Proceedings of Proceedings of IDETC/CIE 2005
2005 ASME International Design Engineering Technical Conferences
& Computers and Information In Engineering Conference
Long Beach, CA, USA, September 24-28, 2005

DETC2005-85363

CONVERTING MOLECULAR MESHES INTO SMOOTH INTERPOLATORY SPLINE SOLID MODELS

Joel Daniels II
Elaine Cohen
David Johnson
School of Computing
University of Utah
Salt Lake City, Utah 84112
Email: {jdaniels, cohen, dejohnso}@cs.utah.edu

ABSTRACT

The study and understanding of molecules, once the domain of blackboards and stick-and-ball models, has become more and more exclusively linked to the use of computer-aided visualizations. Our project seeks to return the physical facsimile to the biologists, allowing the use of tactile senses while interacting with and manipulating a physical model, thus aiding educational and research endeavors. To increase the effectiveness of such a tool, the model is constructed such that multiple levels of information are viewable within the single physical form, stressing the interaction between the assorted components within the molecule. We use the term *3-D physical visualizations* to refer to the fabricated model, to avoid confusion with the common usage of model as a virtual representation on the computer.

To effectively combine multiple components into a smooth manufacturable physical visualization, all components of the model must be in a homogeneous format. Our research sets forth a method for converting triangulated mesh data, as provided by the molecular modeling packages, into spline models. Spline models have the attractive qualities that they are smooth without triangular facets, can be combined using traditional boolean operations (and, or, not), and can be directly fabricated using modern CAD/CAM techniques. Our method divides the polyhedral representation into multiple rectangular grids, then fits interpolatory spline surfaces to the data in each region, while focusing on smoothly stitching the boundaries and corners of the spline surfaces in order to create a near G^1 continuous model.

Figure 1. A physical representation of a protein subunit of a hemoglobin. The model, fabricated using the developed system, shows a peptide chain colored based amino acids snaking through a clear plastic representing the associated molecular surface.

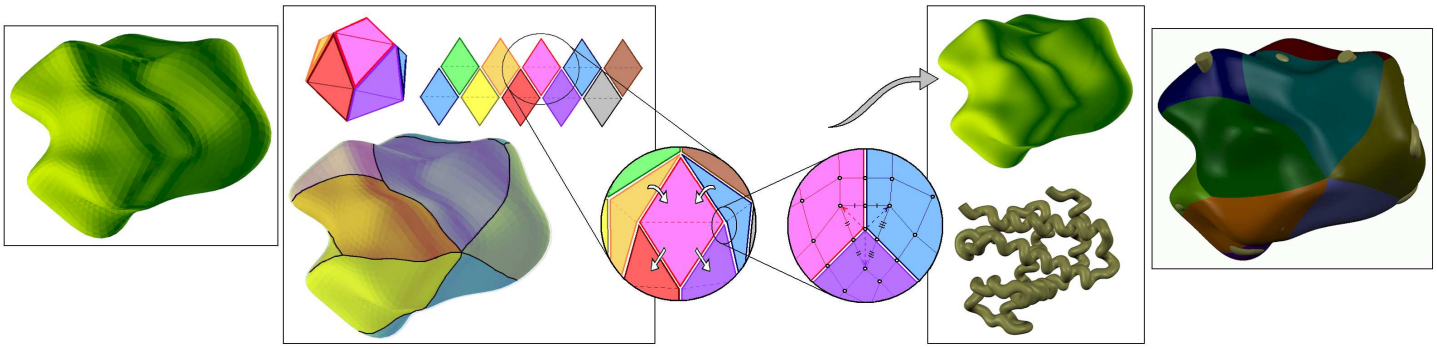

Figure 2. Illustration of the conversion pipeline of a molecular triangular mesh into a spline model. The system consists of the following steps: (a) the input triangular mesh has connectivity equivalent to the n^{th} subdivision level of an icosahedron, (b) the data is segmented into 10 rectangular data grids, (c) tangents are fit across adjacent boundaries of each grid, (d) tangents and twists are computed to smoothly stitch each corner region, and (e) complete spline interpolation creates spline surfaces from the collected data. (f) Boolean operations combine two spline models to form a solid model during the fabrication process.

