EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2008)
C. Alvarado and M.- P. Cani (Editors)

Repoussé: Automatic Inflation of 2D Artwork

Pushkar Joshi[1,2] Nathan A. Carr[2]

[1]U.C. Berkeley, [2]Adobe Systems Inc.

Abstract

We describe a new system for the interactive enhancement of 2D art with 3D geometry. Repoussé creates a 3D shape by inflating the surface that interpolates the input curves. By using the mean curvature stored at boundary vertices as a degree of freedom, we are able to control the inflated surface intuitively and efficiently using a single linear system. Repoussé handles both smooth and sharp position constraints. Position constraint vertices can also have curvature constraints for controlling the inflation of the local surface. We show the applications of our system in font design, stroke design, photo enhancement and freeform 3D shape design.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Line and Curve Generation I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

Sketch based interfaces are becoming popular as a method for quick 3D shape modeling. With an ever-increasing set of modeling features, the powerful 3D sketching interface can construct shapes that range from rectilinear, industrial objects ([ZHH96]) to smooth, organic shapes ([IMT99], [KH06], [NISA07]). Usually, the application of these shape sketching interfaces is 3D shape design, but such interfaces can also be helpful for adding shadows to 2D images [PFWF00]. The 2D curves drawn by the user are simply a means to get to the end result: the 3D shape. Instead of using the 2D curves to design 3D shapes, we use the resulting interpolating 3D shapes to *enhance* the existing 2D curves. That is, we apply the shape sketching interface to another, highly interesting application: 2D art creation and design.

Contribution: By using the 'inflate' metaphor common in shape sketching interfaces, we construct a 3D surface that interpolates a closed input curve. We allow designers to modify the inflated surface with popular 3D shape modeling features such as sharp creases, smooth interpolation curves and local mean curvature control. Our system demonstrates that sophisticated geometric modeling techniques otherwise found in 3D shape modeling tools can effectively be used via a simple user interface to design interesting-looking images.

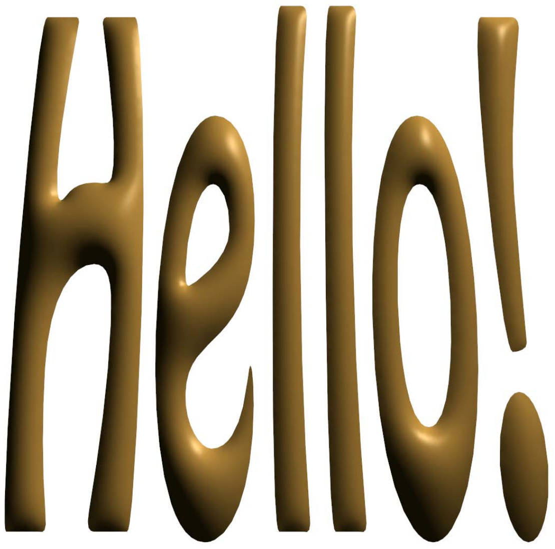

Figure 1: *Boundary curve inflation using Repoussé.*

1.1. Technical Overview

In our system, we construct the 2-manifold surface that interpolates the input boundary curves. The surface is computed as a solution of a variational problem. We formulate the system so as to solve for the final surface in a single, sparse linear equation, without requiring an additional strip

© The Eurographics Association 2008.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

of triangles at the boundary. In our formulation, the mean curvature of the vertices on the boundary curve is a user-controlled degree of freedom; the surface is inflated by increasing these mean curvature values. Due to the variational setup, the inflated surface is always smooth except near position constraints (boundary curves and internal constraints). The designer can add and modify internal constraints at any time of the design phase, and these constraints can be smooth or sharp. Moreover, the internal constraints can also have curvature control (similar to the boundary curves), thereby allowing the designer to locally inflate or deflate the surface near the constraint.

1.2. Related Work

Currently, there are a few 3D shape modeling tools in readily available 2D artistic design software. A commonly used modeling primitive is shape extrusion, where a closed 2D curve is swept along a straight line (or a curve) to create prismatic shapes. Similar to extrusion is curve rotation, where a curve is rotated along an axis to construct a rotationally symmetrical surface of revolution. Another commonly used primitive is beveling, where the input 2D curve is offset and raised to provide an appearance of a 3D shape that has thickness and sharp edges. These 3D modeling primitives are limited in the type of surface features they can support. For example, no image design application supports adding sharp creases in the interior of a beveled image.

