83–84.

Xie, X. S. & Wünsche, B. C. (2010), 'Efficient contour line labelling for terrain modelling'. [Submitted for publication].

Yang, R. (2009), Life sketch - a tool for sketch-based modelling and animation, Master's thesis, Department of Computer Science, University of Auckland, Auckland, New Zealand. (to be published).

Zeleznik, R. C., Herndon, K. P. & Hughes, J. F. (1996), SKETCH: An interface for sketching 3d scenes, *in* 'Proceedings of SIGGRAPH '96', ACM Press, pp. 163–170.

Zimmermann, J., Nealen, A. & Alexa, M. (2008), 'Sketching contours', *Computers & Graphics* **32**(5), 486–499. http://www.cs.rutgers.edu/~nealen/research/sc_preprint.pdf.