INTRODUCTION

The historic stick-and-ball molecular model was advantageous because it enabled scientists to use multiple senses to understand and reason about their data. Unfortunately, these physical models have become inadequate given the complexity of modern structural biology, where multiple layers of information are embedded within the molecular structure. Our work seeks to address these concerns by combining multiple elements into a single realizable physical model. We will henceforth term such a model a *3-D physical visualization*, reserving the term *model* to refer to a virtual computer model.

Computer graphics and immersive environments have become the standard method for investigating and visualizing molecules. Recently structural molecular biologists have collaborated with computer scientists to manufacture interesting physical visualizations. Unlike virtual models, these concrete constructions allow the added understanding given by sensory and tactile feedback, as well as a better understanding of the 3D relationships between molecules.

The concept of the 3D physical visualization is shown in Figure 1. Here, two elements are incorporated within the constructed molecule; the molecule's backbone and the associated molecular surface. The protein backbone protrudes from the molecular surface in multiple locations (based on the degree of the spherical harmonic used to approximate the molecular surface), thus becoming part of the defining surface of the molecule. Both the backbone and the surface geometry are necessary to the understanding of the molecules' function. Consequently, we seek to combine these two elements in the fabrication in a manner that facilitates scientists' understanding of the molecular function.

The backbone and molecular surface are represented in heterogeneous formats. The molecular surface is stored as a triangular mesh, produced by the sampling methods of the molecular visualization system [1] [2] [3]. The protein strand is represented as a spline model, formed by a circular tube swept through the center of the key defining atoms. While computer visualizations can combine disparate data sets, manufacturability is improved by higher level representations such as spline surfaces.

In this paper, we define a system that converts a triangular molecular mesh into a spline model. The methods maintain an accurate representation of the original data by constraining the spline model to interpolate the vertices of the input triangular mesh. The new model has two main advantages over the mesh representation. First, the continuity of the splines more accurately captures the smooth flow of the molecular surface, removing the planar artifacts inherent in triangular representations. Second, splines are conducive to performing the boolean operations which are needed to combine the surface and backbone elements in modern CAD systems. These operations form solid models from two components, from which a split away mold is created by dividing the model into two halves.

For example, the Stratasys Fused Deposition Modeling Rapid Prototyping Machine is used to construct a physical form from a solid model of a molecular surface combined with its associated protruding protein strand. A flexible split away mold is created from the physical visualization. Meanwhile, the protein backbone is separately produced in opaque plastics using the Z-corp 3D color printer. By placing the realized backbone in the mold and injecting a clear plastic, a 3D physical visualization of the hemoglobin, the constructed protein strand and molecular surface shown in Figure 1, is achieved.

Our algorithm creates a near G^1 continuous model by decomposing the input mesh into ten spline surfaces, as seen in Figure 2. The splines are defined to maintain C^2 continuity over the interior points of each spline surface. The boundaries and

corners require special considerations to ensure smoothness. The key to our algorithm is its ability to minimize the ridges along these boundaries, thus generating smooth spline models, while utilizing conventional complete spline interpolation techniques to fit the surface data.

System Overview

Figure 2 illustrates the pipeline developed for the conversion process. The pipeline input is a triangular mesh that is topologically equivalent to an icosahedron. The mesh's connectivity is the byproduct of sampling techniques used to obtain some molecular data, including the models that motivate this work. The triangles of the input mesh are mapped to the 20 faces of an icosahedron.

The system unrolls and pairs the 20 faces of the icosahedron, forming 10 rectangles that define the data grid for each bicubic surface. Each rectangular data grid is extracted by walking the vertices of the triangles within the paired faces. Next, cross-boundary tangents are estimated across all edges of adjacent grids. The system then computes a configuration of cross-boundary tangents and twists for each corner, where either 3 or 5 surfaces meet, to minimize G^1 discontinuities. Conventional complete spline interpolation methods create the final spline surfaces from the collected data grids and tangential boundary vectors.

