Computer Graphics International 2011

Interactive Extraction and Re-Design of Sweep Geometries

James Andrews · Pushkar Joshi · Carlo Séquin

Abstract We introduce two interactive extraction modules that can fit the parameters of generalized sweeps to large, unstructured meshes for immediate, high-level, detail-preserving modification. These modules represent two extremes in a spectrum of parameterized shapes: rotational sweeps defined by a few global parameters, and progressive sweeps forming generalized cylinders with many slowly varying local parameters. Both modules are initialized and controlled by the user drawing a few strokes onto the displayed original model. We demonstrate the system on various shapes, ranging from clean, mechanical geometries to organic forms with intricate surface details.

Keywords Computer-Aided Design · Object Modeling · Reverse Engineering · Rotational Sweeps · Generalized Cylinders · Progressive Sweeps

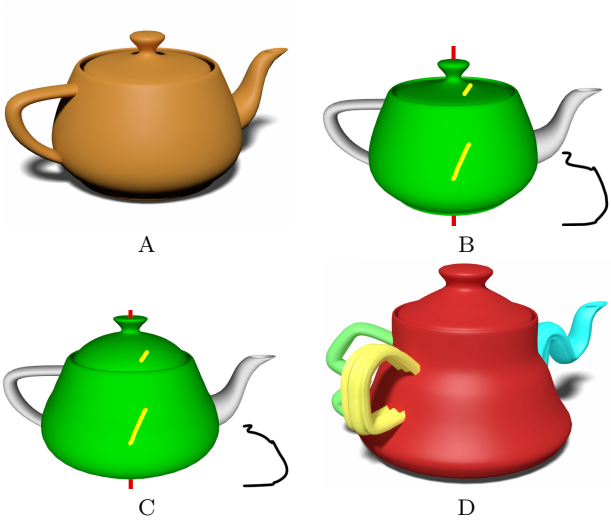

Fig. 1 A: the input surface; B: user draws two strokes (yellow) to extract the teapot body as a stationary, rotational sweep; C: user edits the cross section; D: final surface with modified cross sections and sweep paths of progressive sweeps (fitted locally to the spout and handle) and modified profile of a rotational sweep (fitted to the teapot body). Two different modified versions of the handle were added for variety.

1 Introduction

Shape editing and re-design efforts often start with dense triangle meshes. Many *reverse engineering* approaches to this problem aim to fit a higher-level model such as a set of NURBS patches [9]. The best model depends on user intent, which often changes during redesign.

We propose a system that lets the user interactively indicate a desired high-level model, and which then returns a fitted model in seconds so that the user can quickly switch to the best model for their current task. We choose to focus on *sweeps*, since sweeps are versatile constructions that can represent a wide variety of shapes with a modest set of parameters.

We demonstrate two practical modules for reverse engineering, for two quite different types of geometry. The first one extracts what we call *stationary* sweeps characterized by a few *global* parameters. These are generalizations of rotational sweeps. The second module extracts what we call *progressive* sweeps, characterized by a larger number of smoothly varying *local* parameters. These produce a kind of generalized cylinder. Each module can extract shapes that would be difficult or impossible to approximate with the other one. Fig. 1 shows both modules in use.

The user is integral to our system. With one cursor stroke, the user sets the starting parameters for a good

J. Andrews · C. Séquin
UC Berkeley, Soda Hall, Berkeley, CA 94720
E-mail: jima@eecs.berkeley.edu

P. Joshi
Adobe Systems Inc., 345 Park Avenue San Jose, CA 95110

sweep extraction. This user initialization focuses computational effort and produces just the sweep marked by the user stroke. After a sweep has been extracted, the system offers the sweep parameters as edit handles, so that the user can re-design the shape (Fig. 2).

2 Related Work

Extracting high-level structure from a low-level shape representation has been extensively studied [15] and is available in many modeling packages (e.g., [3]). In CAD, the goal of reverse-engineering was part remanufacturing. Therefore, CAD approaches typically convert point clouds into spline patches by a slow and mostly manual process, prioritizing accuracy over speed or memory usage. In graphics, the focus has been on full automation [8] or artistic control [9].

There is a long history of fitting sweep primitives. A few systems have fit sweep surfaces with fixed orientation of the sweep cross-section, e.g. [4] and [14]. Most of the work on sweeps has focused on finding linear extrusions of planar curves, and on identifying local rotational or helical symmetry, e.g. [10]. These are examples of what we call *stationary* sweeps. Pottmann et al. [12] describe how to convert a point cloud to a rotational sweep primitive using line geometry. Gelfand and Guibas [6] and Benko et al. [1] use the optimization framework of Pottmann and Randrup [12] to identify regions that are 'slippable' (i.e. kinematically equiform) regions. Similarly, our module for extracting stationary sweeps uses the method from Hofer et al. [7] (conceptually similar to [12]) immersed in an optimization loop that also considers input provided by the user.