However, research is underway to improve the range of surface edits possible in a 2D design tool. Williams [Wil91] introduced the concept of inflating the surface bounded by 2D curves for different shading effects. Similarly, Johnston's 'Lumo' system [Joh02] generated normals (on a flat domain) that gave the flat surface an inflated appearance. Sourin demonstrated work on virtual embossing [Sou01]; Levinski and Sourin [LS07] followed that work with a more general, function-based surface modeling system. Both these papers as well as the ShapeShop [SWSJ05] modeler use an implicit surface representation to model their surfaces (the surface is interactively polygonized for rendering purposes). Liu et al. [LTJ05] and Prasad et al. [PZF05] solve the closely related problem of depth reconstruction for photographs by computing height values over a regular grid. In contrast, we use a polygon (triangle) mesh to inflate the given curves.

The Repoussé approach can be considered in the same category as Teddy [IMT99] and FiberMesh [NISA07], albeit with a different application. Similar to Teddy, we use the inflation metaphor (but without using the chordal axis) and similar to FiberMesh, we allow both smooth and sharp constraint curves drawn directly on the inflated surface to modify the surface. The key difference is that we use the mean curvature constraints to control the amount of inflation, which allows us to formulate the problem as a single, convenient linear system (unlike the two linear systems required in FiberMesh).

As applications of our system, we show examples of font design, stroke design, enhancing photographs and modeling 3D shapes from scratch. In general, Repoussé can be used to improve the functionality of 2D art design tools.

2. Work Flow

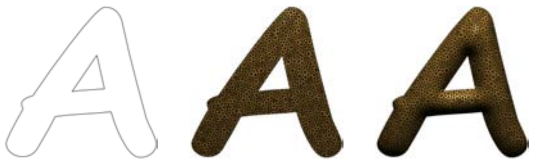

Figure 2: *Repoussé workflow: the flat domain bounded by the input curves is triangulated and the resulting surface is inflated.*

As illustrated in Fig. 2, the Repoussé system takes closed 2D curves as input, triangulates the area bounded by the curves to generate the initial surface, and then inflates the surface while maintaining a fixed boundary. The 2D curves are read in as simple poly-line approximations.

2.1. Triangulation

We restrict the surface representation to a triangle mesh that is obtained by triangulating the surface bounded by the input curves. We use Shewchuk's popular triangulation software 'Triangle' [She96] to generate a high-quality triangulation. We maintain a maximum area constraint (provided as a configurable option in Triangle) to prevent rendering artifacts due to very large triangles.

2.2. Surface Inflation

The unconstrained parts of the surface are obtained by solving a variational system that maintains surface smoothness. Smoothness is necessary because it gives an organic look to the inflated surface and removes any unnatural and unnecessary bumps and creases from the surface.

The variational formulation in Repoussé is based on the principles of PDE-based boundary constraint modeling [BW90], where the Euler-Lagrange equation of some aesthetic energy functional is solved to yield a smooth surface. We picked the 'thin-plate spline' [Wah90] as the desired surface; the corresponding Euler-Lagrange equation is the biharmonic equation. That is, for all free vertices at position $\mathbf{x}$, we solve the PDE $\Delta^2(\mathbf{x}) = 0$. The solution of this PDE yields a C^2 continuous surface everywhere except at the position constraints (where the surface can be either C^1 or C^0 continuous). We use Pinkall and Polthier's [PP93] cotangent-weight based discretization of the laplacian operator $\Delta(\mathbf{x})$ (first used in Equation 2).

The fourth-order PDE ($\Delta^2(\mathbf{x}) = 0$) is too slow to be solved interactively. We convert the non-linear problem into a linear one by assuming that the parameterization of the surface is unchanged throughout the solution. In practice, this means that the contangent weights used for the Laplacian formulation are computed only once (using the flat, non-inflated surface) and are subsequently unchanged as the surface is inflated. This approximation works well enough for our purposes and has been used extensively for constructing smooth shape deformations (for example, see [BK04]).

