Automatic Joint and Skeleton Computation for the Animation of Sketch-based 3D Objects

Rong Yang
Graphics Group, Department of Computer Science
University of Auckland,
Private Bag 92019, Auckland 1142, New Zealand,
Email: wizard.yang@gmail.com

Burkhard C. Wünsche
Graphics Group, Department of Computer Science
University of Auckland,
Private Bag 92019, Auckland 1142, New Zealand,
Email: burkhard@cs.auckland.ac.nz

***Abstract*—Animated models are essential for simulations and virtual worlds. In many applications approximate models are sufficient and an efficient model creation and animation suitable for untrained users is required. Sketch-based modelling has been shown to be a suitable interface for creating such models because the underlying pen-and-paper metaphor is intuitive and effective. However, there is no similarly easy process for animating these models.**

In this paper we present an automatic skeletalisation and rigging algorithm for sketch-based models. Our algorithm analyses sketched contours and creates fully automatically a hierarchical skeleton with joints. The surface mesh is bound to the curved skeleton bones using skinning techniques and the resulting model can be animated using skeletal animation techniques.

Our analysis and user evaluation suggests that the joints placements are perceived as natural. The models can be animated using traditional skeletal animation techniques such as key-framing and motion capturing, or can be used as input to physically-based animation techniques and evolutionary algorithms.

I. Introduction

Animated 3D models are essential for many applications in science, engineering, education, medicine and arts. In many instances a rough approximation of an object's shape is sufficient and easy model creation and animation is more important than physical realism. Examples are storyboards for animation production [1] and the initial stages of design processes. In medical imaging and scientific computing, rough prototypes (frequently termed *templates* or *default models*) are used for segmentation, feature recognition and object tracking [2], [3]. Approximate models are also useful for demonstrating basic concepts, e.g., in education [4].

Creating models and animations with sketches is particularly attractive since it encourages creativity [5] and enables users to concentrate on the overall problems rather than details [6]. The past decade has seen a tremendous increase in the design and use of sketch-based interfaces.

In this paper we present an automatic skeletalisation and rigging algorithm for models created using contour-based approaches such as Igarashi et al.'s famous "Teddy" system [7]. The main contribution of our work is an animation system which automatically detects movable parts and enables skeletal animation based on curved bones.

Section II reviews previous work on animating sketch-based objects. Section III presents the design of our system. We evaluate our tool in section IV and conclude the paper with section V.

II. Literature Review

Sketch-based modelling systems for 3D objects 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.

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, often referred to as contour, can be 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. This assumption can be relaxed by allowing the user to draw additional sketches to indicate the shape of the cross-section.

The best known system in this class is Igarashi et al.'s "Teddy" application [7]. 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 [8], smoothing the underlying skeleton [9], or for modifying the shape by sketching contours of local features [10].

Karpenko et al. use implicit surfaces to "inflate" contours to 3D bodies. As a result different sketched components can be easily blended together [11]. Similar ideas are employed in ShapeShop [12] and MIBlob which use implicit surfaces to inflate contours traced in medical images [13]. Other authors have shown that complex 3D objects can be edited using stylus strokes that retrace an object's silhouette [14], [15]. The modification of a models silhouette subsequently rescales it so

978-1-4244-4698-8/09/$25.00 ©2009 IEEE

that it remaps itself to the new silhouette. Sketched-contours can also be used to deform 3D templates in order to create new models [16].

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. Motion Doodles [17] allow the user to sketch a motion path for a sketched character which can consist of up to seven components with predefined functionalities, i.e., the algorithm is not suitable for general sketched objects. Oshita and Ogiwara animate crowds by sketching example motion paths for some characters and using them to estimate crowd parameters such as moving speed and crowd regularity [18].

Davis et al. [19] and Mao et al. [20] achieve animations of articulated 3D characters by creating 2D sketches of the character in key frame poses. Igarashi et al. [21] 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" [22]. The user can animate a shape by selecting arbitrary points within it and moving them. The 2D shape is deformed by triangulating it and computing a configuration containing the moved points (and corresponding 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 other sketch-input devices (mouse, tablet and pen).

III. System Design

We use a sketch-based 3D model generated with a contour-based approach. For simplicity we generate our model with the "Teddy" algorithm. However, any contour-based method where the final 3D shape corresponds to the originally sketched contour is suitable. For animation purposes we 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 [23].

The problem we hence have to solve is to create a hierarchical skeleton model with bones and joints from a sketched contour.