Our module for fitting progressive sweeps builds on the general recognition-and-optimization framework described by Ramamoorthi and Arvo [13]. That system extracts relatively simple sweeps by asking the user to place a cylinder as a first approximation, and then iteratively optimizing parameters to deform it into the given surface. However, it is often not clear where to place the cylinder to align it with a complex sweep surface (such as Fig. 4). Moreover, the general framework in [13] simultaneously optimizes too many parameters to permit fast optimization.

Compared to previous work on sweep finding, our method can find sweeps along arbitrary paths and with arbitrary rotations (twists) and non-uniform scaling of the cross-section. We also focus on interactive performance for an integrated mesh editing application. We demonstrate our algorithm on some detailed organic shapes that would not typically be handled by sweep-fitting methods. Our system can fit sweeps quickly to a wide variety of shapes, and can easily be used for structured mesh editing.

3 Sweep Extraction Modules

3.1 Stationary Sweeps with Global Parameters

Our *stationary sweep* surfaces are generated by rotating an arbitrary planar profile around a fixed axis in space, while optionally also translating it and/or scaling it at a constant rate. This includes surfaces of revolution, helices and spirals. All surfaces generated this way are composed of trajectories in a vector field describable by only seven global parameters [7].

Our fitting algorithm is a region growing approach: It starts from a few seed points provided by the user and looks for neighboring points that fit into the same 7-parameter vector field. Alternatingly, we fit a sweep model to the given points and then locally expand the surface covered by adding points in a flood-fill mode that fit within the current error margins. This process is given as Algorithm 1.

Algorithm 1 Fit stationary sweep

1: Initialize sweep s to set of points m marked by user
2: **while** s continues to change **do**
3: Estimate parameters $\mathbf{p}$ from s [7]
4: Find max distance t from m to $\mathbf{p}$
5: s = flood fill from m to points closer to $\mathbf{p}$ than t
6: **end while**

Our low-level fitting algorithm (line 3 in Algorithm 1) follows directly from Hofer et al. [7]. We differ in the remainder of our approach because we prefer a bottom-up region growing approach to their top-down method. Specifically, a bottom-up process fits naturally with the user specifying the portion of the model they want to extract, by painting points onto a given surface. If the extracted sweep does not cover enough of the given model surface, the user may mark additional points that should be included. The algorithm will then increase the current error bounds sufficiently to allow inclusion of those new points, – and possibly several other regions as well.

3.2 Progressive Sweeps with Local Parameters

The *progressive sweep* module is designed to follow generalized cylinders with many unpredictable turns and variations of the local cross section. These sweeps are defined by a smoothly changing, affinely transforming cross section, moving along a finely sampled polyline. Our fitting process is given as Algorithm 2.

Our process starts by extracting an initial cross-sectional template near the middle of the user stroke.

Algorithm 2 Fit progressive sweep

```
Initialize template T based on stroke drawn by user
for d = −1 to 1 step 2 do {Go forwards and backwards}
  while error of fit below threshold do
    Add segment to sweep in direction d
    k := 2
    repeat
      Optimize newest k segments
      k := 2 + k
    until error of fit below threshold or k > 6
  end while
  Remove last segment (for which the fit failed)
end for
```

It traces around the local mesh geometry in a plane that perpendicularly bisects the drawn user stroke. This template is updated during a subsequent optimization step in which a first prismatic sweep segment is fit to the given mesh. The algorithm then refines the cross-sectional template with a new cut perpendicular to this prism axis.

