Interactive Inverse 3D Modeling

James Andrews[1], Hailin Jin[2] and Carlo Séquin[3]

[1]University of California, Berkeley, jima@eecs.berkeley.edu
[2]Adobe, hljin@adobe.com
[3]University of California, Berkeley, sequin@cs.berkeley.edu

ABSTRACT

"Interactive Inverse 3D Modeling" is a user-guided approach to shape construction and redesign that extracts well-structured, parameterized, procedural descriptions from unstructured, hierarchically flat input data, such as point clouds, boundary representation meshes, or even multiple pictorial views of a given inspirational prototype. This approach combines traditional "forward" 3D modeling tools with a system of user-guided extraction modules and optimization routines. With a few cursor strokes users can express their preferences of the type of modeling primitives to be used in a particular area of the given prototype to be approximated, and they can also select the degree of parameterization associated with each modeling routine. The results are then pliable, structured descriptions that are well suited to implement the particular design modifications intended by the user.

Keywords: reverse engineering, shape redesign, parameterized procedural modeling, interactive structure extraction, generalized sweeps.
DOI: 10.3722/cadaps.2012.xxx-yyy

1 INTRODUCTION

Many engineering tasks, re-design efforts, or artistic creations start from an existing prototype shape, which may exist only as a physical artifact, as a virtual shape in some computer file, or as a set of images or sketches. Unfortunately, because of insufficient record keeping or incompatibilities between different design systems, high-level design information that could readily be adapted to new requirements is often not available when a redesign or refinement becomes necessary. In other cases, previous high-level design information may not provide the best structure for making a desired change: for example, a vase modeled with a subdivision surface might be edited more easily by re-interpreting it as a surface of revolution with an editable profile curve. Likewise, when a model is reconstructed from point cloud data [17], photographs [9], or sketches [11], the representation used for the reconstruction may not be the most convenient for the desired re-design. For effective design optimization it is highly desirable to capture the prototype shape as a well-structured, parameterized, procedural geometry description, which can then be fine-tuned to meet the current specifications and design requirements.

Interactive Inverse 3D Modeling is a user-guided approach to extract such a pliable geometry description from unstructured, hierarchically flat input data; we primarily work with boundary mesh representations, but discuss extensions to point cloud data and multiple pictorial views in Sec. 4. Our framework consists of an integrated system of geometrical extraction and fitting routines, which are applied to user-designated portions of the given input shape, combined with a set of traditional "forward" 3D modeling tools that allow the extracted geometry to be edited and fine-tuned.

While *Interactive Inverse 3D Modeling* makes it possible to re-construct a proper geometrical model from some unstructured, incomplete, or ambiguous input data, its primary purpose is to create a well-structured, parameterized model that can easily be modified in some specific way. For instance, we may wish to make the handles of a vase (Fig.2) loopier, or change the number of handles, or change the vase profile. We may want to adjust the pose of an organic creature, treating the limbs as editable sweeps (Fig.5). We may want to arch the wings of a bird (Fig.14b) or selectively thin a swept sculpture in order to recreate an originally plastic or wooden sculpture (Fig.8) in bronze, at a large scale. While the system could readily be used for a wide range of shapes from creatures to crankshafts, in this paper we are especially interested in handling challenging, free-form shapes, which have been less extensively covered in the literature. We assume that the designer trying to extract a good geometrical description from a prototype shape already knows what the intended design modifications are, and thus can make sure that the appropriate degrees of freedom and edit-handles are incorporated in the extracted model description. We involve the user (re-designer) right from the beginning of the extraction process, and let them begin editing each primitive interactively as it is extracted.

Interactive Inverse 3D Modeling can be seen as a generalization of an intuitive approach that has been used by one of the authors since the mid-1990s in the development of several special-purpose computer programs to create models of abstract geometrical sculptures. One of the first examples started from a manually-generated wood sculpture by artist Brent Collins (Fig. 1a) The key geometric element, a sequence of biped saddles and holes, was captured in a parameterized program, called *Sculpture Generator I* [30]. This program allowed the user to change the number of saddles and holes in the chain, the degree of the saddles, as well as details of their geometry such as the thickness and extension of the flanges. For a particular setting of the various parameters, the program could closely reproduce the original inspirational shape shown in Figure 1a; but it could also be used to generate a multitude of new sculptures that all seem to belong to the same "family" (Fig. 1b, 1c). Note that these redesigned shapes are topologically different and could not be created using shape warping alone. Moreover, they are warped and twisted in different ways, so they cannot be created with symmetry-based part duplication alone.

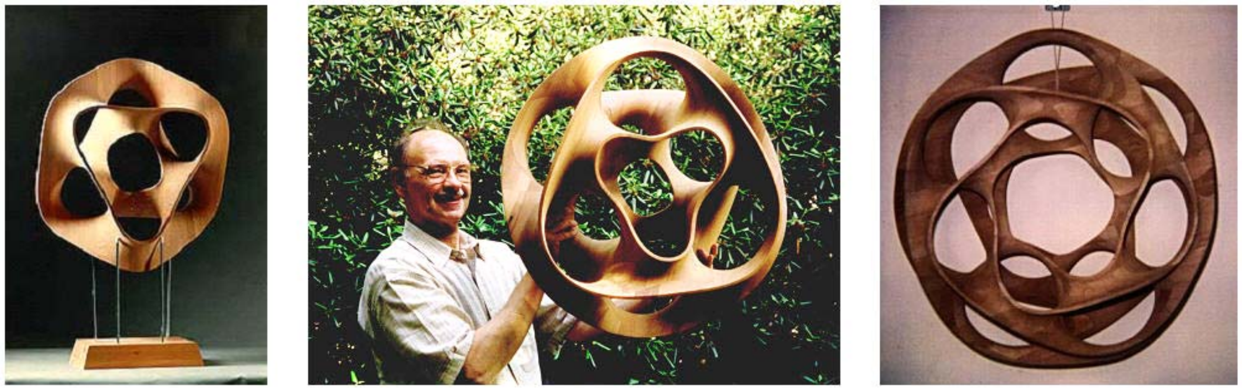

(a) (b) (c)

Figure 1: Inspirational examples of re-designs using the manually-programmed *Sculpture Generator I*. (a) Collins' original *Hyperbolic Hexagon*; (b) the generated *Hyperbolic Hexagon II*, an enhanced version with 3rd-order saddle surfaces; (c) *Heptoroid*, with an added 7th "story" and some twist, resulting in a single-sided Moebius geometry. Shapes (b) and (c) were designed on Séquin's *Sculpture Generator I* and realized in wood by Brent Collins (b) from blue prints produced by *Sculpture Generator I*.

The goal of our current effort is to generalize this approach to the reverse engineering of a much wider domain of shapes, ranging from engineering parts to consumer products and to the organic shapes of plants and creatures. We are in the process of constructing a unified suite of fitting methods, specifically designed to work together for the application of *Interactive Inverse 3D Modeling*. To make these tools truly useful for the intended purpose, we believe they should be:

- Easy to direct by the user with one or two menu clicks or cursor strokes;
- Fast enough to permit re-fitting with only seconds of interaction and computation time;
- Flexible enough to fit primitives with substantial non-matching surface details;
- Controllable with respect to the extent to which they cover a part of the input shape;
- Integrated with instant editing facilities that permit making high-level shape changes.

In addition to presenting the overall vision behind our integrated framework, this paper presents results (and some lessons learned) obtained with a few specific extraction modules realized during the last two years, in particular: two sweep modules that can produce generalized and highly parameterized rotational or translatory sweep geometries; some modules that extract CSG primitives and smooth surface patches; and a voxel-based description that starts from the visual hull, constructed from a few 2D depictions of the given prototype. The paper concludes with a plan of how the various modules will be integrated to jointly reconstruct a complicated piece of compound geometry.

2 RELATED WORK

The idea of fitting modeling primitives and other high-level structures to a given shape in order to facilitate editing and redesign has been pursued by many other researchers. Reverse engineering for CAD applications [2, 17, 36, 37] has traditionally focused on taking potentially noisy, unstructured data (for example from a 3D scan), and automatically segmenting the shape into a set of clean, non-overlapping primitive shapes that fully capture the desired structure and which can be imported into a CAD program. In these systems, user interaction is minimized (ideally, eliminated), and the goal is an error-minimizing fit. Work on primitive-based shape segmentation has covered similar conceptual ground [10, 38]. Some work has focused on reverse-engineering to a more easily edited primitive, for example a kind of sweep-morph primitive [29], but still with the general goal of finding a single best fit. We use many of the techniques developed for these systems (see Sec. 3), but instead of focusing on finding a single, ideally-fitting set of primitives, we focus on building a flexible system that may abstract some detail in favor of a more structured, high-level representation that can easily re-interpret a shape on the fly as the user's ideas and re-design goals may change. Where a traditional reverse-engineering goal is often to minimize user-interaction, ideally resulting in a fully automatic system, our goal is to have a fully-controllable, interactive system.

Some ground-breaking previous systems have introduced user-guided abstraction of a shape for redesign purposes. One such system is the method of Krishnamurthy and Levoy [16] in which a B-spline patch network is fit to an input shape in the form of a dense triangle mesh, using optimization routines to perfect the fit; the B-spline control points then allow artists to edit the reconstructed mesh more easily. Another stepping stone for our development are sweep-specific shape editing systems that let a user manually fit a set of sweeps to a model and then use the parameters of the sweep representation to edit the underlying shape [1, 39]. Several sketch-based modeling methods have also allowed the user to interactively specify primitives fitting a target shape, for modeling purposes: for example, by placing generalized cylinders, ellipsoids, and symmetry constraints over a sketched object [11]. Our focus is on integrating many such modules in a unified framework, where each module provides the few control parameters that are most useful for the redesign intensions of the user.