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

1) Model and spine generation
2) Skeleton generation - bones and joints
3) Skelelal animation and skinning setup

A. Model and Spine Generation

The sketch-based model is created using the "Teddy" algorithm [7]. The user sketches a 2D outline of the shape which is sampled. The sample points 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 mid-points of internal edges and centroids of junction triangles. Small side branches of the skeleton are eliminated by using a pruning operation [7]: Starting from the terminal triangle, triangles are merged until the merged 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. The complete details are given in [24].

B. 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 figures 1 and 2. So far all nodes and bones in the spine tree are rigid. In the next step we will add joins to the spine tree. The joins will be selected from spine nodes directly

connected to branch nodes (i.e., they lie on the internal edges of junction triangles).

Marr and Nishihara [25] 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 [26]. The results approximate how a piece of paper of the sketched shape can be bent. 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 1 and 2. Whereas the folding axis in the middle of the torso of figure 1 is still acceptable, the axis separating the right shoulder from the body is unintuitive. The same problem occurs for the folding axes separating groups of two and three fingers in figure 2.

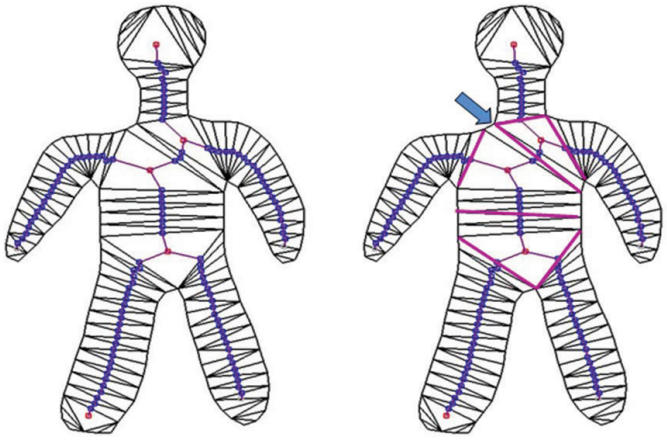

Fig. 1. Left: the chordal axis (blue) and branch nodes (red) of a sketched contour of a doll. Right: Folding axis (red lines) constructed using a paper metaphor from [26].

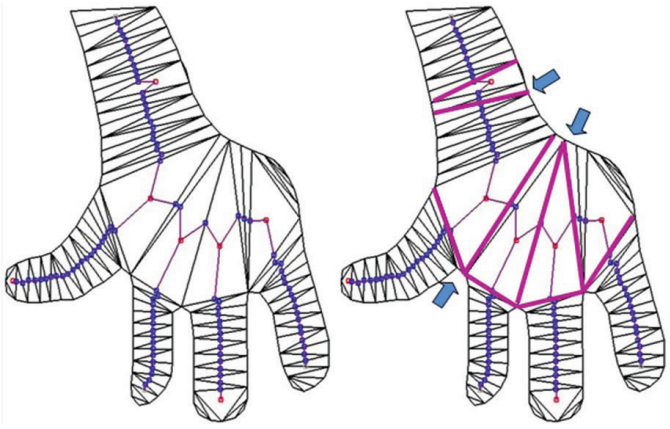

Fig. 2. Left: the chordal axis (blue) and branch nodes (red) of a sketched contour of a hand. Right: Folding axis (red lines) constructed using a paper metaphor from [26].

Since we have no information about the semantic of an object and its physical meaning we cannot guarantee physically correct joints. Instead we want to create joints which are perceived as natural and plausible. The results above indicate that this can be achieved by finding a subset of folding axes, where the user perceives all elements as clearly separated from each other.

We achieve this by clustering folding axes according to their distance within the triangulation: 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 the edge nearest to the parent spine node is a joint. The algorithm is illustrated in figure 3. The five red dots in the image on the bottom right are the final joints of the skeleton. Note that by using the circumcircles of junction triangles we implement a relative distance criteria. This is preferable to using an absolute value which would not work if an object has features of strongly varying size.

Many advanced sketch-based modelling implementations allow the user to model objects by sketching different shapes and combining them. In this case we will have separate spine trees which are joined where the shapes are combined. Since drawing components separately usually indicates functional or physical separation, we suggest in this case to always use a joint at the connection point.

