Gesture-based design of 2D contours: an alternative to sketching?

Thomas Delame, Jean-Claude Léon, Marie-Paule Cani, Renaud Blanch

EUROGRAPHICS Symposium on Sketch-Based Interfaces and Modeling (2011)
T. Hammond and A. Nealen (Editors)

Gesture-based design of 2D contours: an alternative to sketching?

T. Delamé[1] & J.C. Léon[2] & M.P. Cani[2] & R. Blanch[3]

[1]Laboratoire Le2i - Burgundy University
[2]Laboratoire Jean-Kuntzmann - Grenoble Universities & INRIA
[3]Laboratoire d'Informatique de Grenoble

Abstract

In addition to being a very expressive media, 2D sketches representing the contour of a shape are commonly used as a basis for 3D sketch-based modeling. This paper investigates an alternative to the standard way of creating such sketches: instead of carefully following the contour with a pen and erasing or over-sketching, the user progressively shapes the contour from a simple input curve, only through intuitive deformation gestures. No menus or sliders are used. This is achieved by providing an automatic selection mechanism between a minimal set of deformation operators, inspired from Michael Leyton's perceptual theory of shapes. The shape representation and the active operator parameters are kept transparent to the user. This enables user to focus on the design and makes the system immediately usable by anybody. We validate this new paradigm through a user study that includes a comparison with standard sketching.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation H.5 .2 [Information Interfaces and Presentation (e.g., HCI)]: User Interfaces—Graphical user interfaces (GUI)

1. Introduction

In standard modeling software, most of the available tools for creating shapes rely on parametric curves and surfaces. Based on specific interfaces to edit the parameters of the underlying geometric model, these tools cannot easily be used by unprepared users. Even trained digital artists such as those working for the film industry, tend to first draft and refine a new shape with pencil and paper, rather than creating it directly with a computer. Indeed, sketching remains the easiest way for them to freely express what they have in mind and iteratively refine it through trial and error.

Sketch-based interfaces try to bring to the digital media the natural interaction we have with pencil and paper. They have recently been very successful. They have been developed both for creating 2D drawings and as a basis for inferring 3D models. In particular, 3D digital design was proposed, for the first time, to general public thanks to sketch-based modeling tools [Ske].

In this paper, we would like to take a step back and study the way a user creates and refines a closed 2D contour. In real life, we have nothing more convenient than pencil and paper to create such 2D shapes. Is it still the case with a computer? Is the standard pen-based metaphor the best one towards the incremental and intuitive design a user is seeking? Can we do better with gesture-based deformations, in terms of easiness, speed and user satisfaction?

To answer these questions, we present a purely gesture-based 2D deformation system, dedicated to the intuitive editing of 2D contours. For an easily comparison with sketching: this system requires knowledge of the underlying geometric model, no parameter tuning. It is immediately usable by any unprepared user. Our first contribution is an extension of Michael Leyton's perceptual theory of shapes [Ley88], to express the dual process of iterative shape design. This leads us to a representation of 2D contours based on their symmetry axes, associated with a new family of editing operators. Our second contribution is the way this set of operators are automatically activated and tuned, based on user expecta-

tion, inferred through their interaction gesture. We validate this method through a user study that compares the resulting gesture-based design method with standard sketching of the same target shapes, and we conclude.

2. Related work

This paper focuses on 2D contours. However, We review methods introduced for designing both 2D and 3D shapes since many techniques developed for 3D models can also prove relevant for editing 2D shapes.

The oldest shape modeling methods in Computer Graphics are model-based: they provide interfaces for directly editing the degrees of freedom of a given model. Such interfaces were developed for spline curves and surfaces, implicit metaballs and subdivision surfaces. Still now, many curve modeling tools require the user to edit control points and tangents. Adding extra degrees of freedom requires some explicit interaction with a scissor tool. This is very far from the natural interaction a user is looking for. Thus, most recent work in shape design either rely on sketching or sculpting metaphors, which is closer to the way humans really create shapes.

2.1. Sketching Metaphors

Sketching digital curves or surfaces like human do with real pencil and paper is now a mature area [Alv07], [JS09]. Whatever the dimension of the shape to design, the user typically starts by sketching a 2D contour, edited either by erasing or through over-sketching. Although sometimes used as they are [IMT99], the resulting free-hand curves are generally approximated by a smoother representation such as B-Splines [BCD01], clothoids [MS09], or implicit curves [KHR02], [SWSJ05], [BPCB08].

These methods are not well suited for quickly draft and then improve a complex contour: first, an optimization process is generally used to approximate the user's input, which leads to some global re-computation after each edition. Second, in these systems editing is typically reduced to an undo-redo process: for instance, if the user sketches the detailed profile of a character and simply wants to bend the head a bit more, the only solution he has is to erase and re-draw the head from scratch.