2.2.1. Linear System

For the sake of completeness, we now describe the entire formulation of our variational linear system. We design a system $\mathbf{A}\bar{\mathbf{x}} = \bar{\mathbf{b}}$ where the matrix $\mathbf{A}$ is a sparse $n \times n$ matrix ($n = 3\times$ the number of free vertices) that represents the local geometric relationships between the vertices and their neighbors. The vector $\bar{\mathbf{x}}$ of length n represents the positions of free vertices and the vector $\bar{\mathbf{b}}$ of length n represents the known quantities. For all three coordinates of every free vertex $\bar{\mathbf{x}}$, we formulate an equation that is linear in terms of the $\bar{\mathbf{x}}$'s neighbors. The formulation that follows is based primarily on two papers on discrete geometric modeling. Botsch and Kobbelt [BK04] describe how to formulate a linear system that can handle smooth or sharp internal constraints; unfortunately the formulation from the paper requires a strip of triangles to complete the one-ring neighborhood of the boundary vertices. Generating this strip of triangles, especially when the boundary curve has large concavities, is not trivial. Schneider and Kobbelt [SK01] describe a surface modeling system that takes G^1 boundary constraints and does not need the special triangle strip on the boundary. But this requires two linear solutions: one for the mean curvature scalar field and another for the positions that satisfy the computed mean curvature field. We combine the benefits of these two approaches: our system does not need a strip of triangles on the boundary and we solve only one linear system. We can do so because we consider the mean curvature at the boundary vertices as a degree of freedom that can be used to inflate the surface.

Consider a mesh vertex of position $\mathbf{x}$ and one-ring neighbors $\mathbf{y}_i$ as shown in Fig. (3). The required C^2 smooth surface can be obtained if we solve for a surface with a vanishing bi-Laplacian at all vertices:

$$\Delta^2(\mathbf{x}) = \Delta(\Delta\mathbf{x}) = 0 \tag{1}$$

The Laplacian at a mesh vertex is given by its one-ring neighborhood $\Delta\mathbf{x} = \mathbf{x} - \sum_i w_i \mathbf{y}_i$, where w_i are the contangent weights [PP93]. Substituting in Equation (1),

$$\Delta^2\mathbf{x} = \Delta(\mathbf{x} - \sum_i w_i \mathbf{y}_i) = 0 \tag{2}$$

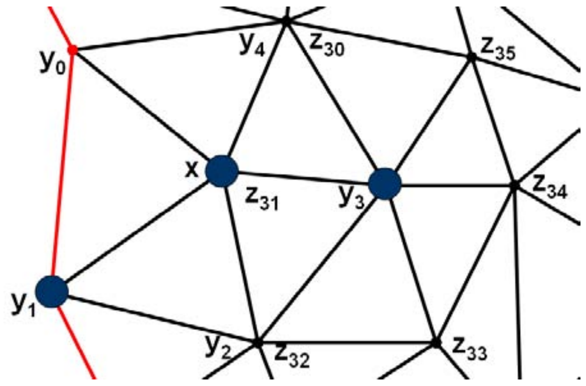

Figure 3: *Variables used in Equation (5) — $\mathbf{x}$ has neighbors $\mathbf{y}_0$,$\mathbf{y}_1$ on the boundary (constrained mean curvature) and neighbors $\mathbf{y}_2$,$\mathbf{y}_3$,$\mathbf{y}_4$ in the interior with a full one-ring neighborhood (unconstrained mean curvature).*

Since Δ is a linear operator,

$$\Delta^2\mathbf{x} = \Delta\mathbf{x} - \sum_i w_i \Delta\mathbf{y}_i = 0 \tag{3}$$

Now consider the situation in Fig. (3), where some one-ring neighbors are in the mesh interior, and some are on the boundary. We assume that the mean curvature of the boundary vertices is *given* as a constraint. Assume $\mathbf{y}_j$ represents the one-ring vertices whose mean curvatures $h_{\mathbf{y}_j}$ are known. For those vertices, we can compute the Laplacians simply by using the expression $\Delta\mathbf{y}_j = (h_{\mathbf{y}_j}\mathbf{n}_{\mathbf{y}_j})/2$ [MDSB02]. Moving such known Laplacians to the right hand side of Equation (3), we get

$$\Delta^2\mathbf{x} = \Delta\mathbf{x} - \sum_i w_i \Delta\mathbf{y}_i = \sum_j \frac{w_j h_{\mathbf{y}_j}\mathbf{n}_{\mathbf{y}_j}}{2} \tag{4}$$