In addition to fitting modeling primitives, recent work has also demonstrated progress in automatically finding higher-level structure in a given input shape, for instance, finding the symmetries and alignments between parts [18, 24]. This permits inverse *procedural* modeling, in which a generative model describing an existing shape is discovered [3]. Interactively modifying the underlying symmetry of a given shape also played a key part in *Sculpture Generator I* [30], and we therefore include some symmetry-based editing tools in our framework.

Much recent work on variational surface editing has focused on factoring a model into a smooth surface with low-frequency detail and a fine, detailed surface, which can then be moved, transferred or edited separately [5, 32, 33]. This concept is broadly useful, and all our modules are designed to permit some fine details to be handled separately from the fitted high-level primitive structure.

Sketch-based methods for shape editing have often focused on direct editing of feature curves such as silhouette or contour lines [40], or more general feature lines [15]. Another sketch-based editing system focuses on editing meshes by editing the shading [12]. Focusing on such direct features, rather than primitives, is an alternative approach that can work in tandem with our system. In most cases, a general, mesh-based, smooth surface representation (see Sec. 3.4) is used as the underlying primitive representation to support such edits – this is a powerful, general primitive that we use as well.

3 BASIC FITTING MODULES

Our first step in building a useful *Interactive Inverse 3D Modeling* framework is to build a robust library of geometry extraction and fitting routines for diverse 3D modeling primitives. These primitives will serve as a basis for a useful high-level reconstruction of a given artifact. We focus on the parametric primitives most often found in today's CAD tools, because these familiar, practical primitives for shape design form a baseline of what can be expected from a 3D modeling tool. In many cases, suitable fitting methods for these primitives already exist, but they may need to be adapted to suit our interactive context. We require our fitting methods to be fast and user-controllable. We prefer that the user spend a few seconds indicating the primitive they would like to extract and the degrees of parameterization they desire, rather than relying on a fully automated process "to do the right thing." In addition, we like our extraction methods to be able to either overlook or preserve, if desired, intricate surface details that cannot be described conveniently with the higher-level primitive itself. The difference between the original data and the simplified extracted primitive is remembered as a decoration, which, if so desired, can be re-applied to a modified version of the extracted primitive. In the following we will discuss in some detail two types of sweep modules, as well as extractors for CSG-like primitives and for smooth surface patches – including offset surfaces and envelopes to our shapes.

3.1 "Stationary" or generalized rotational sweeps

The first extraction modules that we focused on were generalized sweeps. Sweeps are very powerful procedural geometry generators, which can readily be enhanced with various parameters that scale and orient the swept cross section or define the sweep path itself. It is practical to create two independent sweep modules for different classes of geometry. The first one is optimized to handle "stationary" sweeps; this includes simple rotational or helical sweeps based on a fixed rotational axis in space. The second one, discussed subsequently, can handle the more general cases of "progressive" or generalized translatory sweeps.

Stationary sweeps are relatively restricted, and because of that they can be extracted in an extremely efficient and unambiguous manner. We can find a stationary sweep fit for a mesh with a million triangles in about a quarter of a second on a typical laptop computer. The extraction method follows the ideas of [10, 28], which we summarize here:

The key idea is that any element of a surface generated by a stationary sweep must be tangent to the local sweep motion, and the surface normal must be perpendicular to the sweep motion. If we can express the local directions of the sweep motion as a velocity field, $\mathbf{v}(\mathbf{p})$, for example by differentiating the motion, then we can use the tangency criteria (known as "slippability" [10]) as a metric of how well a given point and normal $(\mathbf{p}, \mathbf{n})$ match the sweep motion: If the point is tangent to the motion, then $\mathbf{v}(\mathbf{p}) \cdot \mathbf{n} = 0$.

For the class of helices and surfaces of revolution, differentiating the local sweep motion gives the direction of that motion in the form of a velocity field over space: $\mathbf{v}(\mathbf{p}) = \mathbf{r} \times \mathbf{p} + \mathbf{c}$. The parameter $\mathbf{r}$ is the axis of revolution, and the parameter $\mathbf{c}$ can be broken into two parts: The component parallel to $\mathbf{r}$,

$\mathbf{c}_{\parallel} := \frac{\mathbf{c}\bullet\mathbf{r}}{||\mathbf{r}^2||}\mathbf{r}$, is the axis-aligned speed component of helical motion (zero for a surface of revolution), while the component perpendicular to $\mathbf{r}$, $\mathbf{c}_{\perp} := \mathbf{c} - \mathbf{c}_{\parallel}$, is related to the position of the axis in space; specifically, it is the Plücker momentum vector of the axis. One point on the axis is $\mathbf{r} \times \mathbf{c}_{\perp}$.

With this linear velocity field, we can express the *slippability* metric as:

$$\mathbf{v}(\mathbf{p})\bullet\mathbf{n} = (\mathbf{r} \times \mathbf{p} + \mathbf{c})\bullet\mathbf{n} = \mathbf{r}\bullet(\mathbf{p} \times \mathbf{n}) + \mathbf{c}\bullet\mathbf{n} \equiv \mathbf{m}\bullet\mathbf{l}(\mathbf{p},\mathbf{n}), \tag{3.1}$$

where $\mathbf{m}$ is a 6-dimensional vector formed by concatenating $\mathbf{r}$ and $\mathbf{c}$, and likewise $\mathbf{l}(\mathbf{p},\mathbf{n})$ is a concatenation of $\mathbf{p} \times \mathbf{n}$ and $\mathbf{n}$. To avoid a trivial, zero-vector solution, this metric must be normalized: for helices and surfaces of revolution, Pottmann and Randrup suggest normalizing by the squared magnitude of the rotation axis $\mathbf{r}$ [28]. Fitting the field to a given set of (user-marked) surface points is then a matter of minimizing:

$$\begin{aligned}
&\min_{\mathbf{m}} \sum_i \frac{(\mathbf{v}(\mathbf{p}_i)\bullet\mathbf{n}_i)^2}{||\mathbf{r}||^2} \\
&= \min_{\mathbf{m}} \sum_i \frac{(\mathbf{m}^{\mathbf{T}}\mathbf{l}(\mathbf{p}_i,\mathbf{n}_i))^2}{\mathbf{r}^{\mathbf{T}}\mathbf{r}} \\
&= \min_{\mathbf{m}} \frac{\mathbf{m}^{\mathrm{T}}(\sum_i \mathbf{l}(\mathbf{p}_i,\mathbf{n}_i)\mathbf{l}(\mathbf{p}_i,\mathbf{n}_i)^{\mathrm{T}})\mathbf{m}}{\mathbf{r}^{\mathbf{T}}\mathbf{r}} \\
&\equiv \min_{\mathbf{m}} \frac{\mathbf{m}^{\mathbf{T}}\mathbf{M}\mathbf{m}}{\mathbf{m}^{\mathbf{T}}\mathbf{N}\mathbf{m}}
\end{aligned} \tag{3.2}$$

Here $\mathbf{M}$ is defined by an outer product of $\mathbf{l}(\mathbf{p}_i,\mathbf{n}_i)$ with itself, and $\mathbf{N}$ is a diagonal matrix in which the first three diagonal elements are 1 and the rest are 0. This minimization can then be solved by the eigenvector associated with the minimum eigenvalue of the generalized eigenvalue problem $\mathbf{M}\mathbf{v} = \lambda\mathbf{N}\mathbf{v}$ [28]. Once a best fitting sweep motion is found, we flood-fill the given surface from the points marked by the user, gathering all surfaces elements that are at least as "slippable" as the user-marked points, with respect to the best fitting motion. Optionally, we can then re-fit the global sweep parameters using all the points currently contained in the segmentation, and re-grow the flooded surface domain. This process can be iterated until convergence. While this is not strictly necessary for this fitting technique to work, we do so in our system because it yields better results for sparse user input.

Note that this process simply considers tangency to a globally-defined sweep motion, and does not define a specific swept profile curve. Thus this method tends to accept some points that match the motion, but are not conceptually part of a coherent sweep surface. We may perform some light filtering using morphological operators to remove narrow disconnected surface strips, as suggested by [14]. However, some user correction may still be needed in cases such as the one shown in Figure 2, because the erroneously accepted regions may be fairly large, and there is no well-defined metric reason to reject them. Here is again a situation, where a small amount of user input can easily circumvent the problem. In one approach, the user first extracts the four handles and the top-loop with a progressive sweep (see below); this then automatically eliminates these areas from consideration when the stationary sweep is started on the main body. Alternatively, the problem can be easily corrected after a first sweep has been initialized on the main body. Our system allows all the extracted surface parts to be collapsed into a 2D *profile view* by undoing the sweep motion for all selected surface elements. In this *profile view* (Fig.2c) the undesirable portions (shown in orange) can readily be selected and eliminated from the extracted sweep surface. The same 2D *profile view* can also be used to generate an idealized sweep profile curve by doing a skeletonization of the green areas of the profile view.