C. 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 [27]. 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 the 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 during animation linear blend skinning results in gaps between or overlaps of the bones' surfaces. 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. For example, if a vertex has an equal distance to two bones then its weights are 0.5 for each bone. In our case distance

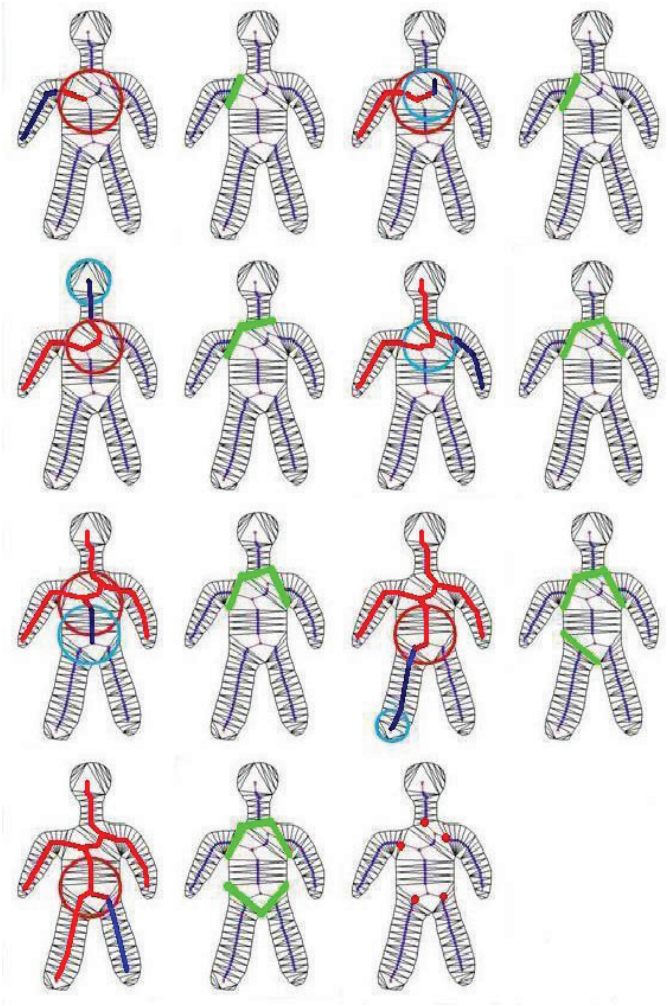

Fig. 3. The joint detection algorithm for finding spine nodes which divide the sketched contour into separate parts for animation. The algorithm traverses branches (red and blue lines) in the spine tree. Each branch connects two branch nodes or one branch node to a leaf. Each step tests the intersection (red and light blue circle) between two adjacent branch nodes' circumcircles. If there is no intersection between two circumcircles 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.

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 (undeformed shape) with respect to bone k. The position v of the deformed vertex is computed by [27]:

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

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.

IV. Results

A. 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 4. 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.

B. Correctness

All models we created were plausible with no major artifacts such as holes or non-manifold surfaces. Figure 4 demonstrates that most detected joints are meaningful. In particular note that the egg (stone) has no joints even though its triangulation contains junction triangles. Very few, if any, of our examples have "natural" joints missing. In Figure 4 (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 crocodile in part (h) of the figure should have joints for each claw, but some claws were drawn so short that the corresponding junction triangles were eliminated by the pruning algorithm. The top joint of the desk lamp in part (j) of the figure is not placed correctly. This is due to an abnormality in the sketched contour. Some users might expect the eyes of the lobster (c) to be movable. Joints would have been created if the eyes were more protruding as is the case for the dragonfly (i).

The algorithm produces some unexpected extra joints, e.g., the mouth and the left elbow of the lobster, the mouth and body of the deer, the sides of the lamp shade, 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 too 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.

Figure 4 also shows that the root of the spine tree (indicated by a red dot) is usually in the component which would be