Note that the term $(h_{\mathbf{y}_j}\mathbf{n}_{\mathbf{y}_j})/2$ essentially represents a force of magnitude $0.5h_{\mathbf{y}_j}$ in the direction $\mathbf{n}_{\mathbf{y}_j}$ applied by the neighboring vertex $\mathbf{y}_j$ on vertex $\mathbf{x}$. In the default system, the force is applied in the direction of the *initial* vertex normal (the normal in the flat configuration — the Z axis). We do not use the vertex normals from the inflated state as that produces non-linear vertex motion that is path-dependent and unintuitive.

Therefore, by increasing the value of $h_{\mathbf{y}_j}$, we increase the magnitude of the force on the vertex $\mathbf{x}$, effectively pushing it up.

Finally, we expand the laplacians of vertices with unknown mean curvatures in Equation (3) to get the linear equation for the free vertex $\mathbf{x}$:

$$\mathbf{x} - \sum_i w_i \mathbf{y}_i - \sum_i w_i \left[\mathbf{y}_i - \sum_k w_{ik}\mathbf{z}_{ik}\right] = \sum_j \frac{w_j h_{\mathbf{y}_j}\mathbf{n}_{\mathbf{y}_j}}{2} \tag{5}$$

Constructing such equations for every free vertex gives us the linear system $\mathbf{A}\bar{\mathbf{x}} = \bar{\mathbf{b}}$, whose solution provides the inflated surface.

2.2.2. Internal Constraints

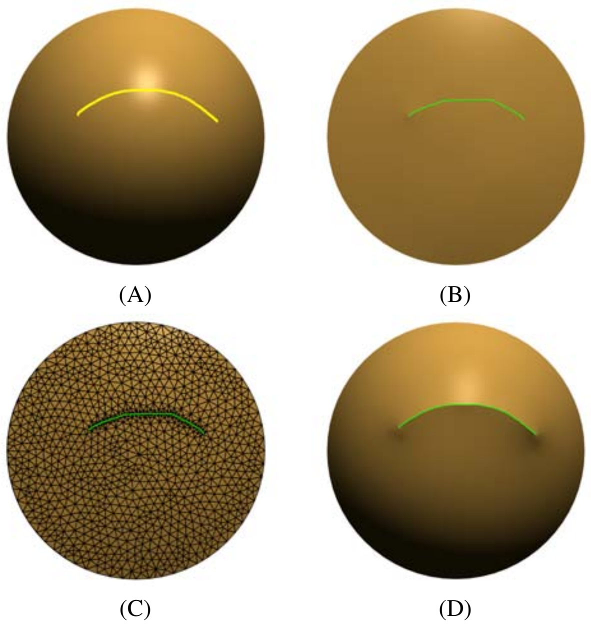

Figure 4: *Steps for adding constraints: suppose the user wants a sharp interior constraint. The user draws a curve on the surface (A). The Repoussé system flattens the inflated surface and computes the 2D position of the constraint curve (B). Repoussé then re-triangulates the domain subject to the constraint (C) and re-inflates the surface and moves the constraint to its original location (D).*

The user can draw constraint curves anywhere on the inflated surface to immediately obtain the new inflated surface with the constraint in place. Upon adding a new constraint, the linear system $\mathbf{A}\bar{\mathbf{x}} = \bar{\mathbf{b}}$ needs to be re-initialized and solved. In order for this to happen interactively, the following steps have to occur (illustrated in Fig. 4).

1. The existing surface (without the new constraint) is flattened by changing the boundary vertex mean curvatures to zero and moving all internal position constraints to their 2D positions.
2. The 2D positions of the new constraint curve vertices are computed by using the barycentric coordinates within the flattened triangle mesh.
3. The area bounded by the boundary curves is triangulated subject to the 2D positions of the internal constraint curves.
4. The resulting surface is re-inflated by setting the boundary vertex mean curvatures to the original values and moving the position constraint curves to their old positions.