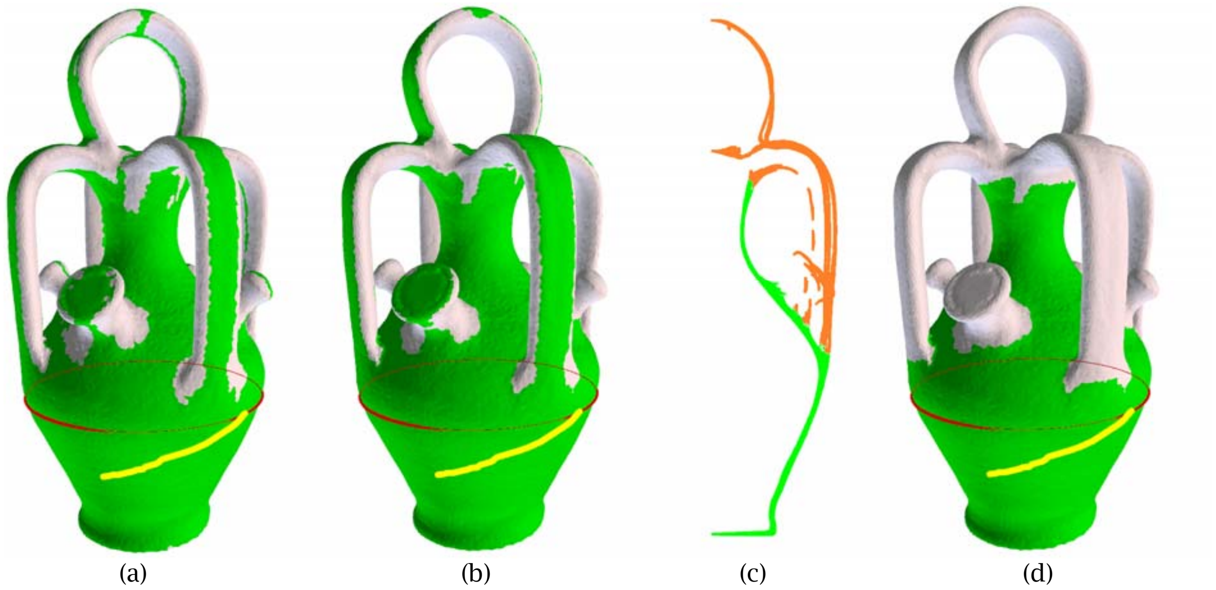


Figure 2: (a) From an initial stroke (yellow) drawn by the user, the circular sweep path (red) is extracted, and matching surface elements are marked in green. (b) The result is filtered by morphological opening and closing, removing some spurious selections. (c) Undoing the rotational sweep component generates a 2D *profile view*. (d) By de-selecting portions of this profile (orange) the user can manually cull undesired areas from the body sweep.

Simple editing operations in the 2D profile view can also be used to edit select portions of the extracted sweep surface. In Figure 3 a not-very-circular statue has been fit with a rotational sweep (a). The corresponding 2D *profile view* (b) has substantial thickness due to the variation of the radial distance to the central rotation axis. A small portion of this profile has been highlighted and selected for editing (b, c). If the highlighted profile portion is moved to the right, the parts of the sculpture containing the four noses and eight cheeks are being bulged outwards (d). Alternatively, if just the portions of the profile corresponding to the nose-tips are selected (e), then only the four noses are elongated; the cheeks of the face do not bulge outwards (f).

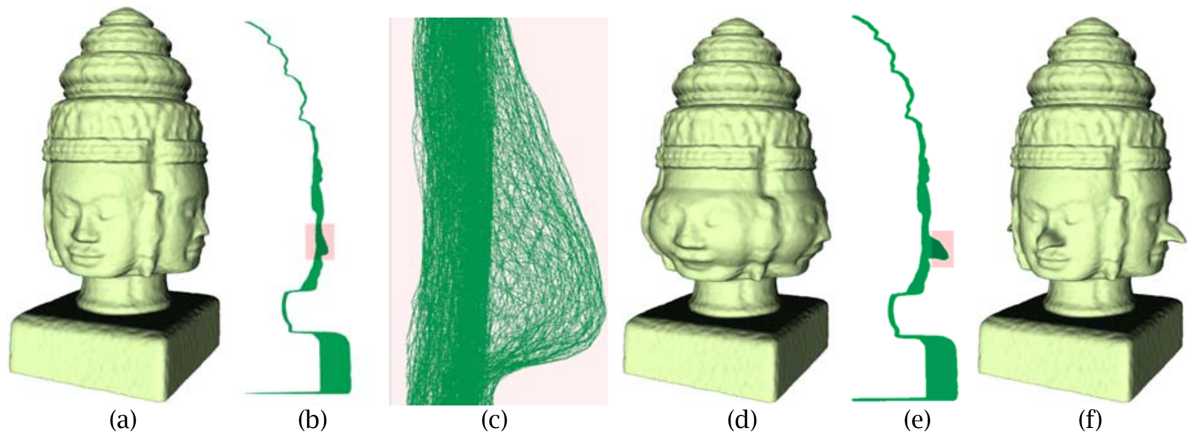


Figure 3: (a) A not-very-circular statue is fit with a rotational sweep. (b) The corresponding "thick" 2D *profile view*. (c) A small portion of the profile selected for editing. (d) Modified statue after the selected profile portion has been moved to the right. (e) Just the nose has been selected in cross section view. (f) A pointy nose has been created by moving the selected portion to the right.

3.2 "Progressive" or generalized translatory sweeps

For the more general cases of sweep fitting, we follow the method of [1] and fit a "progressive" sweep geometry, which is specified by an arbitrary 2D cross section swept along an arbitrary 3D sweep path. Because this class has many more degrees of freedom and is thus inherently more ambiguous, we also start with more specific hints by the user. As a minimum we require: a stroke indicating a good start location and an initial direction for the sweep (yellow strokes in Fig.4); – but possibly sketching out the whole extent of the sweep path. This yields significant control over how the shape will be interpreted. For example, a short vertical stroke on the lid of the teapot produces a vertical sweep (Fig.4a), but a stroke on the spout produces a mostly horizontal sweep (Fig.4b). In both cases freeform scaling of the cross section by an arbitrary 2x2 matrix was permitted. Depending on how the user plans to use and/or modify the teapot model, one or the other representation may be preferable.

In the situation shown in Figure 4b, the sweep fit, which starts in the middle of the spout, would naturally stop at the tip of the spout and where the spout meets the teapot body. However, the user can force the sweep fit to continue throughout the body of the teapot by adding or extending sweep path strokes. Internally, the error margins that specify what surface elements are acceptable to be included in the sweep fit are increased appropriately, so that the sweep can continue. This may then allow the sweep to gather surface portions beyond what the user had in mind. To deal with that situation, the user is also provided with a command to stop a sweep at a particular location.

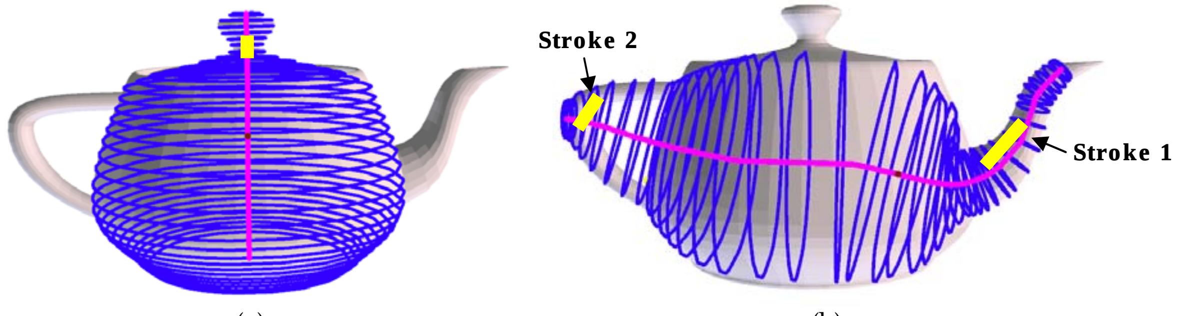


Figure 4: Different initial strokes result in different sweep fits. (a) A vertical stroke on the lid produces a vertical sweep. (b) Stroke 1 on the spout produces a mostly horizontal sweep. Additional strokes, e.g. stroke 2 near the handle, may be used to extend the range of the sweep fit.

Specifically, our progressive sweeps are parameterized by a 2D polyline cross-section that is swept along a 3D polyline path and transformed relative to the rotation-minimizing frame at each 3D path point by a matrix transformation: Depending on user preference, the allowed transformations are: pure rotation, pure scaling, rotation and scaling, or arbitrary stretching and skewing (parameterized by an unconstrained 2x2 matrix).

The fitting process used is a fairly straightforward error minimization [1], which we outline here. First, the algorithm takes the point at the center of the initial user stroke, intersects the ray to that point with the mesh, and slices the mesh perpendicular to the stroke. A small initial sweep model is generated from this initial cross section. Then we iteratively alternate between the following two steps: (1) An off-the-shelf Levenberg-Marquardt optimization [21] is used to minimize the distances of vertices on the sweep model from the given input surface, with some additional error terms for regularization. (2) The sweep model is extended by adding a new segment to the 3D sweep path. In the first step of this optimization, as the sweep direction is refined, the cross section estimate is also updated to match. The regularization term is necessary to get an efficient and smooth result: high curvature, any changes in the parameters, and short segments in the sweep path are all penalized. To restrain the cost of the optimization, only the last few segments of the sweep are optimized at any given time.