2.2. Sculpting Metaphors

An alternative to sketching a contour is to shape it by progressively deforming a simple shape, as if it is made of clay [CIW08], [ISE]. Among the resulting sculpting methods, space deformation is one of the most attractive: it acts by defining a deformation field in the space in which the shape is embedded. It can be applied to arbitrary shape models and extended to intuitive, constant-volume deformations [GB08]. The main drawback comes from the fact that the deformation range of influence is defined independently from the embedded geometry: for instance, if the edited shape is the contour of a human hand, shortening a finger without affecting the neighboring ones is difficult. To avoid this problem, variational methods such 'as rigid as possible' deformation [IMH05] and Laplacian-based editing [Sor06] directly generate a deformation field at the surface of the shape to be edited. Based on feature-preserving shape optimization, these methods give impressive results. We, however, do not use them in this work since studying user-expectation leads us to some more direct control on the range of deformation to be associated with each editing gesture. Our method, we present in the next section, belongs to sculpting metaphors and is built on a perceptual theory of shapes.

In the remainder of this paper, a *shape* is a smooth 2D closed contour.

2.3. Michael Leyton's perceptual theory of shapes

M. Leyton, who is interested in human shape understanding [Ley88], proposes a new way to look at shapes in terms of modification/deformation. He claims humans perceive objects by mentally creating a history of its construction from the simplest possible shape, i.e. a circle. He developed a minimal grammar of Deformation Processes (DPs), able to progressively simplify any shape back into a circle. M. Leyton's theory relies on the evolution of an intrinsic property of curves: their curvature. Each operator of his grammar [Ley88] starts at a curvature extremum and acts along symmetry axes and transforms the *curvature word* of the shape, i.e. sequence of curvature extrema identified by a letter, 'm' for minimum and 'M' for maximum, and their sign (see Fig. 1). These symmetry axes, called Process-Inferring Symmetry Analysis (PISA) axes are attached to a curvature extremum. They are defined by the set of points Q, midpoints of arcs AB of circles tangential to the shape at points A and B. These axes are not well defined because, for example, there are two arcs AB, thus two midpoints and their trajectories do not form a continuous axis.

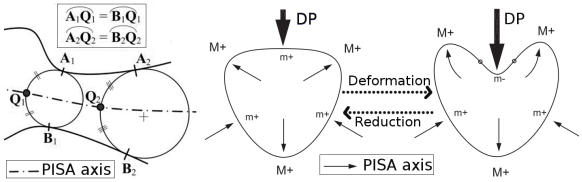

Figure 1: *a) definition of a PISA axis, b) a deformation operator called 'continuation at m^+' applied to the word: $m^+M^+m^+M^+m^+M^+$.*

From a dual, constructive viewpoint, which is more relevant to us, these symmetry axes can be interpreted as the footprints of the DPs which shaped the current contour. It gives us the intuition we were looking for, to achieve natural, gesture-based shape design, i.e. symmetry axes are gestures.

3. Toward purely gesture-based deformations

Let us come back to our goal, from a user-centered perspective. We would like to compare 2D sketching with another intuitive media for humans: deformation gestures. As we have just seen, all previous work on sculpting-like deformation requires the user to choose at least the size and range of influence of a deformation tool or to explicitly express geometric constraints to be met. In contrast, our goal is to let any unprepared user interact with the designed shape through natural gestures. The effect and range of influence of each action will be inferred from his expectations.

We need a well founded methodology to achieve this. To this end, we firstly extend Leyton's theory to build a dual, constructive theory, expressing the way a human predicts the effect of a distortion gesture. This leads us to a set of constructive deformation operators and enables us to design user interactions. Our technical solutions for the shape representation and the real-time deformation operators will be presented in Sec. 4.

3.1. Constructive Theory of Shape

Although introduced to simplify shapes back to a circle, M. Leyton's theory can serve as an inspiration towards our goal. As M. Leyton stated, symmetry axes can be seen as the footprints of DPs. This leads us to the following interpretation: gestures will start at curvature extrema and their trajectory will define the symmetry axes. This way we can reproduce Leyton's grammar operators.