Smoothness of Position Constraints The user can specify either smooth (C^1) or sharp (C^0) position constraints. We can vary the smoothness value by assigning a weight to the constrained vertex in Equation (5). The details are in the paper by Botsch and Kobbelt [BK04] and are not repeated here. As mentioned by Botsch and Kobbelt, the weight that controls the smoothness of the constraint curves can actually take any floating point value between 0 (C^0 continuous) and 1 (C^1 continuous). However, for now, we found it more useful to have only two options (smooth/sharp), and to draw curves with a fixed smoothness for all vertices. Future work will explore the use of varying smoothness within individual curves.

2.2.3. Curvature Constraints

The user can specify mean curvature constraints along with position constraints. The curvature constraint can be used to locally inflate or deflate the surface around the position constrained vertices. As such, we get a new sketch-based modeling gesture. In the current implementation, the initial value for the curvature constraint is set to zero, but could easily be set to any arbitrary value.

Options for Curvature Constraints — Gravity As mentioned above, assigning a curvature constraint to a vertex is an indirect method of applying a force to their one-ring neighbors along the direction perpendicular to the initial, flat surface. However, we can easily modify the default behavior and apply additional forces in arbitrary directions. As one example, we add a ‘gravity’ option to the curvature constraints where another force is applied in a slightly downward direction (to the right hand side of Equation 5), causing the entire surface to bend downwards. The goal is to create the illusion of a viscous material on a vertical plane, as seen in Fig. 6 and Fig. 8

3. Applications

3.1. 3D Font Design

We envision one primary application of Repoussé in font design: the outline of the font character can be inflated to provide depth to the font. Moreover, the shape of the inflated character can be controlled and enhanced by adding smooth or sharp position and curvature constraints. Fig. 5 illustrates one example of 3D font design.

3.2. Stroke Design

Instead of inflating already complete 2D curves (like a font outline), Repoussé can also be used as a tool for inflating 2D elements as they are generated. One example is that of strokes. Currently, 2D design tools support vector brush strokes with a variety of brush shapes, thickness, and incident angles. We can add another option to a stroke: depth. We can easily implement varying stroke depth by changing

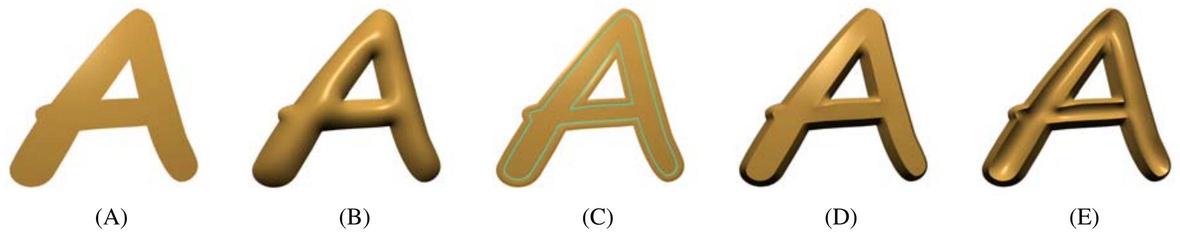

Figure 5: *3D Font design using the Repoussé system — the letter 'A' is read in as a pair of closed boundary curves (A). By increasing the mean curvature at the constrained boundary vertices, the surface bounded by the input curves is inflated (B). Next, internal offset curves are read in as curvature and sharp position constraints (C). The internal curves are raised to give a beveled effect (D) and then made sharper by increasing the curvature at the curve vertices (E).*

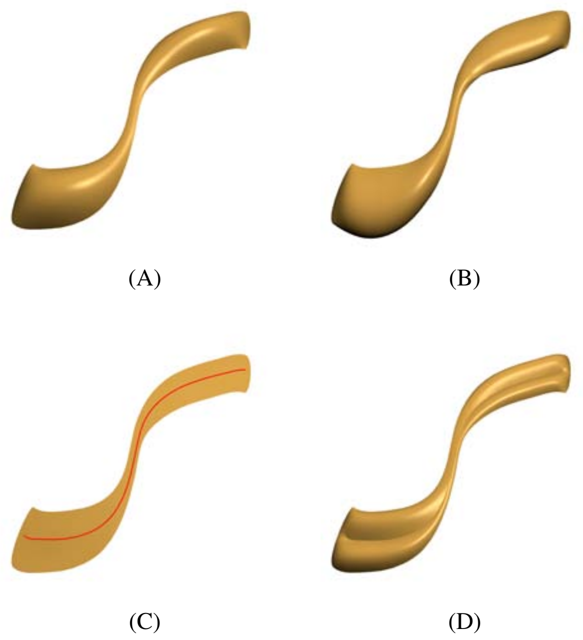

Figure 6: *Stroke design: we used Repousseé to inflate the outline of a paint brush stroke (A). We added the gravity option (Sec. 2.2.3) to produce an effect of paint drip (B). We add the stroke axis as a sharp position constraint (C) and produce a grooved stroke by inflating the rest of the surface (D)*