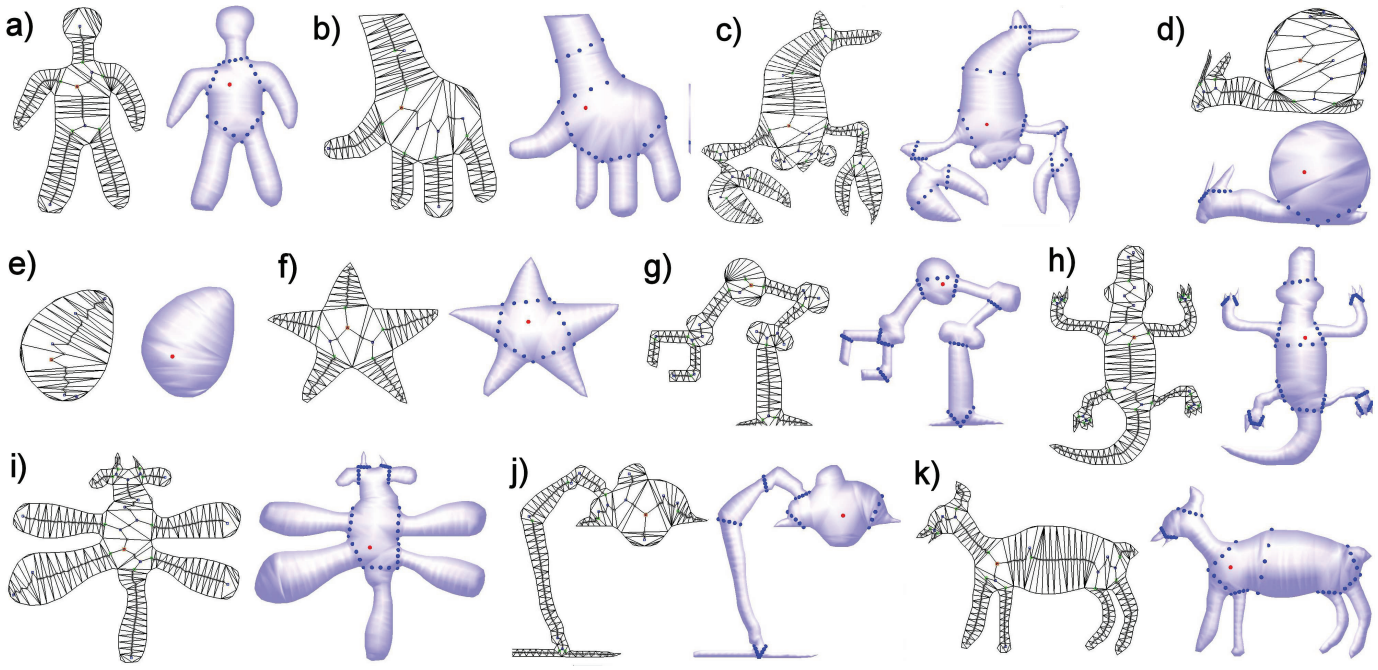

Fig. 4. Examples of sketched models: (a) doll, (b) hand, (c) lobster, (d) snail, (e) egg/stone, (f) sea star (star fish), (g) robot, (h) crocodile, (i) dragonfly, (j) desk lamp, (k) deer. Each image shows the sketched contour and resulting skeleton (left), and the Gouraud shaded 3D model (right). The 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.

naturally perceived as body or base of the object. Notable exceptions are the robot (g) and the desk lamp (j).

C. User Experience

We conducted a user study with 11 participants (8 male, 3 female) and asked them to draw the doll and the hand model shown in figure 4. Users were also able to rotate object components around the joints as indicated in figure 5. We asked users whether the joint positions are what they expected and recorded responses using a seven-level Likert scale ("strongly disagree" (-3) to "strongly agree" (3)). The mean response was 0.91 for the hand model and 2.27 for the doll model. The standard derivations were 1.64 and 0.65, respectively. The subsequent interviews showed that for the hand model the joint positions were perceived as intuitive but six participants thought that the rotation of the mesh around joints is not intuitive. The main problem was that all joints are classified as ball joints (3 degrees of freedom), whereas finger joints are hinge joints (one degree of freedom). This allows unnatural movements and makes animation control more difficult.

Automatically detecting the type of joint is an interesting and important task for automating the animation process. Methods similar to Xu et al. could be used to this [28]. The authors analyse joint constraints in 3D models in order to support the direct manipulation of an arbitrary mix of rigid and deformable components.

D. Applications

The skeleton constructed with our algorithm can be used in different ways to animate sketch-based models efficiently:

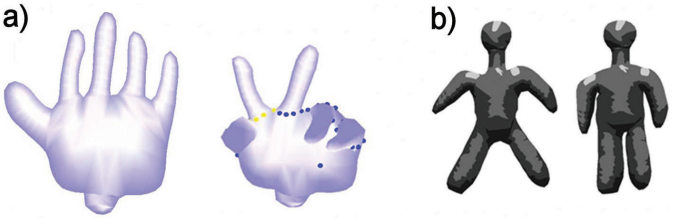

Fig. 5. Animation of sketched objects: (a) Hand-model fingers rotated around the automatically detected joints. (b) Doll model with limbs rotated around the automatically detected joints.

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 [29]. 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 a locomotion controller computed with an evolutionary algorithm [30], [31]. This means the modelled shape learns to walk. 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 [23].

Finally a sketched object can be animated efficiently using sketch input as explained in subsection II. 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.