The remainder of the paper is organized as follows. The previous work section summarizes similar efforts in surface subdivision, spline fitting, and corner stitching. The implementation section further describes the details, methods, and motivations of the pipeline highlighted by this section. Analysis of the conversion process results and the mathematical properties of the corner configuration is presented in the case studies section. The last section provides concluding remarks, reporting the results of the conversion process, as well as stressing the importance of the issues addressed by this work.

PREVIOUS WORK

Many research topics exert efforts in solving mesh smoothing to better define a model. The most popular techniques utilize subdivision surfaces. Catmull-Clark [4], Doo-Sabin [5], and Loop [6] schemes each develop refinement methods to recursively define new smoother meshes. Stencils weight existing information in order to compute the locations of new vertices, edges and faces. Subdivision schemes produce smooth surfaces, which, in their limit are equivalent to a spline surface of a given degree. These refined meshes add smooth detail between the known data; however, they fail to interpolate the original vertices as each recursive step shrinks the model. Thus, such an implementation will not maintain an accurate representation of the input molecular data.

Algorithms designed to fit surfaces to point cloud data sets address similar smoothing and data interpolation considerations. Xie et al. [7] create C^1 models by growing a surface over the data set. The prioritized expansion fits quadrics to the local data points along the frontier of the growing surface, further adding to the known surface. The recent efforts of Cheng et al. [8] present an iterative method to fit a Loop subdivision surface to an unorganized point cloud. The defined surface converges toward the original data by optimizing a defined square distance minimization method. Similarly Hoppe et al. [9] fit Loop subdivision surfaces to scattered data, focusing on constructing smooth piecewise surfaces. Hoppe also presents modifications to Loop's subdivision rules in order to model sharp features. The quadric surfaces and triangular meshes closely approximate the input data without the model volume reduction of subdivision surfaces. However, these methods do not provide the means to combine, within the model, distinct sets of molecular data.

Alternatively, multiple non-uniform B-spline surfaces (NURBS) may be used to define complex models. The surfaces are stitched together in order to match the tangential properties across shared boundaries. Geometric modeling texts, such as [10], describe interpolation techniques that leverage tangential information while defining spline surfaces. NURBS may be computed such that they exactly interpolate a set of data and produce seamless models.

Some efforts recognize these advantages and focus on defining splines to model input data. Krishnamurthy and Levoy [11] describe an interactive algorithm, allowing a user to paint the boundaries for spline surfaces on the input mesh. They fit the splines to the user-partitioned data with a least square approximation to a grid of re-sampled data. While this method represents a model with spline surfaces, exact interpolation is not guaranteed, and inter-surface continuity is not considered. Additionally, leveraging the inherent nature of our input data, we are able to remove the time intensive requirement of user interactivity by automating data segmentation.

Other techniques also create spline models. Grimm et al. [12] produce manifold surfaces of medical data by leveraging a user produced generator polyhedron. The user input aids in the calculation of multiple spline surfaces that are fit to the original data. This work handles inter-surface continuity by overlapping boundaries. Loop [13] fits a G^1 continuous surface of an irregular mesh by utilizing quad-nets to generate smooth spline surfaces. These methods produce numerous spline surfaces to model the original data. In order to avoid heavy computations with boolean operations to form the solid models, our approach limits the number of spline surfaces used to describe the model.

In a related work, Livingston [14] explores inter-surface continuity when stitching the regions where three spline surfaces meet. His work modifies the location of the corner point to produce smoother results. Our problem prohibits the freedom to move the corner point, requiring that the end model interpolates

the original data set.