This simple interpretation is appealing, but not sufficient for constructive shape design. Firstly, the user would not be able to freely interact with the shape, as the grammar is minimal and initially dedicated to simplify a shape. Thus, some operators are lacking either because they are not needed to simplify the shape or they can be obtained by composing other operators. Secondly, M. Leyton does not consider the curvature intensity whereas different shapes can have the same curvature word, but exhibit highly different appearances. Thirdly (the most important one): curvature extrema points cannot suffice to drive deformation operation! People do not notice curvature equally. Curvature extremum points seem not to be the key points of a shape [Sch06]. And users may want to interact with their shape elsewhere than at a curvature extremum or modify a curvature amplitude while preserving the curvature extrema [CCP*05]. We choose instead to consider the DP in symmetry axis terms. Our deformation grammar is thus set as follows:

- **Structuring operators** give the shape its main appearance by adding new local symmetry axes as M. Leyton's operators do. They can be seen as the action of adding/removing material to the shape,
- **Posturing operators**, change the shape pause by extending, reducing or bending existing symmetry axes. This category complements the lack observed in M. Leyton's set of operators, and put the current shape into a given posture, without changing the perception that this is the *same* shape.

We validated this theory on a small set of 15 users. We showed them two categories of figures on a paper (see an example in Fig. 2). The first one contained two shapes. The users were asked to draw the footprints of the gestures they perceived to evolve from one to the other. The second one corresponds to a shape with arrows that indicate deformation gestures. We asked the users to sketch the deformed shape they expected. This test validate well the statements above: the deformation gesture was a symmetry axis in both cases for structuring operators, and users do not all understand posture operators when a symmetry axis was not drawn inside the shape.

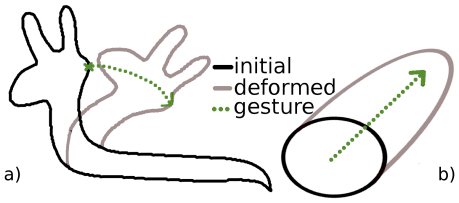

Figure 2: *Example of test figures. a) the user was asked to draw the gesture footprints of the DP leading to the deformed shape, b) here, the deformed shape had to be specified given the DP.*

About the selection of structuring vs. posturing operators, a first result is: if the deformation gesture started on the contour itself, it is generally interpreted as indicating a posture change. A second very interesting result is about the range of influence of these sketched deformations. A posturing deformation was understood as if it only influenced the area of the shape defined by the edited axis. Concerning the structuring operators, their range of influence changed with the deformation gesture of the initial position: the closer to the contour, the more local the deformation.

This preliminary user-study led us to several conclusions: it validated our methodology, confirming that the geometric model we need has to be defined from a set of local symmetry axes, and must accept the set of deformation operators mentioned above. Secondly, it gave us some clues on user expectations during interaction, which leads us to design interactions as follows.

3.2. Interaction Design

To design an interactive gesture-based system, with an interface free of any menu or button, all operators have to be activated and tuned automatically. In our case, this is achieved through user's interaction and from the morphology of the current shape, using the methodology we have just presented. This makes the modeling process predictable and, therefore, reduces the learning curve.

In practice, the user starts his design with a simple shape, an ellipse (chosen so that a first symmetry axis is already defined). Then, he/she freely applies deformation gestures that progressively change either the structure or the posture of the shape, in any order. Each user's gesture is interpreted depending on its deformation starting point.

We chose to activate a structuring operator whenever the user starts his gesture inside the shape. In this case, a new symmetry axis is created along the trajectory of the gesture. The 'thickness' or deformation extent of the shape modified area along this symmetry axis is derived from the depth of the cursor initial position inside the shape. If the cursor is outside, we want the system to remove material from the shape. However, the results are not straightforwardly usable: symmetry axes outside the shape are unintuitive to most people. Consequently we consider in this paper that no such operator is activated.

When the tool is on the contour, a posturing operator is activated. If it is very near the contour, a snapping mechanism (similar to a magnetic effect) is used to attract the tool on the contour. This saves time and effort, making careful positioning unnecessary: a technique well suited for unskilled public. Note that it is not required to set the tool radius of influence, because the edition of one of the symmetry axes defining the shape will necessarily affect its corresponding part.

4. Shaping a Contour Through Intuitive Deformations

This section describes our technical solution to implement the constructive theory and the interaction design we exposed in the previous section.

4.1. Shape Geometric Model

The geometric model used is convolution implicit surfaces defined with a piecewise linear skeleton. Implicit surfaces are the level set of a function F: the shape is the isocontour $S = \{p \mid F(p) = l\}$, for an isovalue l. Here, the convolution skeleton, Sk_c, is a geometric source of potential, emitting in every direction a value that decrease with the distance. $F(p)$ is the sum of the contributions $f(q)$ of all points q on the skeleton: $F(p) = \int_{q \in Sk_c} f(q)dq$. We use convolution weights to finely tune the thickness of the isocontour around the skeleton, which reflects the amount of potential emitted by those points q.