The specific error term used is:

$$\Sigma_i d(\mathbf{p}_i)^2 + w\Big(\Sigma_{j=N-k}^{N}\big(\frac{1}{||\,\mathbf{s}_j - \mathbf{s}_{j-1}\,||}\big)^2 + \Sigma_{j=N-k}^{N}\,||\,\mathbf{x}_j - \mathbf{x}_{j-1}\,||^2 + \Sigma_{j=N-k}^{N-1}\kappa(\mathbf{s}_{j-1},\mathbf{s}_j,\mathbf{s}_{j+1})^2\Big) \tag{3.3}$$

where $d(\mathbf{p}_i)$ is the distance from the sample points on the sweep model surface $\mathbf{p}_i$ to the input surface; $\mathbf{s}_i$ are 3D positions of the sweep path; $\mathbf{x}_i$ are the parameters of the sweep transformation; and κ is the curvature metric on the sweep path.

Note that a parameter w controls the weight of the entire regularization term. This parameter intuitively specifies how much the system is allowed to prefer smooth, simple sweeps at the cost of conforming less accurately to the true surface. We provide a good default value for w, but also believe the meaning of this variable is clear enough to let the user take control of it.

Due to the large ambiguity of the problem, this fitting module will sometimes stop sooner or later than desired, so we provide a basic interface to delete portions of the sweep or to extend the fit. In some cases, the progressive sweep module may even find an error-minimizing fit that is not the sweep envisioned by the user. In these cases the user deletes the erroneous portion of the path and draws (in 2D) a new target path. We then re-run the fit with a new regularization term forcing it to follow the corrected path. Examples of different sweeps extracted on the well-known Armadillo model are shown in Figure 5. Note the rich variety of extracted sweeps based on just short strokes (shown in yellow) drawn by the user.

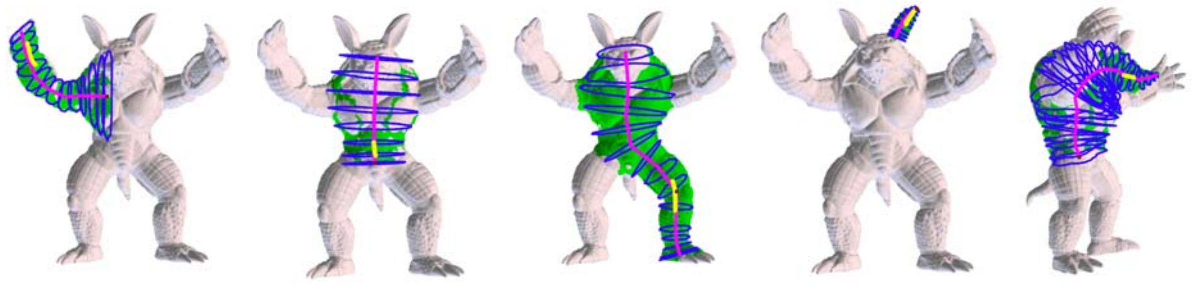


Figure 5: The user can quickly extract any number of diverse sweeps on an armadillo.

3.3 CSG Primitives

Automatic fitting of CSG primitives (quadrics, cones, half-planes and cuboids) is well studied, and the common methods used are very similar to those used for stationary sweep fitting: The fitting problem can be linearized as an eigenvalue problem [34], and RANSAC [7] or Lloyd iteration [20] can be used to robustly fit primitives [38]. Automatic structure analysis has been used to generate extremely high quality decompositions from even noisy point cloud data [17].

We have implemented a quadric fitting algorithm in the same framework as the stationary sweep fitting algorithm discussed above, because quadric surface patches cover a good set of the useful CSG primitives and can be fit very efficiently. As with stationary sweep fitting, the user marks example points on the given model to initialize a fit by some CSG primitive, and the system then automatically finds the primitive that best fits all those points (Fig.6). Alternatively the user may explicitly restrict the fitting process to a particular type of primitive, e.g., a cylinder.

We follow the method of [34] for our fitting process, based on the explanation of [38]. Specifically, we consider a quadric parameterized as:

$$f(\mathbf{p}) = C_0 + C_1p_x + C_2p_y + C_3p_z + C_4p_xp_y + C_5p_xp_z + C_6p_yp_z + C_7p_x^2 + C_8p_y^2 + C_9p_z^2 \equiv \mathbf{m}\cdot\mathbf{l}(\mathbf{p}) \tag{3.4}$$

where $\mathbf{m}$ is the vector of all parameters C_i and $\mathbf{l}(\mathbf{p})$ is the function from $\mathbf{p}$ to a corresponding coefficient vector, $<1, p_x, p_y, p_z, p_x p_y, p_x p_z, p_y p_z, p_x^2, p_y^2, p_z^2>$. We minimize Taubin's first order metric of distance from sample points on the input surface to the quadric surface, $d^2(p) = \dfrac{f^2(\mathbf{p})}{\|\nabla f(\mathbf{p})\|^2}$, which is further approximated by minimizing for all points:

$$\begin{aligned} &\min_{\mathbf{m}} \frac{\sum f^2(\mathbf{p}_i)}{\sum \|\nabla f(\mathbf{p}_i)\|^2} \\ &= \min_{\mathbf{m}} \frac{\mathbf{m}^{\mathbf{T}}\left(\sum \mathbf{l}(\mathbf{p_i})\mathbf{l}(\mathbf{p_i})^{\mathbf{T}}\right)\mathbf{m}}{\mathbf{m}^{\mathbf{T}}\left(\sum \mathbf{l_x}(\mathbf{p_i})\mathbf{l_x}(\mathbf{p_i})^{\mathbf{T}} + \mathbf{l_y}(\mathbf{p_i})\mathbf{l_y}(\mathbf{p_i})^{\mathbf{T}} + \mathbf{l_z}(\mathbf{p_i})\mathbf{l_z}(\mathbf{p_i})^{\mathbf{T}}\right)\mathbf{m}} \\ &\equiv \min_{\mathbf{m}} \frac{\mathbf{m^T M m}}{\mathbf{m^T N m}} \end{aligned} \tag{3.5}$$

where $\mathbf{l_x}$ is the partial derivative of $\mathbf{l}$ with respect to x, and $\mathbf{M}$, $\mathbf{N}$ are matrices defined by the outer products of $\mathbf{l}(\mathbf{p_i})$ s. This minimization can then be solved by the eigenvector associated with the minimum eigenvalue of the generalized eigenvalue problem $\mathbf{Mv} = \lambda \mathbf{Nv}$ [38].

As with the stationary sweep fitting, we use a flood-fill to select points that are close to the estimated quadric surface, and we can optionally iterated this process by re-fitting the quadric based on the new points added in the flood-fill. For our flood-fill process, we use two metrics to test whether a given input point and normal is a match with the current quadric surface: the distance to the quadric surface [35] and how parallel surface normal is to the gradient of the quadric: $|\nabla f(\mathbf{p}) \cdot \mathbf{n}| - 1$. We only grow the fitted surface to a new point if it is close to the quadric by both metrics.

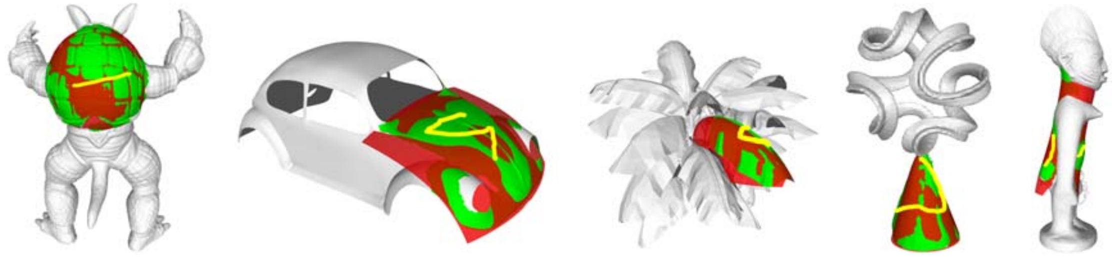


Figure 6: Strokes (in yellow) are used to initialize various quadric fits on a variety of shapes. The quadric is shown in red, and the given input triangles are shown in green.

3.4 Smooth Surface Patches

Many types of smooth surface primitive exist: for example, NURBS patches, subdivision surfaces, or even mesh-based variational patches [4] can be used. The former two types tend to be faster to evaluate and more precise, but more dependent on a carefully controlled patch network topology for good results, which renders them intimidating for novices compared to variational patches, which, however, are computationally more expensive. Smooth surface patch fitting can be used in several different fitting contexts: It can match an existing surface topologically, it can abstract a surface by matching a smooth surface envelope to the surface [22], or it can also be used for modeling offset surfaces and for blending between surface portions extracted by other modules.

In our initial system, we focus on editing triangle meshes, so the most convenient method is the mesh-based variational patch, specifically a linearized thin-plate spline [4]. In this method, we simply take the original mesh, fix some control points, and generate a smooth surface minimizing the surface's Laplacian at each point. We do this by solving:

$$\sum_{i=1}^{n} (\Delta \mathbf{p}_i)^2 + \sum_{c=1}^{m} (\mathbf{t}_c - \mathbf{p}_{f(c)})^2 \tag{3.6}$$

where $\{\mathbf{p}_i\}$ are the n vertices of the original mesh, $\{\mathbf{t}_c\}$ are the m control points, and $f(c)$ is a mapping from control points to vertices. $\Delta\mathbf{p}$ is the mesh-based Laplacian operator [27]. The resulting surface will be a smooth abstraction of the original surface [31], interpolating the selected control points and smooth everywhere, as long as the mesh is tessellated finely enough to allow smoothness. In a first application this capability can be used to suppress some details on the input shape. In the example of Figure 7, the portion of the mesh that covers the elephant's ear (yellow) is selected and modeled as a thin-plate spline. Optimization of this surface area will smooth it out and remove all the details (Fig.7b).