We define a method to best approximate the information needed to compute complete spline interpolation [10]. The system creates interpolatory spline surfaces, thus maintaining the original data while creating homogeneous working environments. Additionally, the algorithm described focuses on producing near G^1 inter-surface continuity, smoothly stitching the boundary across adjacent spline surfaces.

IMPLEMENTATION

Molecular models are smooth by nature without ridge lines or creases on their surface. When converting a triangular mesh into a parametric spline model, obtaining this smoothness across adjacent spline surfaces may become a major challenge. It is impossible to specify equivalent cross-boundary tangents, without introducing a singularity on one of the surfaces, at a corner formed by an odd number of spline surfaces. The following section describes the pipeline in Figure 2, explaining our method to stitch adjacent surfaces where this challenge is an issue. The system limits potential G^1 discontinuities to corner regions and employs a novel technique to minimize such ridges.

The system requires the input triangular mesh to have a known connectivity. This assumption allows our methods to focus on achieving inter-surface continuity instead of the tangential challenge of data segmentation. The connectivity of the input data is dictated by the sampling techniques of the molecular visualization system used to generate the molecular meshes that motivate our work.

In particular, the molecular data is sampled and represented as a mesh with connectivity equivalent to the n^{th} subdivision level of an icosahedron [15]. The mesh's triangles are recursively grouped forming parent triangles, until all the triangles are mapped to the original 20 faces of an icosahedron. These 20 triangles are paired forming 10 rectangles, as shown in Figure 2, such that no T-junctions occur on the on the boundary edges and every triangle is paired once and only once. After the data is segmented, the mesh's vertices are marched, extracting the rows and columns for each of the 10 rectangular data grids.

Non-uniform open cubic B-spline surfaces can be computed to interpolate these data grids using complete spline interpolation [10]. Interpolation techniques require that each data point is assigned a parameter value $\{((u_i, v_j), p_{ij})\}$, and that cross-boundary tangents are provided specifying the boundary conditions. In our molecular model, each data point is sampled at a regular interval, making the parameter assignment simply the data points' row and column within the grid. The remainder of this section details a method to compute tangents, assigning the boundary conditions for the data grids, to ensure inter-surface smoothness between adjacent spline surfaces.

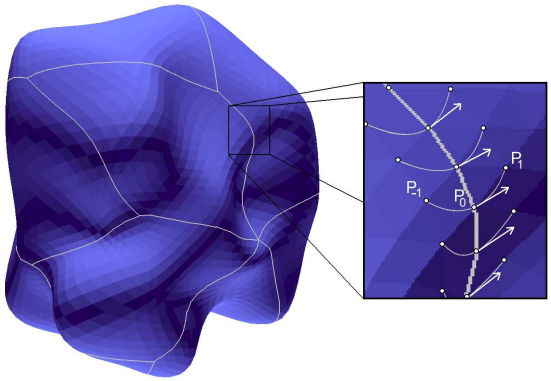

Figure 3. The points used to fit quadratic polynomials and evaluate cross-boundary tangents along shared boundaries.

Cross-Boundary Tangents

Frequently, complete spline interpolation is performed to create isolated spline surfaces. In this case, the boundary conditions are defined by fitting a quadratic curves to the first three data points of each column and row, computing incoming tangents. Similarly, the final three data points are fit to obtain outgoing tangents. Complete spline interpolation returns a spline surface with the computed tangent vectors as its boundary conditions.

In order for two adjacent surface to be continuous, equivalent cross-boundary tangents must be assigned along their shared edge, matching corresponding incoming and outgoing tangents. The system extends the isolated surface approach, for the computation of the cross-boundary tangents. Quadratic polynomials are fit across the boundary, utilizing information equally from the two surfaces. As shown in Figure 3, the tangent is evaluated at the shared point, then this vector is assigned to both surfaces for their corresponding row or column.