Sk_c is planar and piecewise linear, thus it can be represented by a planar graph $G(N,E)$. Each vertex n_i has a weight w_i which represents its convolution weight. The convolution weight of a point q that belongs to an edge (n_i, n_j) is obtained by linear regression between w_i and w_j.

A convolution skeleton is an approximation of a topological skeleton of a shape, which is a connex – much more user intuitive–, robust and well-known symmetry set. In [BPCB08], the initial convolution skeleton is even computed by a topological skeleton extraction algorithm. The topological skeleton is the set of all inner maximal disks centres, and it is used in place of Leyton's symmetry set since PISA axes do not behave continuously (see Sec. 2.3). We use convolution skeletons to take advantage of the underlying model, which produces smooth contours. The weight w_i of a point $q_i \in Sk_c$ is used as if it is the radius r_i of the corresponding point s_i on the topological skeleton. This relationship must be maintained throughout the user's interactions since these two models differ.

The skeleton of the implicit model, the topological skeleton and the PISA axes don't coincide. But their coincidence is not mandatory in the present context. As far as we could investigate, they are all similar in the sense that their number of branches is identical and their associated contours behave similarly, i.e. curvature extrema have the same distribution along the contour (see Fig. 3). The similarity between skeletons is a key feature to preserve their meaning during DPs. This similarity context represents the current framework set up throughout this paper to evaluate the interest of shaping by gesture. A more robust consistency handling any kind of contour configuration will be addressed in the future.

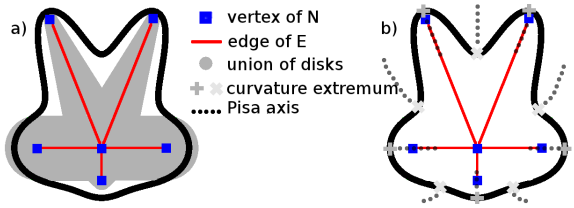

Figure 3: *Shape representation model. The contour, is the only entity seen by the user. a) Sk_c and the corresponding union of disks if we consider the topological skeleton associated with Sk_c. b) detection of all curvature extrema thanks to the branching structure of the skeleton and a gradient technique, and their PISA axes.*

4.2. Structuring Operators

Structuring operators give the shape appearance. To do so, they modify the convolution skeleton by adding a new branch that is a sub-skeleton $\mathfrak{B} = \{(b_i), (b_i, b_{i-1})\}$ to Sk_c. The weight of $b_i \in \mathfrak{B}$ is w_i.

Setting up the weights

To assign weights w_i, the concept of 'dimension at a point' is introduced. Every point p in the plane has a weight equal to its distance to a point c of the contour reached by a gradient technique: c is the first point of the series $c_{i+1} = \varepsilon_i \times \nabla F(c_i) + c_i$ very close to the contour, i.e. $\|F(c) - l\| \ll 1$, l is the isovalue (see Sec. 4.1). The sign of ε_i ensures the convergence toward the contour depending on whether the point c_i is inside the shape or not, and its absolute value is reduced each time the segment $[c_{i+1}, c_i]$ crosses the contour.

This assignment gives very good results to define the maximal deformation 'thickness' ρ_M of the operation, i.e. $\rho \leq \rho_M$ defines the dimension of a deforming tool, see Fig. 4. Throughout this shaping process, w_i assigned to b_i decreases such that farther segments generated in $\mathfrak{B}$ minimize the risk of interference with features created earlier. Among rather complex weight assignment laws, a simple decay produces fine results, i.e. whenever a new b_{i+1} is inserted in $\mathfrak{B}$ to fit the user's gesture, $w_{i+1} = r \times w_i$, with $r \leq 1$ (0.9 in our implementation).

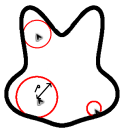

Figure 4: *Example of initial weights w_0. The tool size, is continuously displayed to the user by a circle centered at the tool position, i.e. the cursor. Its initial size is computed by our 'dimension of a point' mechanism.*

Capturing structuring gestures

Assuming the tool is at the nearest position b_k, the gesture should be captured through other positions b_{k+j}. The evaluation time of F function is linear with respect to the number of edges in Sk_c. Thus, for interactivity purposes, we need to keep this number as small as possible. To this end, unnecessary edges are filtered while capturing well the gesture.

The basic principle is as follows: $\vec{d} = -\vec{\nabla}F(b_k)$ is computed to define the direction where the shape bears minimal changes. Now, b_{k+1} is attached to the tool and moves with it, forming the edge (b_k, b_{k+1}). When b_{k+1} reaches a distance greater than w_k from the half-line $[b_k, b_k + \vec{d})$, b_{k+1} is anchored at that position and a new vertex b_{k+2} is attached to the tool (see Fig. 5). This process continues until the user's gesture stops.