The process then alternates between: Phase A: adding a subsequent segment to the sweep path and optimizing its parameters, and Phase B: fine-tuning the fit of this segment and of the last few segments in a joint optimization of all their parameters, so as to achieve an overall smooth fit of the progressive sweep. The addition of each new sweep segment implicitly extends the sweep path, which emerges as a piecewise linear polyline. For each new segment, the algorithm starts out with a conservative, short extension of the sweep path, and then lets the penalty function below (Eqn. 1) determine the most effective parameters for the next segment. To calculate this penalty function, we regularly sample points p_i on the last k segments of the sweep and use the Levenberg-Marquardt algorithm [11] to minimize the following energy:

$$\begin{aligned}\Sigma_i d(\mathbf{p}_i)^2 + w\Big(&\Sigma_{j=N-k}^{N}\Big(\frac{1}{||\mathbf{s}_j - \mathbf{s}_{j-1}||}\Big)^2 \\ &+\Sigma_{j=N-k}^{N}||\mathbf{x}_j - \mathbf{x}_{j-1}||^2 \\ &+\Sigma_{j=N-k}^{N-1}\kappa(\mathbf{s}_{j-1},\mathbf{s}_j,\mathbf{s}_{j+1})^2\Big)\end{aligned} \quad (1)$$

The first term is the squared distance of a sweep point from the input mesh. The other ones are regularization terms that penalize, respectively: short segments (avoid zero-length steps), changes in the transformation parameters x_j (avoid parameter wobbles), and curvature of the sweep path (prefer smooth curves). A small weight w is chosen to scale the regularization terms, so fitting the data has highest priority. To calculate curvature κ we use the discrete integrated curvature metric [2]. To compute distances $d(\mathbf{p}_i)$ efficiently, we use a precomputed, adaptively-sampled distance field [5]. Note that this procedure, like most non-linear optimizations, relies on some arbitrary constants such as the regularization weight w above. For consistency, we used the same values for our constants across all our examples.

This process continues until no segment can be added under the current error bounds, where error is the maximum distance from sweep surface to input mesh. Initial error bounds are set at a small constant factor above the error of the initial sweep segment. The user may implicitly increase the error bounds by marking additional points that should be included in the current progressive sweep extraction.

4 Mesh Editing

As soon as any sweep model is extracted, the user can use the "handles" of the extracted sweep to edit the mesh without losing any surface details contained in the original input data. Mesh editing is illustrated in Figs. 1, 2, 3 and 4. Fig. 2 shows the additional handles provided for a progressive sweep: the sweep path itself (on the leg), the cross-section template, and the adjustable transformation parameters along the path (in this case, scale).

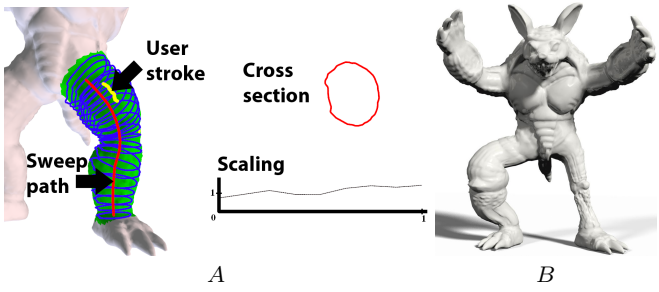

Fig. 2 (A) Key elements of the user-interface: the discovered sweep path and the corresponding cross-sectional template, and a curve that controls the scale transformation applied to this template as it moves along the sweep path. The user can edit the surface by dragging any of the handles. (B) Modified model surface after editing both legs.

Handle-based mesh editing relies on a correspondence between the vertices of the original surface and their closest images on the extracted sweep model. When a structured modification is applied, every mesh vertex is moved exactly as its corresponding image on the sweep surface.

For the stationary, rotational sweeps, we can also reproject all original mesh points with the transformation of the global sweep motion factored out, thereby creating a "fuzzy" profile (Fig. 3C). This collapsed mesh surface can be understood as a collection of all possible profiles encountered on the extracted surface. Portions

of this fuzzy profile can now be selected and moved around, for instance to extend the nose and mouth portions on all the faces (Fig. 3B).

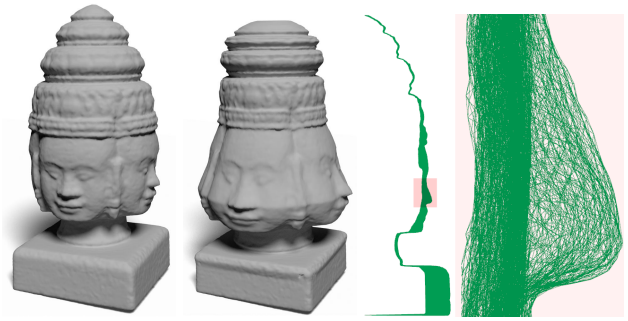

Fig. 3 A: The input surface is a detailed shape with a high-level rotational structure. B: We edit the profile of the shape and transfer that edit back to the original mesh, preserving details. C: The extracted profile: at left: the overall view of the collapsed mesh; at right, a detailed view showing the individual triangles of the input mesh.