The fit considers three points, $\{p_{-1}, p_0, p_1\}$. Point p_{-1} is an interior point of the first data grid, displaced by one location from the shared point. Point p_0 is the shared boundary point at which the tangent is evaluated. Point p_1 belongs to the second data grid and is an interior point also displaced by one location from the p_0. Parameter values, $\{u_{-1}, u_0, u_1\}$, are assigned to the 3 points in the same manner as they had been assigned to the data grids, explained previously. The curve's tangent, $c'(t) = 2a_2t + a_1t$, is evaluated at $t = u_1$, where, $a_2 = \frac{\frac{p_2-p_0}{u_1-u_{-1}} - \frac{p_1-p_0}{u_0-u_{-1}}}{u_1-u_0}$, and $a_1 = \frac{p_1-p_0}{u_0-u_{-1}} - a_2(u_0+u_{-1})$.

It is important to allow each surface equal influence on the cross-boundary tangent, as arbitrary tangents, or tangents evaluated based only on one side of the data grid, may inject undulations on the two surfaces. By fitting the curve across the boundary, the neighborhood of the point, p_0, is considered from

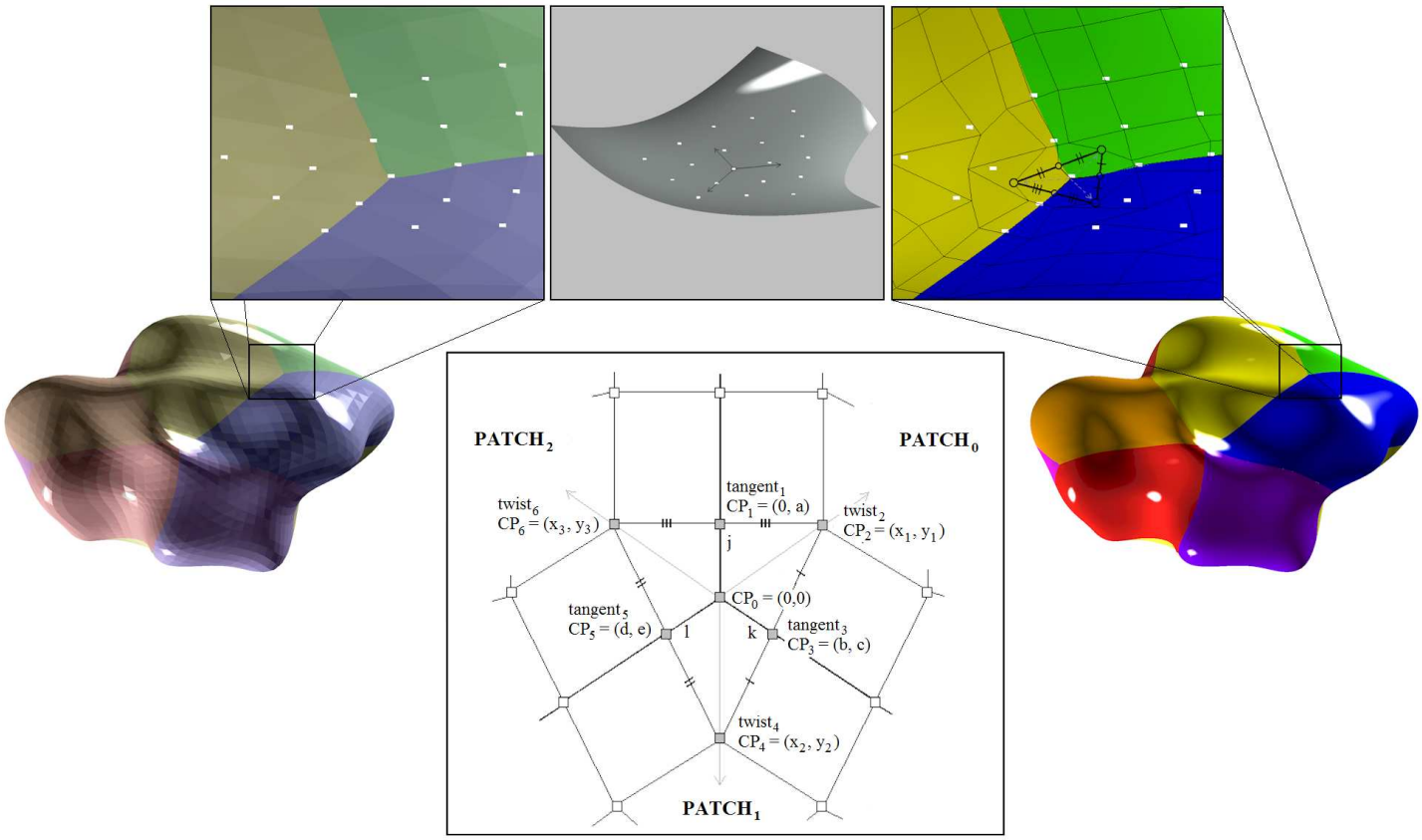

Figure 4. Illustration of the corner tangent and twist computation pipeline. (a) The neighborhood is extracted, (b) weighted least squares fits a cubic surface to the points and tangent vectors are computed on the surface along each boundary, (c) twists are computed by enforcing constraints that create the illustrated regularity about the corner.

either side. Quadratic polynomials require a minimum of 3 data points, thus each involved data grid is given equal influence over the direction of the tangent. Consequently the method produces tangents that minimizes undulations within the resultant surfaces.

Corner Tangents and Twists

Defining cross-boundary tangents in this manner is capable of handling all points along the boundaries, save the corner points. It is impossible to specify the cross-boundary tangents at corners formed by an odd number of spline surfaces such that the boundaries are C^1 without producing a singularity on one of the surfaces [14]. As a result, an algorithm is, at best, capable of achieving a near G^1 continuous stitching around such corners.