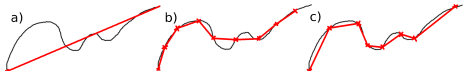

Figure 5: *Capturing a user's gesture with a skeleton branch. The tool trajectory is approximated by a) its ends points, b) equidistant sampling, c) our method.*

Feeling the shape

To enhance the user's sketching experience and help him/her during this interaction, the user's interface behaves directionally to guide him/her in the $\vec{d}$ direction. To do so, the tool position is projected onto an egg-like shape as illustrated in Fig. 6. When performing this projection at every tool displacement, the user feels the shape resisting deformation in directions orthogonal to $\vec{d}$. It produces a pseudo-haptic feeling since the egg-like shape varies the control/display ratio. This principle erases noise at the tool positions due to the hardware or to hand tremor (with elderly people for example). The shape of the egg at its rear, where the distance $\|b_i - b_{i+1}\|$ is divided by 2, helps the user to finely correct the deformation.

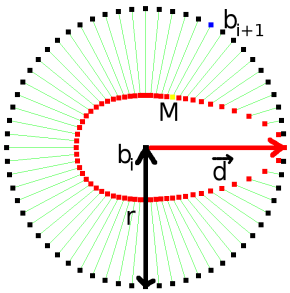

Figure 6: *Projecting the tool's position onto the egg-like shape of Hugelschaffer.*

Connect a branch to Sk_c - hierarchy

When a gesture ends, $\mathfrak{B}$ needs be connected to Sk_c to build a connex skeleton. Key points $b_i \in \mathfrak{B}$ become vertices of Sk_c. Generating this connection is also an opportunity to build a hierarchy: each edge (b_i, b_{i+1}) is numbered by its rank r_i reflecting its contribution to the appearance of the shape. The larger r_i, the more predominant (b_i, b_{i+1}). Setting up this hierarchy cannot be achieved a posteriori, otherwise the shape history would be lost. We have noticed that people set up the most important areas prior to insert fine shape details. Consequently, connecting $\mathfrak{B}$ to Sk_c gives us the level of detail of this whole branch.

To connect $\mathfrak{B}$ to Sk_c, a good anchor point q is mandatory such that the generation the edge (b_0, q) has a minimal visual effect on the shape. A first candidate for q is the point q_{grad} defined by a gradient technique similar to the one exposed before to project b_0 on Sk_c. Generally, q_{grad} cannot coincide with an existing vertex of Sk_c. Thus when we connect $\mathfrak{B}$ at q_{grad}, it splits an edge of Sk_c, increasing Sk_c complexity unnecessarily. Consequently, nearest candidates, $q_{nearest} = \arg\min_{n_i \in Sk_c} \|b_0 - n_i\|$, or q_{union} which is the vertex Sk_c that minimizes the divergence from the topological skeleton for example, are candidate vertices. The selected candidate is the one that best suits the link between the implicit model and the corresponding union of disks *and* such that the edge (b_0, q) do not disturb too much the shape contour.

Once q is defined, w_0 and w_q are compared. If $\|b_0 - q\| > w_0$, then the operation starts at a location closer to the contour than to Sk_c and adds a smaller protrusion to the shape than the ones induced by edges of Sk_c incident at q. In this case, $\mathfrak{B}$ adds detail to Sk_c and gives to all edges of $\mathfrak{B}$ a rank smaller than the maximal rank r_{MAX} of the incident edges at q; key points b_i become nodes $n_k \in Sk_c$. If not, all the edges of $\mathfrak{B}$ are assigned a rank equal to r_{MAX} (see Fig. 7).

Fig. 7 illustrates a hierarchy obtained after an interaction

during our user study. The ears are the results of a DP starting closer to the contour than to Sk_c. Thus, it is assigned a rank smaller than the rank of the neck, setting the ears as details.

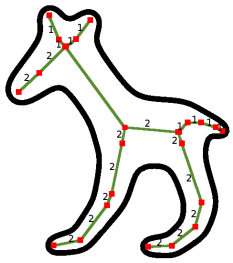

Figure 7: *Building a hierarchy in the skeleton.*

4.3. Posturing Operators

Posturing operators change the pause of a shape. Indeed, they modify a branch of Sk_c, $\mathfrak{B} = \{(b_i),(b_i,b_{i-1})\}$ corresponding to the area of the contour which is attached to the cursor. The weight of $b_i \in \mathfrak{B}$ is w_i.

Extraction of the Characteristic Branch

When the cursor coincides with a point p of the contour, the extraction of a branch $\mathfrak{B}$ which corresponds to the area of the shape near p is realized. The problem is to find the corresponding level of detail the user wants, i.e. p is located on a protrusion, belongs to the nostril of a nose which in turn belongs to a face...: which extent should be modified?