This method also can be changed only slightly to preserve detail in full. Instead of minimizing the mesh Laplacian, we minimize the difference between the current mesh Laplacian and the Laplacian of the original mesh vertices $\{\mathbf{o}_i\}$:

$$\sum_{i=1}^{n}(\Delta\mathbf{p}_i - \Delta\mathbf{o}_i)^2 + \sum_{c=1}^{m}(\mathbf{t}_c - \mathbf{p}_{f(c)})^2 \tag{3.7}$$

This detail preserving version is called Laplacian surface editing [32]. In both cases, the system can be solved efficiently as a sparse linear system, using a sparse direct Cholesky method [6]. Figure 7c shows how this capability can be used to preserve continuity and smoothness in the right fore-leg of the elephant when the foot (marked by the green patch) is moved, while the yellow thin-plate spline serves as an anchor.

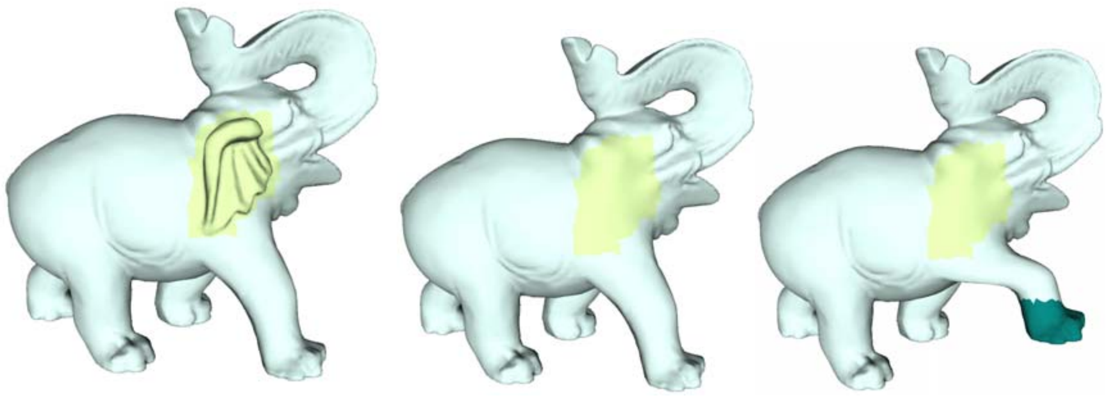

(a) (b) (c)

Figure 7: (a) The elephant's ear region is selected; (b) when optimized as a thin-plate spline, all surface details get smoothed away. (c) The foot marked by the green patch is moved, and continuity in the leg above it is preserved by standard Laplacian surface editing.

4 VARIOUS TYPES OF INPUT DATA

The description of the original artifact can come in many different forms. Our first implementation focuses on triangle meshes, but the concepts apply much more generally, and we have developed some support for additional forms of input data.

4.1 Boundary Representations

We started our exploration with boundary representations in the form of finely tessellated meshes. There is a wide variety of models in that format available on the web. Also, such representations often exist in-house of a design studio or engineering company; they may be the final STL files that were sent to a 3D printer or to another rapid prototyping machine. Often these files can still be found in some data repository, even when the original design files that contained a more structured description of the part have been lost.

4.2 Point Clouds

If only a physical prototype is still available, but no actual data files can be found, it may be most expedient to use a 3D-scanner to quickly capture the geometry of that prototype. It may or may not be necessary to create a completely legal B-rep, before an attempt is made to fit some of the parameterized primitives to that data. Traditional reverse engineering approaches can be used to create a legal B-rep when desired [2, 17]. In most cases, the parameters of the primitives can just as easily be optimized by trying to minimize the sum of the squared distances of the disconnected points in the cloud from the generated geometry. Note that all of the primitive fitting methods we present, except the spline surface method, are based on work that originally was aimed at fitting point clouds.

4.3 Visual Hulls and Other Constraining Features

A further step in the above direction is to have the extracted geometry be guided and constrained by an even sparser set of primitives – possibly at the cost of somewhat more involvement by the user. A number of systems currently allow users to model from photographs; these systems can be used to generate geometry suitable for our current system [9, 26]. We also plan to explore integrating inverse 3D modeling more directly in the process: our recent investigations have used the visual hulls generated from a few 2D views of the prototype, as well as a few pronounced geometrical features, annotated by the viewer to define targets for the procedurally generated geometry. By combining basic user hints and a straightforward non-linear optimization, we were able to reconstruct the sculpture shown in Figure 8 from just nine images.

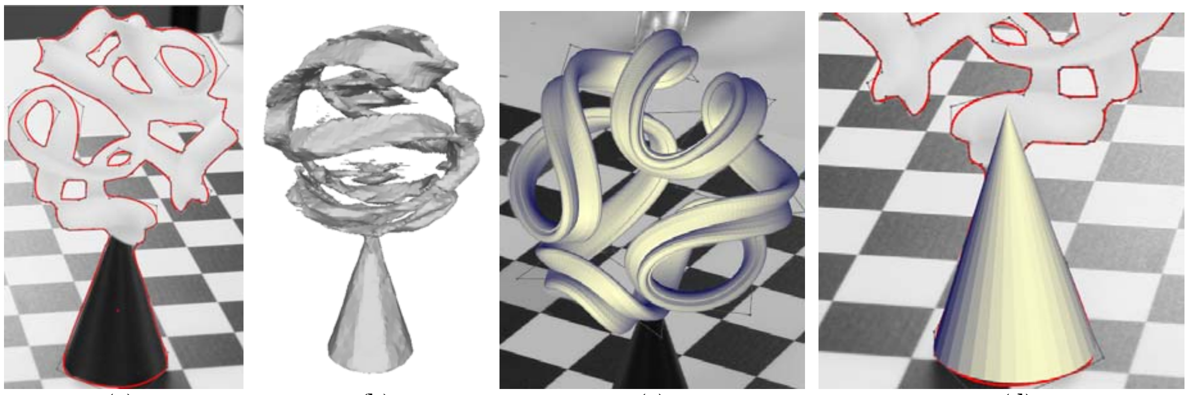

(a) (b) (c) (d)

Figure 8: The silhouette edges in one image (a) are used for constructing a complete visual hull (b). A sweep is fit to fall within the visual hull (c). A cone is used to fill the visual hull of the base (d).

4.4 Skeletons, Symmetry Axes, and Mirror Planes

Even more abstract geometrical elements, that are not by themselves part of the object boundary, can be used to steer the extraction system towards a representation with the semantically most useful structure for the intended re-design. Stems of plants or legs of an animal could readily be outlined by an approximate skeletal line drawn by the user into a few 2D views of the artifact, and these skeletal lines could then serve as a path for a progressive sweep.

Approximate symmetries, readily apparent to the designer, can be specified as constraints and can then be used to fill in domains in the prototype description that may be missing in the given data set, or to average and smooth the available input data in order to generate a truly symmetrical description. The high-level description of such a shape relies on multiple instances of one extracted and averaged segment. Thus we can readily change the symmetry of such an object by changing the number of instances after the prototype segment has been properly modified, e.g., rotationally compressed in a cylindrical coordinate system (Fig.9).

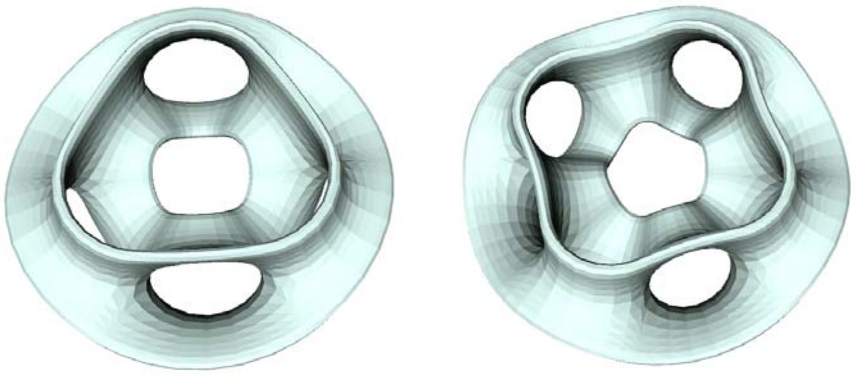

(a) (b)

Figure 9: Given a symmetric object, our system can edit the symmetry, for example by changing the 3-fold rotational symmetry in (a) to a 4-fold rotational symmetry in (b). We do this by scaling a 120˚ slice of the original model to 90˚ and instancing it four times.