The data segmentation technique described earlier produces two corner scenarios victim of this condition. The rectangular grids form corners where either 3 or 5 spline surfaces will meet at a common point. This subsection outlines a technique to compute a configuration of tangents and twists for each spline surface surrounding the corner that minimizes resulting creases, or G^1 discontinuities. The twist is defined as $\frac{\partial^2 \sigma}{\partial u \partial v}$ where u and v are defined along the boundaries for the surface. While the following discussion applies the algorithm only in the context of the 3 surface corner, the technique is scalable to any number of meeting surfaces. In fact, it must be scaled to the 5 surface corner case in order to complete the conversion process.

The system analyzes the properties of a cubic surface, fit to the corner region, in order to estimate tangent and twist values that will smoothly stitch the boundaries. First the neighborhood of the corner is extracted, as shown in Figure 4a. After assigning parameter values and weights to this double ring of neighbor points, a weighted least square cubic surface is fit to the region. The surface's equation is defined within (u,v) parameter space as,

$$\begin{aligned}\sigma = a_{00} + a_{10}u + a_{01}v + a_{20}u^2 + a_{11}uv + a_{02}v^2 \\ + a_{30}u^3 + a_{21}u^2v + a_{12}uv^2 + a_{03}v^3.\end{aligned} \quad (1)$$

The partial derivatives, $\frac{\partial \sigma}{\partial u}$ and $\frac{\partial \sigma}{\partial v}$ evaluated at $\sigma(0,0)$, span the tangent plane located at the origin. The tangent vectors are computed in the direction of each boundary edge by multiplying the parameter values for the corresponding boundary point with the computed partial derivatives.

$$tangent_i(u_i, v_i) = \frac{\partial \sigma}{\partial u} u_i + \frac{\partial \sigma}{\partial v} v_i \tag{2}$$

The system evaluates the control points along each boundary equal to one-third the length of the corresponding tangent value. The variables, *a*,*b*,*c*,*d*, and *e*, are computed corresponding to the control points locations on the tangent plane, as shown in Figure 4. The diagram in Figure 4 illustrates the (u,v) coordinates of each control point on the tangent plane, as well as the variables, *j*,*k*, and *l*, which are scalar values that control the distance along the boundary tangent of the corresponding tangent control point.