the mean curvature at the stroke boundary. Moreover, the medial axis of the stroke could be provided as a constraint curve, further increasing the range of stroke shapes possible. Fig. 6 illustrates one example of stroke design.

3.3. Photograph Inflation

A user can interactively add depth to an input photograph simply by drawing and dragging curves on the photograph. The constraint curves can be smooth (for images containing rounded edges) or sharp (images of rectilinear shapes). See Fig. 7 for an example.

3.4. 3D Shape Modeling

While we have targeted Repoussé to be primarily a 2D art enhancement tool, we can also use it to design arbitrary 3D shapes. By adding smooth or sharp position and curvature constraints, we can modify the default inflated shape. Please refer to Fig. 8 for an example.

4. Discussion

In this paper, we have shown that modeling a complete 3D interpolating surface can greatly increase the options for the design of 2D curves. All the examples shown in this paper were computed interactively, with the maximum initialization time less than a second and the maximum iterative update time less than a tenth of a second. The number of faces in the meshes used for surface inflation were in the 2000 to 10000 range; all processing was done on an Intel Core-2 2.66 Ghz. processor with 2GB RAM.

However, we believe we can even further improve the system performance. Currently, we use an in-house conjugate-gradient implementation to solve the linear system $\mathbf{A}\bar{\mathbf{x}} = \bar{\mathbf{b}}$. Since the matrix $\mathbf{A}$ is sparse, symmetric and positive-definite, we could greatly speed up the iterative update times by factorizing the matrix. For example, Botsch et al. [BBK05] demonstrated real-time deformations for much larger meshes by performing a Cholesky decomposition of the matrix $\mathbf{A}$ and using a direct solver to solve the linear system. We plan on implementing this improvement in our current system.

We have developed a prototype user interface to test our algorithm. Our system allows the user to interactively draw