5 TRANSFORMATION BETWEEN FITTING PRIMITIVES

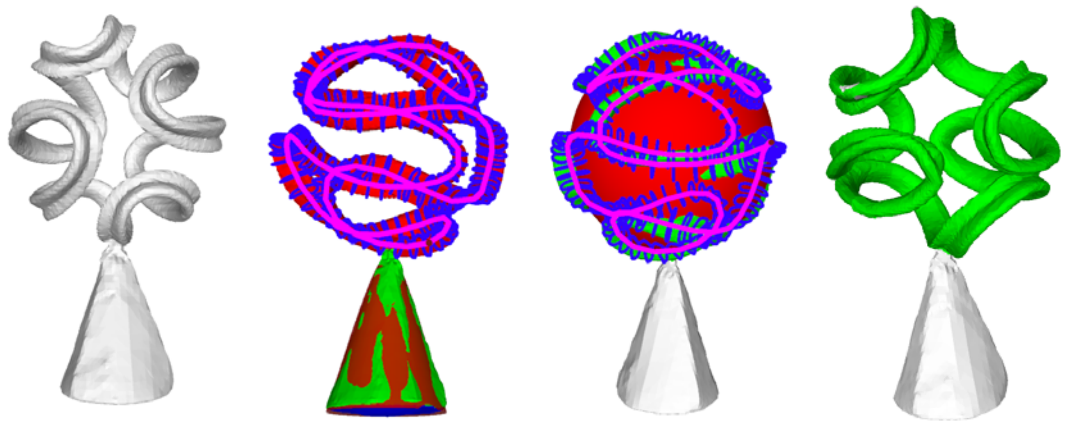

(a) (b) (c) (d)

Figure 10: The sculpture model (a) is fit with multiple primitives (b): a progressive sweep on top, a cone in the lower half, and a plane on the bottom. The sweep path itself can be fit onto a sphere (c). This spherical structure can be retained along with any specified symmetry (here D_2) while editing the sweep path.

As the user specifies various modeling primitives to approximate different portions of a given shape, the covered surface areas will tend to grow into one another. There are several ways to handle this overlap. A first approach uses segmentation to find a natural boundary between the areas modeled by different primitives and then restricts the growth of the covered surface area to one side of that boundary. Alternatively, a hierarchical distinction can be used. Several primitives can apply to the same geometry, but at different levels of abstraction. For example, the overall structure could be an ellipsoid, but small details could be modeled as thin plate splines. Thirdly, extracted higher-level structures themselves can be fit to some geometrical model. For instance, a progressive 3D sweep path could be fit against a quadric surface. This might reveal that the sweep path of the sculpture in Figure 10 lies on a sphere, and this property could then be locked in during a subsequent shape editing session.

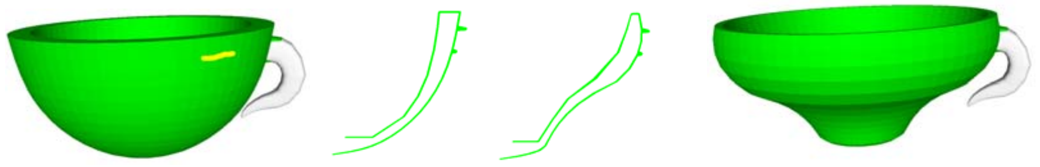


Figure 11: (a) teacup is approximated as a surface of revolution; (b) the corresponding profile; (c) edited profile, and (d) the resulting shape of the cup. The handle, modeled as a Laplacian surface extension moves with the changes in the bowl.

Our system initially captures an incoming surface as a thin-plate surface and preserves the Laplacians. This permits Laplacian surface editing and preservation of details. As the user marks regions and selects specific modeling tools for them, these surface areas are handed over to their respective higher-level primitives. As individual primitives are edited, deformed or moved, the uncovered areas between them are still modeled by the original Laplacian patches and tend to interpolate between the surface pieces belonging to the various models. This leads to a shallow hierarchical structure, in which the fit primitives may be organized in one or more hierarchical layers below the initial Laplacian surface, with the option to have different primitives in different hierarchical layers for the same surface area. Shape edits will then take place in these layers in sequence from the bottom to the top of the hierarchy. As an example, a teacup is approximated as a surface of revolution (Fig.11a) and a corresponding profile is extracted (Fig.11b). This profile can then be edited (Fig.11c), thus changing the shape of the cup (Fig.11d). Since the handle did not get covered with the surface of revolution (green), it is still modeled as Laplacian surface extension with boundary conditions derived from the shared contact points with the green surface; it thus automatically moves along with the changing bowl shape.

We allow several specific interactions between different fitting modules:

(1) We allow quadric fitting on the 3D points of the progressive sweep paths, which is straightforward because this fitting algorithm applies to arbitrary point clouds.

(2) We can also edit 2D and 3D curves with Laplacian curve editing [32], a natural extension of the surface optimization to 1-manifolds.

(3) We allow for conversion from specialized to more general fitting primitives. Currently we allow quadrics to be converted to surfaces of revolution (with an additional scaling factor), and surfaces of revolution to be converted to progressive sweeps.

These conversions allow the user to easily add degrees of freedom to an existing fit. For instance, to convert a surface of revolution to a progressive sweep, we place a straight-line sweep path along the axis of revolution and use a circular cross section that scales to match the profile curve. In cases where the profile curve folds back on itself, and multiple cross section scales would be needed simultaneously, we can break the sweep into separate segments that join at the turn-around points.

To convert an arbitrary quadric to a surface of revolution, we first rotate the quadric into a canonical space that is convenient for analysis. To do so, we follow the method of [8] and express the quadratic terms in matrix form:

$$\mathbf{p}^{\mathbf{T}}\mathbf{A}\mathbf{p} + \mathbf{b}^{\mathbf{T}}\mathbf{p} + C_0$$
$$\text{with } \mathbf{A} = \begin{pmatrix} C_7 & C_4/2 & C_5/2 \\ C_4/2 & C_8 & C_6/2 \\ C_5/2 & C_6/2 & C_9 \end{pmatrix},\ \mathbf{b} = \begin{pmatrix} C_1 \\ C_2 \\ C_3 \end{pmatrix} \tag{5.1}$$

We can rotate the quadric to eliminate the cross-terms by using an eigendecomposition, $\mathbf{A} = \mathbf{R}\mathbf{D}\mathbf{R}^{\mathbf{T}}$; then in the rotated space $\mathbf{p}_{\mathbf{r}} = \mathbf{R}^{\mathbf{T}}\mathbf{p}$ the new quadric becomes $\mathbf{p}_{\mathbf{r}}^{\mathbf{T}}\mathbf{D}\mathbf{p}_{\mathbf{r}} + (\mathbf{R}\mathbf{b})^{\mathbf{T}}\mathbf{p}_{\mathbf{r}} + C_0$. The columns of the rotation matrix $\mathbf{R}$ are the axes of the quadric. A point $\mathbf{p}_a$ on the axes can then be found as $\mathbf{p}_a = \frac{1}{2}\mathbf{R}\mathbf{D}^{\text{-}\mathbf{P}}\mathbf{R}^{\mathbf{T}}\mathbf{b}$ where $\mathbf{D}^{\text{-}\mathbf{P}}$ is a Moore-Penrose pseudo-inverse [13] in case $\mathbf{D}$ is singular. The terms of the diagonal matrix $\mathbf{D}$ also tells us how space is scaled along each axis. If all the terms have the same sign, we pick the column of $\mathbf{R}$ corresponding to the smallest term as the axis of revolution $\mathbf{r}_1$; otherwise, we choose the axis corresponding to the terms with the opposite sign. The remaining two axes $\mathbf{r}_2$ and $\mathbf{r}_3$ then describe the major and minor axes of an ellipse scaled by the corresponding scale terms D_{22} and D_{33}. By scaling space with scale matrix $\mathbf{S} = (1 - \frac{\mathbf{D}_{22}}{\mathbf{D}_{33}})(\mathbf{I} \text{ - } \mathbf{r}_3 \otimes \mathbf{r}_3)$, this ellipse becomes a circle, and the scaled space can then be edited as a surface of revolution. (In the degenerate case where one or both of D_{22} and D_{33} are zero, the quadric surface is planar and can be handled separately as a plane.) The stationary sweep 'velocity field' which fits the surface is finally:

$$\mathbf{v}(\mathbf{p}) = \mathbf{S}^{-1}(\mathbf{r}_1 \times \mathbf{S}\mathbf{p} - \mathbf{r}_1 \times \mathbf{p}_a) \tag{5.2}$$

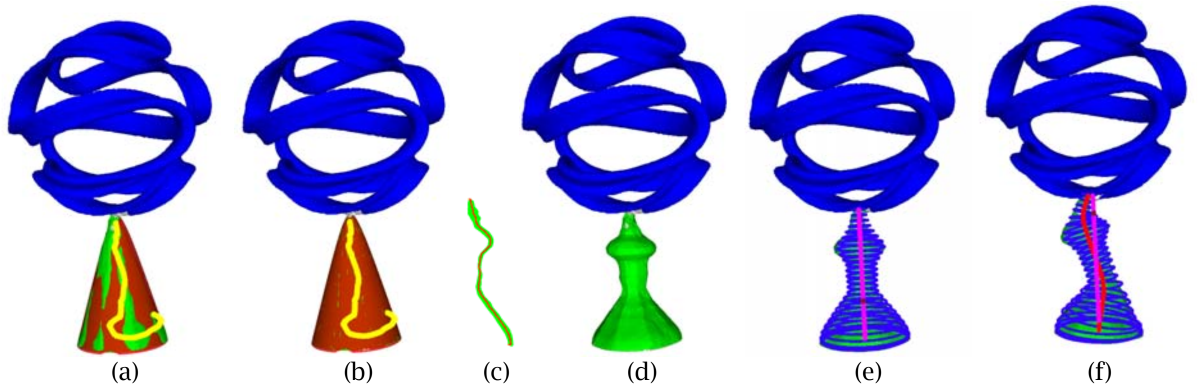


Figure 12: An example of progressive editing: The base (green) of the sculpture from Figure 8b (reconstructed noisily from a visual hull) is first approximated by a quadric surface shown in red (a). Projecting the base mesh to the quadric surface gives a cleaner result (b). The mesh is then converted to a surface of revolution and modified by editing the profile curve (c, d). Finally, the base is converted to a progressive sweep (e), and its sweep path (magenta) is edited to further deform the pedestal (f).