After the control points for the tangent values have been computed, the control points responsible for the corner's twists, $\frac{\partial^2 \sigma}{\partial u \partial v}$, are evaluated for each spline surface. Figure 4d depicts the regularity conditions enforced on the twists' locations. The midpoint between each pair of twist control points is the tangent control point for the boundary between them. Additionally, each twist control point lies on the line defined by the tangent vector opposite it. Further explanation of the mathematical benefits of this configuration is explained later. The constraints allow the variables illustrated in Figure 4d to be defined in terms of j as follows:

$$k = \frac{jad}{be - cd}, \tag{3}$$

$$l = \frac{jab}{cd - be}, \tag{4}$$

$$(x_1, y_1) = (\frac{2jabd}{be - cd}, \frac{2jabe}{be - cd}), \tag{5}$$

$$(x_2, y_2) = (0, -2ja), \tag{6}$$

$$(x_3, y_3) = (\frac{2jabd}{cd - be}, \frac{2jacd}{cd - be}), \tag{7}$$

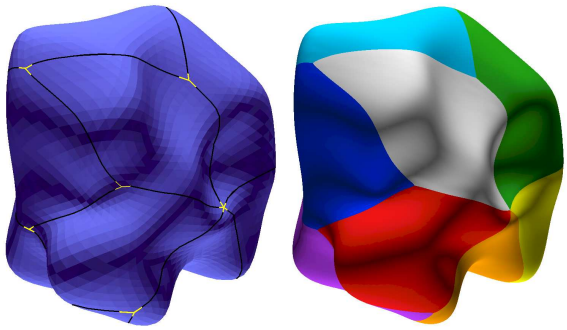

Figure 5. A comparison of the input triangular mesh with the converted smooth spline model. The highlighted regions indicate boundaries with potential non-smooth features.

where, j is set such that $j,k,l <= 1.0$, and $max(j,k,l) = 1.0$.

After computing the desired locations of the co-planar tangent and twist control points around a corner, the system converts these values to tangent and twist vectors for each surface. Referring to Figure 4d, tangent and twist values for each surface are computed as follows:

$$tangent_i = 3(cp_i - cp_0), \tag{8}$$

$$twist_i = 9(cp_{i-1} + cp_{i+1} - cp_i - cp_0). \tag{9}$$

This final portion of the algorithm produces the remaining components required for complete spline interpolation. By compositing each computed surface, the original molecule is reconstructed as a smoother model, as illustrated in Figure 5. The following section quantifies the smoothness results, analyzes the end surface, and provides further understanding of the mathematical details of our corner smoothing technique.

CASE STUDY

The following section analyzes the continuity results of two converted models. The first model is a low curvature molecule, while the second is dominated by areas of high curvature, particularly at the corner regions. The configuration of cross-boundary tangents, by our algorithm, ensures that a majority of the boundaries are G^1 continuous. Only the regions of the first and last knot interval for each boundary are not guaranteed to be smooth.

The graphs of Figure 6 plot the angle differences between normals of shared points along each boundary on their respected models. Two spline surfaces stitch together with G^1 continuity where their boundary plot is equal to 0. The immediate impres-

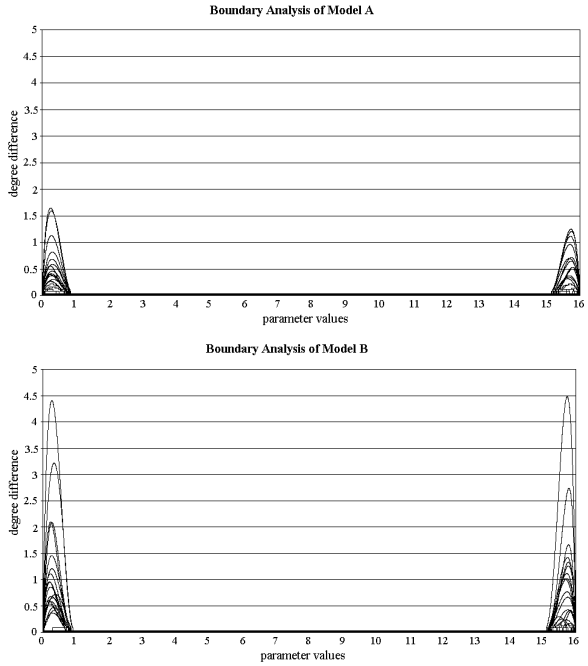

Figure 6. The angles between normals at shared points along each boundary on (a) the lower curvature model and (b) the high curvature model.