5 Results

We tested our modules on a variety of input files (Figs. 1, 2, 3, 4). Fig. 4(A,B) shows the modification of a sweep with a complex path and a twisting cross section. Figs. 2, 3, and 4C demonstrate that we can fit surfaces with complex detail on top of the basic sweep structure.

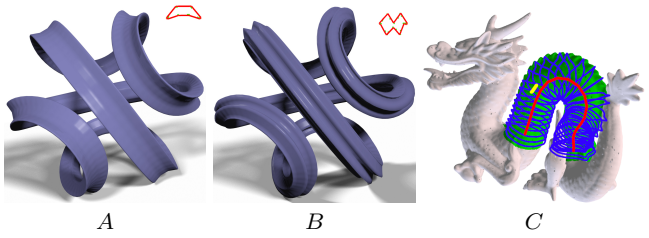

Fig. 4 A: The input surface with its cross-section template in red that twists as it moves along the loop. B: The same sweep with a modified cross section. C: A stroke (yellow) extracts the main hump of the dragon. The default error threshold controls the extent of this sweep

All our examples took less than a minute of user and computer time on a laptop with 2GB RAM and a 2.4 GHz Core Duo processor. All computation times were less than 10 seconds, with model sizes up to 870k triangles (Fig. 4C).

To begin mesh editing, a correspondence from original mesh vertices to the extracted model must be established; this is linear in mesh size, taking 1063 ms for Fig. 4C. Mesh vertices can then be updated interactively, taking 16 ms per update for Fig. 4C.

The two powerful sweep extraction modules discussed in this paper are components of an in-progress *Inverse 3D Modeling* system, which will provide similar interfaces for a wider range of modeling primitives, such as CSG and subdivision surfaces. These sweep modules are an encouraging first demonstration. We found that a few cursor strokes by the user are sufficient to initialize the system, and a straightforward local optimization approach can be made fast enough for our on-demand reverse engineering needs.

Acknowledgements Work was supported in part by the National Science Foundation (NSF award #CMMI-1029662 (EDI)) and by Adobe Systems.

References

1. Benko, P., Martin, R.R., Vrady, T.: Algorithms for reverse engineering boundary representation models. Computer-Aided Design **33**(11), 839 – 851 (2001)
2. Bergou, M., Wardetzky, M., Robinson, S., Audoly, B., Grinspun, E.: Discrete elastic rods. In: ACM Siggraph (2008)
3. Dassault Syst.: Catia. http://www.3ds.com/products/catia
4. Dion D., J., Laurendeau, D., Bergevin, R.: Generalized cylinders extraction in a range image. pp. 141 –147 (1997)
5. Frisken, S.F., Perry, R.N., Rockwood, A.P., Jones, T.R.: Adaptively sampled distance fields: a general representation of shape for computer graphics. SIGGRAPH '00, pp. 249–254 (2000)
6. Gelfand, N., Guibas, L.: Shape segmentation using local slippage analysis. In: Eurographics Sympoium on Geometry Processing (2004)
7. Hofer, M., Odehnal, B., Pottmann, H., Steiner, T., Wallner, J.: 3d shape recognition and reconstruction based on line element geometry. In: Tenth IEEE International Conference on Computer Vision (2005)
8. Hoppe, H., DeRose, T., Duchamp, T., Halstead, M., Jin, H., McDonald, J., Schweitzer, J., Stuetzle, W.: Piecewise smooth surface reconstruction. SIGGRAPH '94, pp. 295–302 (1994)
9. Krishnamurthy, V., Levoy, M.: Fitting smooth surfaces to dense polygon meshes. SIGGRAPH '96, pp. 313–324 (1996)
10. Lai, J.Y., Ueng, W.D.: Reconstruction of surfaces of revolution from measured points. Computers In Industry **41**, 147–161 (2000)
11. Madsen, K., Nielsen, H.B., Tingleff, O.: Methods for non-linear least squares problems (2nd ed.) (2004)
12. Pottmann, H., Randrup, T.: Rotational and helical surface approximation for reverse engineering. Computing **60**, 307–322 (1998)
13. Ramamoorthi, R., Arvo, J.: Creating generative models from range images. In: ACM Siggraph (1999)
14. Ueng, W.D., Lai, J.Y., Doong, J.L.: Sweep-surface reconstruction from three-dimensional measured data. Computer-Aided Design **30**(10), 791–805 (1998)
15. Varady, T., Martin, R., Cox, J.: Reverse engineering of geometric models - an introduction. Computer Aided Design **29**, 255–268 (1997)