6 SHAPE EDITING AND PRESERVATION OF MODEL DETAILS

The most direct application of this inverse modeling tool is that the fitted primitives can be used directly for shape editing. This is in itself a powerful tool for redesign, as a user can quickly re-interpret any shape with the primitive most suited for an intended modification and then make an appropriate edit. Figures 2-3, 7-8 and 10-12 demonstrate several examples of this editing in practice.

Each primitive allows different types of edits. For surfaces of revolution, surface details captured by the sweep can be swiveled into a 2D plane by moving along the velocity field defining the sweep. These details can then be edited directly in this 2D view, and the changes can be propagated back to 3D (Fig.3). For progressive sweeps, the surfaces can be edited by modifying the cross section curve in a 2D view, or by editing the 3D sweep path (Fig.12f). Finally, for smooth surface patches the user can select some vertices as control points and specify new target positions, letting the system solve for the positions of the unselected vertices.

It is noteworthy that any quadric can be converted to a canonical form by scaling and rotation, a fact we used in Section 5. This implies that the natural parameters to edit a quadric are simply linear transformations applied to the whole quadric shape: rotation, scaling, and skew transformations, – which can all be presented to the user in a compact, easy-to-modify form.

For each primitive type, details can be preserved or removed by scaling the vectors or distances describing the details (the target Laplacian vectors for the smooth surface, or the displacement distances or vectors for the other primitives). Where the primitive parameters themselves have been set by some higher-level fitting procedure, for example by fitting a sphere to a sweep path and re-projecting the path onto it, we can then enforce this new-found higher-level structure. For example, in Fig.10d we can allow the user to freely edit the sweep path in the displayed 2D projection, but the system will always project the path back onto the underlying sphere. Alternatively, we could change the sphere into an ellipsoid and thus easily change the path in a more global manner.

As a shape is redesigned, the user may wish to selectively discard or keep details that were abstracted away by the high-level model used for shape editing. Our smooth surface primitive can preserve detail by preserving the Laplacian vectors of the original surface, but the other primitives need additional machinery to preserve detail. There are two main approaches: details can be preserved as a displacement map on the primitive surface, or can be kept as raw geometry with some mapping to the surface of the primitive. The displacement-based method forces the topology of the surface to closely match the fit primitive, while the raw geometry method preserves arbitrarily complex geometry from the input. Either case can be useful, depending on the intended redesign.

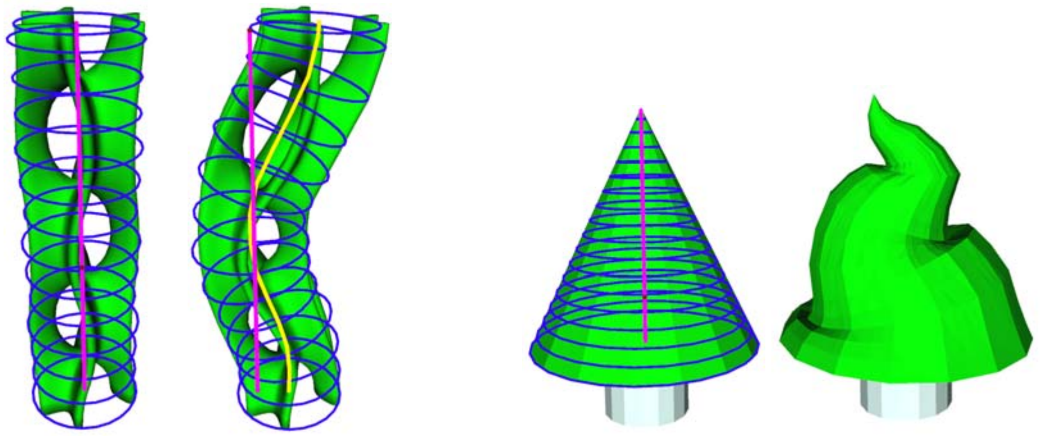


Figure 13: (a) Scherk-Collins tower approximated by a progressive sweep; (b) effect of editing the spine curve (magenta -> yellow); (c) a sweep with a large cross section; (d) “folding” of the surface mesh.

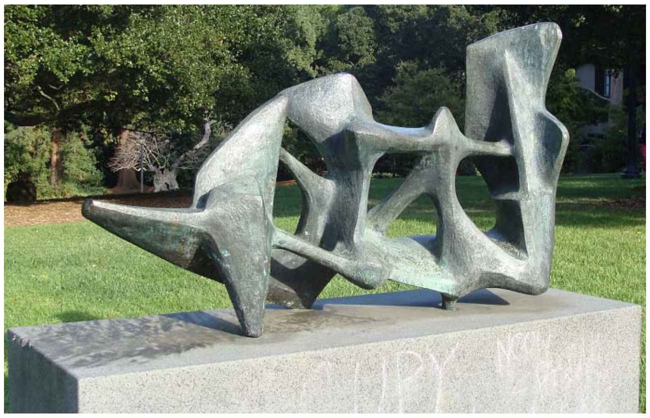
To produce a displacement map, we project triangles back to the primitive surface and rasterize the closest distances to the displacement map. To preserve raw geometry, we have found that it is normally sufficient to simply map the vertices of the detail geometry to the closest points on the primitive surface. This works acceptably when the detail geometry follows the primitive modeling surface fairly closely. However, to avoid awkward behavior near folds in more interesting cases, a more globally smooth and consistent mapping is required. Fortunately, such mappings exist, and one practical method to do this has been described in detail by Peng et al. [25]. Once either type of map is created, details can be transformed in the same way as the shape they exist on; any scaling, rotation, and translation applied to the surface should also be applied to the displacement vectors.

An example is shown in Figure 13. A Scherk-Collins tower is roughly approximated as a progressive sweep. As the spine curve of this sweep is modified, the entire hole-saddle chain is correspondingly deformed. This works well, if the cross section is not too large and the deformations applied to the spine are gentle. Figures 13 c and d show that when these rules are not observed some ‘folding’ of the surface mesh can occur. It is at the discretion of the designer to avoid such situations.

7 CONCLUSIONS AND WORK TO BE DONE

The functional interactive extraction modules described in this paper have shown us that the concept of *Interactive Inverse 3D Modeling* is a versatile and powerful paradigm. We chose to focus on geometry-based modules, guided by the procedural, parameterized primitives most often found in today's CAD tools. Using these primitives, we were able to reconstruct a wide variety of shapes and then immediately perform high-level editing functions on them.

Work, yet to be completed, concerns the integration of the above modules and the creation of a unified interface that allows a complex shape to be described fully in a structured form composed of the parameterized primitives discussed above. Critical domains are where the geometry produced by two or more extraction modules overlap, or in the "un-covered" areas between successfully extracted portions. A mostly automated system of blending and merging functions will yet have to be developed.


(a)

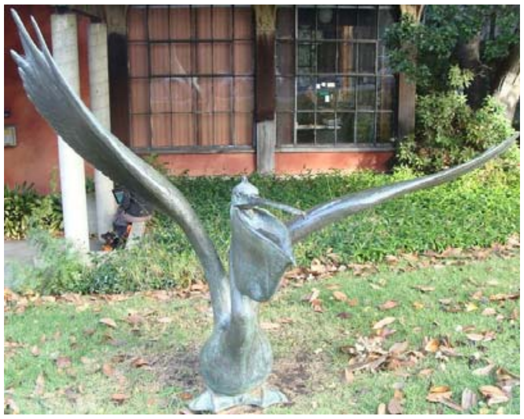

(b)

Figure 14: Two intriguing sculptures that inspire future extensions to our system. (a) An abstract sculpture ("Voyage," by Richard O'Hanlon) composed of blended, ambiguous parts. (b) A pelican sculpture (by Frances Rich) with approximately symmetric structure and wings that could be captured by an offset surface.

As we continue to develop this system, it is helpful to also look at additional examples of shapes we would like to reconstruct and modify with this system, and study what extensions to our system might be necessary. For this purpose we captured several intriguing sculptures on our campus in a series of photographs (Fig.14). We are in the process of reconstructing some of these rather irregular shapes from just a small number of photographs, using the visual hull as the first approximation step.

The Pelican sculpture (Fig.14b) displayed here has already shown a couple of desirable extensions. The wings of this bird cannot be handled well by a tube-like sweep. They would be more appropriately modeled by a 2D-sheet with some slowly varying thickness. A flexible "offset surface" module would give us this capability. Moreover it would be nice if the extracted geometry from one wing could be reflected in the other wing to keep the bird structurally symmetrical. This implies that ideally we want to use only a single prototype wing geometry and then apply some simple transformations and moderate deformations to create two separate instances of wings. The underlying approximate bilateral symmetry of this bird can hopefully be discovered in a semi-automatic manner with the techniques described by [23] and [19].