This level of detail is obtained with the hierarchy described before and a gradient technique: p is projected onto Sk_c, using the gradient pointing towards a point of Sk_c which influences the most p. This produces a point $q \in Sk_c$, which is on an edge (n_i,n_j) (see Fig. 8a, b). If q coincides with a vertex, then the edge is the one incident at q with the highest rank and whose extremity is the closest to q. Then, the rank r of this edge (n_i,n_j) is the desired level of detail.

Now, (n_i,n_j) is expanded iteratively into a branch $\mathfrak{B}'$. Starting with either extremity of the first edges, noted s, at each step, an edge (n_k,s) is inserted into $\mathfrak{B}'$ if this edge has a rank greater or equal to r and if it is the only edge with such a rank incident to s (see Fig. 8c). Then, the second extremity becomes the source of the expansion. Now, an orientation of $\mathfrak{B}'$ is needed to set the tail t of $\mathfrak{B}'$ as an anchor while the head h is the mobile vertex. If we look at the connectivity of $\mathfrak{B}'$ extremities: h is the extremity with the smallest number of edges with $r' \geq r$ (see Fig. 8d). If there is no such extremity, h is the one with the smallest number of edges which separates q from the corresponding extremity.

Finally, h is moved to the closest point n_i or n_j such that the shape is modified only from the point q (p on the contour) until t. This leaves the area between p and h unchanged. Consequently, $\mathfrak{B} = \{(b_i \in \mathfrak{B}'),(b_i,b_{i+1})\}$, with $b_O = h$ and $b_n = t$. Our characteristic branch describes the locality of the shape around p at a given level of detail, with no other user's action than positioning the tool on the contour.

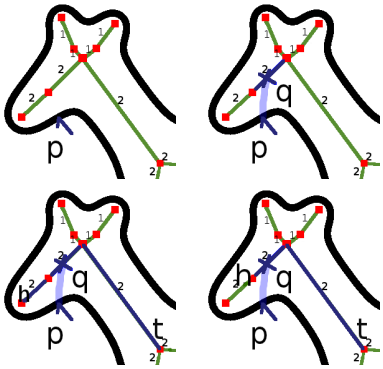

Figure 8: *Characteristic branch extraction process.*

In Fig. 8d, the posturing operator will then deform the shape such that the animal's head will raise or lower. The purpose of $\mathfrak{B}$ orientation is obvious here: the user does not want the animal's body moves around its head!

Formalisation of Posturing Operators

Modifying the posture of an object through a skeleton is a common action in computer graphics, as in skeletal animation. Here, the user should be able to achieve this transformation through a single gesture, modifying $\mathfrak{B}$ in a transparent manner. On purpose, a mechanical model derived from strength of materials discipline has been set up, generating a bending-like deformation of $\mathfrak{B}$ coupled with some extension/shrinking deformations.

Each edge of $\mathfrak{B}$ is now a beam rigidly connected to its neighbours. Each beam (b_i,b_{i+1}) has a trapezoidal section whose bases are w_i and w_{i+i}, respectively. The beam (t,b_{n-1}) is anchored: t cannot move and its section cannot rotate. When moving the cursor, the user applies a force $\vec{C}$, analog to its displacement, on the extreme beam segment (h,b_1). The corresponding deformation changes the position of h (see Fig. 9). The orientation of the subsets of Sk_c connected to h is modified, rotating all those subsets as solid bodies. In Fig. 9, the black contour is the shape before deformation. The blue one depicts the shape at an intermediate position of the user's gesture, it is a *ghost*.

Figure 9: *Changing the posture of a shape.*

Physically, it is a coarse approximation, but visually it produces very good results without expensive computations. The location of h is obtained in linear time with respect to the number of segments of $\mathfrak{B}$.

Moving Details Accordingly

Sk_c has sub-trees connected to $\mathfrak{B}$, those sub-trees must be translated and rotated to ensure the shape coherence: in Fig. 8, the mouth and the ears must follow the displacement of the neck. As mentioned before, these sub-trees are subjected to rigid body movements: all sub-trees connected to h must keep their angle and distance with the edge (h,b_1). Sub-trees connected at each $b_i \in \mathfrak{B}$ except t are subjected to a translation first, to keep their distance to $\mathfrak{B}$. Then, a rotation is applied such that each sub-tree with a root (p_k,b_i) keeps the angle ratio $\frac{\widehat{(b_{i+1},b_i,p_k)}}{\widehat{(b_{i+1},b_i,b_{i-1})}}$.