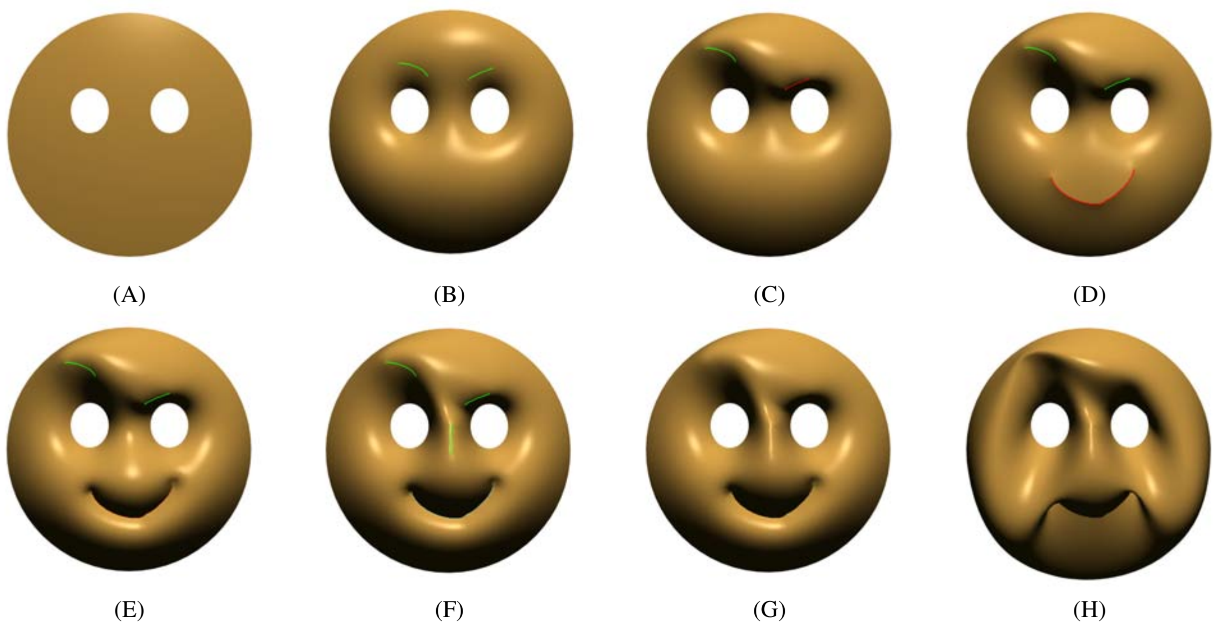

Figure 8: *Freeform 3D shape design: given the outline of a cartoon face (A), we inflate the interior and then add two smooth position constraints near the eyebrows (B). One of the eyebrows is pulled up and the other is pulled down (C). Then we add a sharp position and curvature constraint near the mouth (D) and inflate the nearby surface by increasing the mean curvature (E). Finally, we add a smooth position constraint near the bridge of the nose (F) to get the final surface (G). We can also add the gravity option (Sec. 2.2.3) for this 3D shape to create a droopy effect (H).*

and modify curve constraints directly on the model using the mouse and a collection of hotkeys. The surface is interactively updated by our system proving continuous feedback. As future work we plan on streamlining our user interface toward specific applications making our system even more intuitive to the novice user.

A big advantage of solving for the inflated surface using a triangle mesh (as opposed to a regular grid) is efficiency due to mesh adaptivity: the mesh has high detail only near complicated constraints and is coarse where there are not many constraints. However, currently, the mesh connectivity is not updated as the surface is inflated. Ideally, the mesh should dynamically get denser in parts of the inflated shape that have high curvature. Adding such adaptivity will improve the system even further, making it more efficient and smoother in terms of rendering.

4.1. Conclusions

Our system demonstrates that faster computers, better linear system formulations and improvements in sketching interfaces have now made it feasible to construct a full 3D surface in order to create an appealing 2D image. Our hope is that 3D geometric modeling tools like Repoussé can be used by 2D artists to improve and influence their existing artwork.

Acknowledgments

We would like to acknowledge the members of Adobe's technology labs including Paul Asente, Gerard Baron, Sunil Hadap, Gavin Miller, and Steve Schiller who helped by providing technical assistance and inspiration for this project along the way.

References

[BBK05] BOTSCH M., BOMMES D., KOBBELT L.: Efficient linear system solvers for mesh processing. In *Mathematics of Surfaces XI*. Springer Berlin / Heidelberg, 2005, pp. 62–83.

[BK04] BOTSCH M., KOBBELT L.: An intuitive framework for real-time freeform modeling. *ACM Trans. Graph. 23*, 3 (2004), 630–634.

[BW90] BLOOR M. I. G., WILSON M. J.: Using partial differential equations to generate free-form surfaces: 91787. *Comput. Aided Des. 22*, 4 (1990), 202–212.

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: A sketching interface for 3d freeform design. In *SIGGRAPH* (1999), pp. 409–416.

[Joh02] JOHNSTON S. F.: Lumo: illumination for cel animation. In *NPAR '02: Proceedings of the 2nd interna-*