We are probably about a year away from having a system that could be subjected to a formal user study with participants who want to get some "real work" done with this system. We are quite sure that at that time new demands for new modules and additional capabilities will arise. In the meantime we ask the members of the CAD&A community to give us their forward-looking suggestions.

ACKNOWLEDGEMENTS

This work was supported in part by the National Science Foundation (NSF award #CMMI-1029662 (EDI)) and by Adobe Systems. Thanks to the Image-based 3D Models Archive, Tlcom Paris, and to the Stanford Computer Graphics Laboratory for some of the test meshes used.

REFERENCES

[1] Andrews, J.; Joshi, P.; Séquin, C.: Interactive Extraction and Re-Design of Sweep Geometries, Computer Graphics International 2011.

[2] Benkö, P.; Martin Ralph, R.; Várady, T.: Algorithms for reverse engineering boundary representation models, Computer-Aided Design. 33(11), 2001, 839 - 851. DOI: 10.1016/S0010-4485(01)00100-2.

[3] Bokeloh, M.; Wand, M.; Seidel, H.P.: A Connection between Partial Symmetry and Inverse Procedural Modeling, ACM Transactions on Graphics. 2010. 29(3), 104:1-104:10. DOI: 10.1145/1833349.1778841.

[4] Botsch, M.; Kobbelt, L.: An intuitive framework for real-time freeform modeling, ACM Transactions on Graphics. 23(3), 2004, 630-634. DOI: 10.1145/1015706.1015772.

[5] Chen, Y.; Cheng, Z.-Q.; Li, J.; Martin, R.R.; Wang, Y.-Z.: Relief Extraction and Editing, Computer-Aided Design. 2011. 43(12), 1674-1682. DOI: 10.1016/j.cad.2011.07.011.

[6] Chen, Y.; Davis Timothy, A.; Hager William, W.; Rajamanickam, S.: Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate, ACM Trans. Math. Softw. 35, 2008, 22:1-22:14. DOI: 10.1145/1391989.1391995.

[7] Fischler, M.A.; Bolles, R.C.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, Comm. of the ACM 1981. 24(6), 381–395. DOI: 10.1145/358669.358692.

[8] Fitzgibbon, A.W.; Eggert, D.W.; Fisher, R.B.: High-level CAD Model Acquisition from Range Images, Computer-Aided Design. 1997. 29, 321-330. DOI: 10.1016/S0010-4485(96)00059-0.

[9] Furukawa, Y.; Ponc, J.: Accurate, Dense, and Robust Multi-View Stereopsis, IEEE Transactions on Pattern Analysis and Machine Intelligence. 2010. 32(8), 1362-1376. DOI: 10.1109/TPAMI.2009.161.

[10] Gelfand, N.; Guibas, L.: Shape Segmentation Using Local Slippage Analysis, Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing. 2004, 214-223. DOI: 10.1145/1057432.1057461.

[11] Gingold, Y.; Igarashi, T.; Zorin, D.: Structured Annotations for 2D-to-3D Modeling, ACM Transactions on Graphics (TOG). 28(5), 2009, 148. DOI: 10.1145/1661412.1618494.

[12] Gingold, Y.; Zorin, D.: Shading-based surface editing, ACM Transactions on Graphics. 2008. 27(3), 95:1-95:9. DOI: 10.1145/1360612.1360694.

[13] Golub, G.H.; Van Loan, C.F., *Matrix Computations*. Third ed. 1996.

[14] Hofer, M.; Odehnal, B.; Pottmann, H.; Steiner, T.; Wallner, J.: 3D shape recognition and reconstruction based on line element geometry, Tenth IEEE International Conference on Computer Vision. 2005. DOI: 10.1109/ICCV.2005.2.

[15] Kara, L.B.; D'Eramo, C.M.; Shimada, K.: Pen-based Styling Design of 3D Geometry Using Concept Sketches and Template Models, Proceedings of the 2006 ACM symposium on Solid and physical modeling. 2006, 149-160. DOI: 10.1145/1128888.1128909.

[16] Krishnamurthy, V.; Levoy, M.: Fitting smooth surfaces to dense polygon meshes, SIGGRAPH. 1996, 313-324. DOI: 10.1145/237170.237270.

[17] Li, Y.; Wu, X.; Chrysathou, Y.; Sharf, A.; Cohen-Or, D.; Mitra, N.: GlobFit: Consistently Fitting Primitives by Discovering Global Relations, ACM Transactions on Graphics (Proceedings of SIGGRAPH 2011). 2011. 30(4). DOI: 10.1145/2010324.1964947.

[18] Lia, M.; Langbeina, F.C.; Martin, R.R.: Detecting design intent in approximate CAD models using symmetry, Computer-Aided Design. 2009. 42(3), 183–201. DOI: 10.1016/j.cad.2009.10.001.

[19] Lipman, Y.; Chen, X.; Daubechies, I.; Funkhouser, T.: Symmetry Factored Embedding and Distance, ACM Transactions on Graphics (SIGGRAPH). 2010. 29. DOI: 10.1145/1778765.1778840.

[20] Lloyd, S.P.: Least squares quantization in PCM, IEEE Transactions on Information Theory 1982. 28 (2), 129–137. DOI: 10.1109/TIT.1982.1056489.

[21] Madsen, K.; Nielsen, H.B.; Tingleff, O.: Methods for Non-Linear Least Squares Problems (2nd ed.). 2004, 60.

[22] Mehra, R.; Zhou, Q.; Long, J.; Sheffer, A.; Gooch, A.; Mitra, N.J.: Abstraction of Man-Made Shapes, ACM Transactions on Graphics. 2009. 28(5). DOI: 10.1145/1618452.1618483.

[23] Mitra, N.; Guibas, L.; Pauly, M.: Symmetrization, ACM Transactions on Graphics (SIGGRAPH). 2007. 26(3), 63:1-63:8. DOI: 10.1145/1275808.1276456.

[24] Pauly, M.; J. Mitra, N.; Wallner, J.; Pottmann, H.; Guibas, L.: Discovering Structural Regularity in 3D Geometry, ACM Transactions on Graphics. 2008. 27(3), 1-11. DOI: 10.1145/1360612.1360642.

[25] Peng, J.; Kristjansson, D.; Zorin, D.: Interactive modeling of topologically complex geometric detail, ACM Transactions on Graphics. 2004. 23(3), 635-643. DOI: 10.1145/1015706.1015773.

[26] PhotoModeler, http://www.photomodeler.com/, Eos Systems.

[27] Pinkall, U.; Polthier, K.: Computing Discrete Minimal Surfaces and Their Conjugates, Experimental Mathematics. 2, 1993, 15-36.

[28] Pottmann, H.; Randrup, T.: Rotational and helical surface approximation for reverse engineering, Computing. 60, 1998, 307-322. DOI: 10.1007/BF02684378.

[29] Protopsaltis, A.I.; Fudos, I.: A Feature-Based Approach to Re-engineering CAD Models from Cross Sections, Computer-Aided Design & Applications. 2010. 7(5), 739-757. DOI: 10.3722/cadaps.2010.739-757.

[30] Séquin, C.: Virtual Prototyping of Scherk-Collins Saddle Rings, Leonardo. 30(2), 1997, 89-96. DOI: 10.2307/1576417.

[31] Sorkine, O.; Cohen-Or, D.: Least-Squares Meshes, Shape Modeling International. 2004, 191-199. DOI: 10.1109/SMI.2004.1314506.

[32] Sorkine, O.; Cohen-Or, D.; Lipman, Y.; Alexa, M.; Rössl, C.; Seidel, H.-P.: Laplacian Surface Editing, Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing. 2004, 179-188. DOI: 10.1145/1057432.1057456.

[33] Takayama, K.; Schmidt, R.; Singh, K.; Igarashi, T.; Boubekeur, T.; Sorkine, O.: GeoBrush: Interactive Mesh Geometry Cloning, Computer Graphics Forum. 2011. 30(2), 613-622. DOI: 10.1111/j.1467-8659.2011.01883.x.

[34] Taubin, G.: Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation, IEEE Trans. Pattern Analysis and Machine Intelligence. 1991. 13, 1115-1138. DOI: 10.1109/34.103273.

[35] Taubin, G.: An improved algorithm for algebraic curve and surface fitting, International Conference on Computer Vision. 1993, 658-665. DOI: 10.1109/ICCV.1993.378149.

[36] Várady, T.; Facell, M.A.; Terék, Z.: Automatic extraction of surface structures in digital shape reconstruction, Computer-Aided Design. 2007. 39(5), 379-388. DOI: 10.1016/j.cad.2007.02.01.

[37] Várady, T.; Martin, R.; Cox, J.: Reverse Engineering of Geometric Models - An Introduction, Computer Aided Design. 29, 1997, 255-268. DOI: 10.1016/S0010-4485(96)00054-1.

[38] Yan, D.-M.; Liu, Y.; Wang, W.: Quadric Surface Extraction by Variational Shape Approximation, Geometric Modeling and Processing. 2006. 4, 73-86. DOI: 10.1007/11802914_6.

[39] Yoon, S.-H.; Kim, M.-S.: Sweep-based Freeform Deformations, Computer Graphics Forum. 25(3), 2006, 487-496. DOI: 10.1111/j.1467-8659.2006.00968.x.

[40] Zimmermann, J.; Nealen, A.; Alexa, M.: SilSketch: automated sketch-based editing of surface meshes, Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling. 2007, 23-30. DOI: 10.1145/1384429.1384438.