Rotations are more complex to perform due to floating-point arithmetic, but preserve the shape consistency, avoiding movement of the ears into the small angle between the neck and the mouth of the animal in Fig. 8.

Interaction

As stated at Sec. 3.2, a snapping technique helps the user to locate the tool on the contour in order to activate posturing operators. We also want the user feel the weight of the shape during the interaction: the heavier $\mathfrak{B}$, the more difficult to move $\mathfrak{B}$. However, mechanical equations already incorporate forces and stiffness.

Regarding the tool position, it is hard to compute the force $\vec{C}$ applied to $\mathfrak{B}$ such that p on the contour stays coincident with the cursor during deformation. However, $\|\vec{C}\|$ is proportional to the distance covered by the tool between two interaction loops. This helps the user feel the shape. Consequently, computing $\vec{C}$ to keep the coincidence between p and the cursor is not only hard but useless. Similarly, keeping the distance between p and the edge (n_i,n_j) such that $q \in (n_i,n_j)$ as well as the angle $\widehat{(p,q,n_i)}$ leads to unpredictable effects: the direction of the tool in the control space does not coincide with the direction in virtual space.

5. Validation

The validation of the usability of our gesture-based interactions relies on two aspects: the *a priori* respect of principles recommended by the Human-Computer Interaction (HCI) community; and the *a posteriori* validation by a user experiment.

5.1. Conformance to HCI Principles

Direct manipulation is an interaction paradigm made explicit by Shneiderman when graphical user interfaces (GUIs) came to life: users should be able to directly manipulate objects presented to them using metaphors inspired by the physical world [Shn83]. However, the manipulation is often indirect in GUIs: users manipulate instruments (e.g., scrollbars or editing tools) that, in turn, manipulate the objects of interest. This remark made Beaudouin-Lafon introduce the notion of *instrumental interaction* [BL00] and he proposed to quantify this indirection using 3 properties of intruments: their *degree of indirection*; their *degree of integration* and their *degree of compatibility*.

The *degree of indirection* quantifies 'the spatial and temporal offsets generated by an instrument'. In our case, both offsets are almost inexistent: the interaction always occurs at the place where the user wants to edit the shape; and the modification occurs incrementally as the user interacts. The *degree of integration* measures 'the ratio between the number of Degrees Of Freedom (DOFs) available in the logical part of the instrument and the number of DOFs captured by the input device'. In our case, this ratio is greater than 1, which is very good: the users not only control the 2D trajectory of the tool with a 2D pointing device, but also its mode and size. Finally, the *degree of compatibility* measures 'the similarity between the physical actions of the user on the instrument and the response of the object'. Because the shape deformation follows the movements of the cursor, this similarity is very high: it is a major objective of the operators.

To summarize, our technique ranks extremely positively for the 3 properties quantifying *instrumental interactions*. Those rankings reflect the fact that our interaction technique has been designed as real *direct manipulation* interactions.

5.2. User Study

In order to gather feedback and to get an evaluation of our work, we conducter an informal user testing in which users were asked to draw two shapes (shown on Fig. 10) with our software and also with pencil and paper. The first shape (Fig. 10a) was designed to be difficult to achieve with our system so that users had to understand all the interactions to perform it, and then could give us some constructive feedback.

Figure 10: *a) first (difficult) model and b) second model generated during the user study.*

Ten participants (from 19 to 65 years old) served in the experiment. The completion time was measure, and the users were asked to score their satisfaction regarding the technique and regarding the graphical results.

Since the participants were not trained before the test, we expected them to be slower with our system than with pencil and paper. They were indeed slower most of the time, but not that much (the slowest participant took only twice the time she needed with the pencil, whereas another participant was even faster with our system).

Table 1 depicts the satisfaction results for our user study. We can see that on average our technique was ranked higher

User	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	Avg
Ta	60	100	25	75	-14	17	-17	-50	-33	0	16
Ra	60	250	-25	25	0	-33	67	133	-14	200	66
Tb	14	40	-13	60	50	0	14	40	0	0	21
Rb	167	100	20	13	13	33	60	0	0	30	44

Table 1: *technical (T) and result (R) satisfaction for models (a) and (b) with our system relative to satisfaction with pencil and paper (%).*

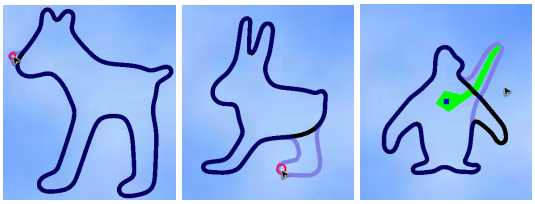

Figure 11: *Shapes created by our gesture-based design tool.*

than pencil and paper for the technical part as well as for the quality of the produced shape, and this result is consistant for the two models. Those results, while very preliminary, are encouraging, as this experiment shows that casual users can draw with our system without requiring prior training.