V. Conclusion and Future Work

Sketch-based modelling and animation is an exciting technology with a wide range of applications. We have presented a skeletalisation and skinning algorithm which automatically computes a hierarchical skeleton model from a sketched object and binds mesh vertices to its bones. 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.

Our evaluation and user study demonstrated that the resulting models and joint positions are perceived as natural and intuitive. More research needs to be done in order to classify joints automatically (degrees of freedom) and in order to reduce the number of redundant joints.

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 physically-based animation system based on ODE [29].

References

[1] J. Hart, *The Art of Storyboard: Storyboarding for Film, TV, and Animation*. Focal Press, 1999.

[2] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models—their training and application," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 38–59, 1995, http://www.isbe.man.ac.uk/~bim/Papers/cviu95.pdf.

[3] V. Lepetit and P. Fua, "Monocular model-based 3d tracking of rigid objects: A survey," *Foundations and Trends in Computer Graphics and Vision*, vol. 1, no. 1, pp. 1–89, October 2005.

[4] Kloonigames Ltd., "Crayon Physics Deluxe homepage," 2008, http://www.crayonphysics.com.

[5] M. D. Gross and E. Y.-L. Do, "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, 1996, pp. 183–192.

[6] Y. Y. Wong, "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, 1992, pp. 83–84.

[7] T. Igarashi, S. Matsuoka, and H. Tanaka, "Teddy: a sketching interface for 3d freeform design," in *Proceedings of SIGGRAPH '99*. ACM Press, 1999, pp. 409–416.

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

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

[10] J. Zimmermann, A. Nealen, and M. Alexa, "Sketch-based interfaces: Sketching contours," *Computers & Graphics*, vol. 32, no. 5, pp. 486–499, 2008, http://www.cs.rutgers.edu/~nealen/research/sc_preprint.pdf.

[11] O. Karpenko, J. Hughes, and R. Raskar, "Free-form sketching with variational implicit surfaces," *Computer Graphics Forum*, vol. 21, no. 3, pp. 585–594, Jul. 2002, http://www.merl.com/papers/docs/TR2002-27.pdf.

[12] R. Schmidt, B. Wyvill, M. C. Sousa, and J. A. Jorge, "Shapeshop: sketch-based solid modeling with blobtrees," in *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*. ACM, 2006, p. 14.

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

[14] V. Cheutet, C. E. Catalano, J.-P. Pernot, B. Falcidieno, F. Giannini, and J.-C. Léon, "3d sketching for aesthetic design using fully free-form deformation features," *Computers & Graphics*, vol. 29, no. 6, pp. 916–930, 2005.

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

[16] V. Kraevoy, A. Sheffer, and M. van de Panne, "Modeling from contour drawings," in *SBIM '09: Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*. ACM, 2009, pp. 37–44, http://people.cs.ubc.ca/~van/papers/2009-SBIM-contourDrawings.pdf.

[17] M. Thorne, D. Burke, and M. van de Panne, "Motion doodles: an interface for sketching character motion," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 424–431, 2004, http://people.cs.ubc.ca/~van/papers/doodle.html.

[18] M. Oshita and Y. Ogiwara, "Sketch-based interface for crowd animation," in *SG '09: Proceedings of the 10th International Symposium on Smart Graphics*. Springer-Verlag, 2009, pp. 253–262, http://www.oshita-lab.org/paper/sg09.pdf.

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

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

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

[22] ——, "As-rigid-as-possible shape manipulation," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 1134–1141, 2005, http://www-ui.is.s.u-tokyo.ac.jp/~takeo/research/rigid/index.html.

[23] I. Baran and J. Popović, "Automatic rigging and animation of 3d characters," *ACM Trans. Graph.*, vol. 26, no. 3, p. 72, 2007, http://www.mit.edu/~ibaran/autorig/.

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

[25] D. Marr and H. K. Nishihara, "Representation and recognition of the spatial organization of three-dimensional images," in *Proceedings of the Royal Society of London, Series B*, vol. 200, 1978, pp. 269–294.

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

[27] J. P. Lewis, M. Cordner, and N. Fong, "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., 2000, pp. 165–172.

[28] W. Xu, J. Wang, K. Yin, K. Zhou, M. van de Panne, F. Chen, and B. Guo, "Joint-aware manipulation of deformable models," *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2009)*, vol. 28, no. 3, pp. 1–9, 2009, http://people.cs.ubc.ca/~van/papers/2009-TOG-jointAware.pdf.

[29] R. Smith, "Open Dynamics Engine home page," 2007, http://www.ode.org.

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

[31] M. Sanders, R. Lobb, and P. Riddle, "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, 2003, pp. 255–256.