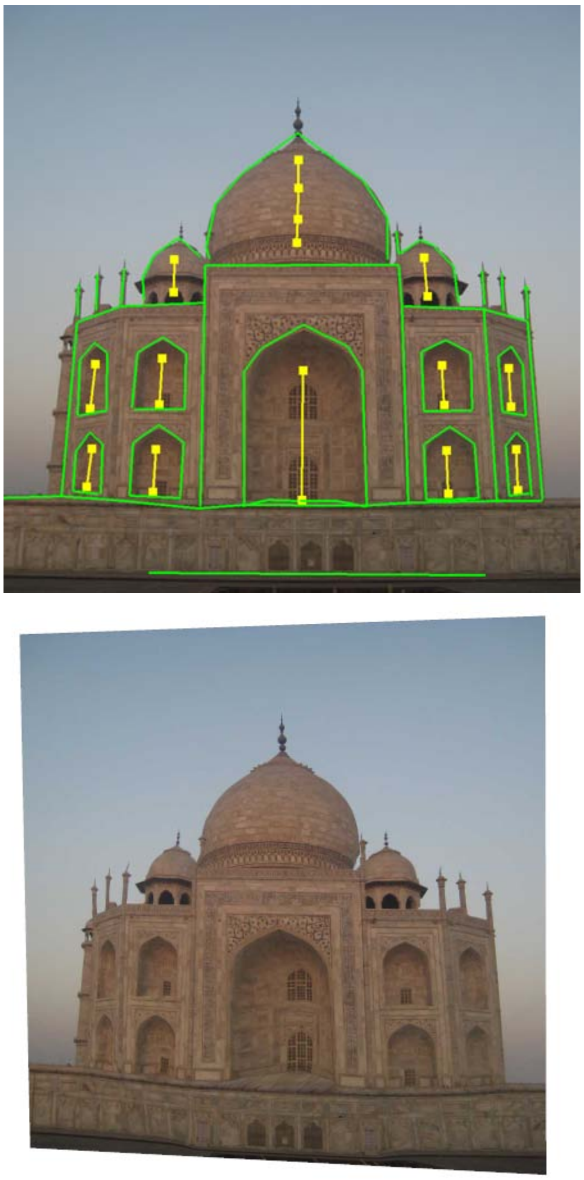

Figure 7: *Photo. Inflation of the Taj Mahal using Repoussé. Constraint curves of varying types(smooth, sharp,and curvature constrained) were specified (top) to define the inflated shape (bottom).*

tional symposium on Non-photorealistic animation and rendering (New York, NY, USA, 2002), ACM, pp. 45–ff.

[KH06] KARPENKO O. A., HUGHES J. F.: Smoothsketch: 3d free-form shapes from complex sketches. *ACM Transactions on Graphics 25/3* (2006), 589–598.

[LS07] LEVINSKI K., SOURIN A.: Interactive function-based shape modelling. *Computers & Graphics 31*, 1 (2007), 66–76.

[LTJ05] LIU Y.-J., TANG K., JONEJA A.: Sketch-based free-form shape modelling with a fast and stable numerical engine. *Computers & Graphics 29*, 5 (2005), 771–786.

[MDSB02] MEYER M., DESBRUN M., SCHR P., BARR A.: Discrete differential geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*. Springer-Verlag, Berlin, 2002.

[NISA07] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: Fibermesh: designing freeform surfaces with 3d curves. *ACM Trans. Graph. 26*, 3 (2007), 41.

[PFWF00] PETROVIC L., FUJITO B., WILLIAMS L., FINKELSTEIN A.: Shadows for cel animation. In *Siggraph 2000, Computer Graphics Proceedings* (2000), Akeley K., (Ed.), ACM Press / ACM SIGGRAPH / Addison Wesley Longman, pp. 511–516.

[PP93] PINKALL U., POLTHIER K.: Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics 2*, 1 (1993), 15–36.

[PZF05] PRASAD M., ZISSERMAN A., FITZGIBBON A. W.: Fast and controllable 3D modelling from silhouettes. In *Eurographics, Short Papers* (September 2005), pp. 9–12.

[She96] SHEWCHUK J. R.: Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, Lin M. C., Manocha D., (Eds.), vol. 1148 of *Lecture Notes in Computer Science*. Springer-Verlag, May 1996, pp. 203–222. From the First ACM Workshop on Applied Computational Geometry.

[SK01] SCHNEIDER R., KOBBELT L.: Geometric fairing of irregular meshes for free-form surface design. *Computer Aided Geometric Design 18*, 4 (2001), 359–379.

[Sou01] SOURIN A.: Functionally based virtual computer art. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics* (New York, NY, USA, 2001), ACM, pp. 77–84.

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

[Wah90] WAHBA G.: *Spline models for observational data*, vol. 59. SIAM, Philadelphia, PA, USA, 1990.

[Wil91] WILLIAMS L.: Shading in two dimensions. In *Graphics Interface* (1991), pp. 143–151.

[ZHH96] ZELEZNIK R. C., HERNDON K. P., HUGHES J. F.: SKETCH: An interface for sketching 3D scenes. In *SIGGRAPH 96 Conference Proceedings* (1996), Rushmeier H., (Ed.), Addison Wesley, pp. 163–170.