6. Conclusion & Future Work

All modeling metaphors use actions that can be achieved by a user in the physical world: sketching or sculpting. But there are users that cannot perform such actions, or who are not skilled in art. Moreover, computer devices give access to interactions that can do more than mimicking the physical world: sketches on paper cannot be deformed, but in a computer they can. The work presented here takes advantage of this property to propose an effective alternative to 2D sketching: progressive design through intuitive deformation gestures.

Our approach requires no a priori knowledge about the geometric model and no parameter setting to tune deformations. The shape perception based on M. Leyton's theory and analyses of natural user's interactions are used to interpret his/her intent and come out as a purely gesture-based contour modeling system. The effectiveness of this approach has been demonstrated through the constructive shape generation process we proposed. It allows users to progressively create and refine the shape they have in mind. It compares well to sketching and we are planning to compare it to other modeling software.

This framework has to be expanded toward complementary operators, such as the removal of material. About the set of available shapes: the current system cannot generate corner points yet. We are currently working on a new geometric model to allow such extensions. The framework also could be generalized to 3D shape design. However, this opens a number of new topics, such as 3D intuitive gesture-based interaction. Other devices, such as multi-touch screens, should be taken into account to enhance the range of interaction and initiate new ones for 3D shape modeling.

References

[Alv07] ALVARADO C.: Multi-domain sketch understanding. In *ACM SIGGRAPH 2007 courses* (2007), SIGGRAPH '07. 2

[BCD01] BOURGUIGNON D., CANI M.-P., DRETTAKIS G.: Drawing for illustration and annotation in 3D. *Comput. Graph. Forum 20*, 3 (2001). Special issue: EUROGRAPHICS 2001. 2

[BL00] BEAUDOUIN-LAFON M.: Instrumental interaction: an interaction model for designing post-wimp user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2000), ACM. 7

[BPCB08] BERNHARDT A., PIHUIT A., CANI M.-P., BARTHE L.: Matisse: Painting 2D regions for modeling free-form shapes. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling, SBIM 2008* (2008). 2, 4

[CCP*05] CHEUTET V., CATALANO C., PERNOT J.-P., GIANNINI F., FALCIDIENO B., LÉON J.-C.: 3d sketching with fully free form deformation features (δf^4) for aesthetic design. *Computers&Graphics 29* (2005). 3

[CIW08] CANI M.-P., IGARASHI T., WYVILL G.: *Interactive Shape Design*. Synthesis Lectures on Computer Graphics and Animation. Morgan & Claypool Publishers, 2008. 2

[GB08] GAIN J., BECHMANN D.: A survey of spatial deformation from a user-centered perspective. *ACM Trans. Graph. 27* (2008). 2

[IMH05] IGARASHI T., MOSCOVICH T., HUGHES J. F.: As-rigid-as-possible shape manipulation. *ACM Trans. Graph. 24*, 3 (2005). 2

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: A sketching interface for 3d freeform design. In *ACM SIGGRAPH'99* (1999). 2

[ISE] In *International Conference on Computer Graphics and Interactive Techniques, ACM SIGGRAPH 2006 Courses: Interactive shape editing*. 2

[JS09] JORGE J., SAMAVATI F.: *Sketch-based Interfaces and Modeling*. Springer-Verlag New York Inc, 2009. 2

[KHR02] KARPENKO O., HUGHES J. F., RASKAR R.: Free-form sketching with variational implicit surfaces. *Computer Graphics Forum* (2002). 2

[Ley88] LEYTON M.: A process-grammar for shape. *Artif. Intell. 34*, 2 (1988). 1, 2

[MS09] MCCRAE J., SINGH K.: Sketch-based interfaces and modeling (sbim): Sketching piecewise clothoid curves. *Comput. Graph. 33* (2009). 2

[Sch06] SCHWARTZ N. Z.: Attneave's cat revisited: Points of high curvature are not important for shape recognition. 3

[Shn83] SHNEIDERMAN B.: Direct manipulation. a step beyond programming languages. *IEEE Trans. on Computers 16*, 8 (1983). 7

[Ske] Google sketchup. [online] http://sketchup.google.com. 1

[Sor06] SORKINE O.: *Laplacian Mesh Processing*. PhD thesis, School of Computer Science, Tel Aviv University, 2006. 2

[SWSJ05] SCHMIDT R., WYVILL B., SOUSA M. C., JORGE J. A.: Shapeshop: Sketch-based solid modeling with blobtrees, 2005. 2