sion of the graphs is that the only region of differences, as stated earlier, is within the first and last knot intervals, the highlighted regions from Figure 5. The graphs also indicate that the first model experiences better results than the second model.

The lower curvature molecule, Figure 6(a), endures a worst case angle difference of 1.74°. This same boundary on the original mesh has a 7.6° angle between the triangle normals. The conversion system produces a smoother model, that, in its worst case, has smaller G^1 discontinuities than the original representation. Additionally, approximately 92% of the model's boundary regions are G^1 continuous, and 99% are within 1° of G^1 continuity. Therefore, the ridges created by the stitching process are confined to a very small portion of the overall boundary space, and of those discontinuities, an even smaller portion has a noticeable impact.

The second model with higher curvature experiences similar success, however, with a higher worst case. The 4.48° angle difference corresponds to a 12.4° angle difference between the triangles on the original mesh. The converted spline model is 91% G^1 continuous and 98% within 1° of G^1 continuity along its boundaries. While the worst case is larger than the first model, the conversion records similar percentages of smoothness. The second model converts well around most corners, and only a few boundaries, 14 of 50, as indicated by Figure 6, have differences greater than 1°.

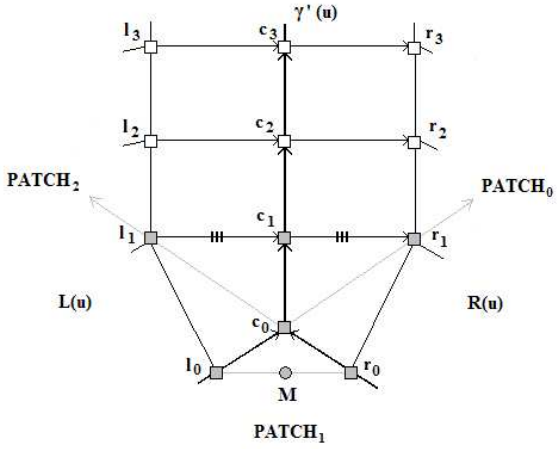

Figure 7. The naming scheme of $L(u)$, $R(u)$, and $\gamma'(u)$, used to further explain the mathematical underpinnings of the corner stitching configuration.

Corner Analysis

The stitching algorithm enforces a degree of regularity about a corner region, minimizing G^1 discontinuities. However, in regions of higher curvature, practice shows that the size of a ridge will grow. The following section examines the mathematical underpinnings that explain this phenomenon, as well as motivate the chosen configuration.

In order to maintain G^1 continuity along a boundary between two surfaces, then $\gamma'(u) \times L(u) = \alpha(\gamma'(u) \times R(u))$. Better written as,

$$\gamma'(u) \times (L(u) - R(u)) = 0. \tag{10}$$

As illustrated in Figure 7, $\gamma'(u)$ is the curve of first derivatives along the boundary; $L(u)$ is the curve of cross-boundary tangents for the left surface sharing the boundary; and similarly, $R(u)$ is the curve of cross-boundary tangents for the right surface. Figure 7 also depicts the variable names used for the associated points. Further defining $\gamma'(u)$, $L(u)$, and $R(u)$, for the first knot interval of the boundary, gives:

$$\gamma'(u) = (c_1 - c_0)\beta_0(u) + (c_2 - c_1)\beta_1(u) + (c_3 - c_2)\beta_2(u), \tag{11}$$

$$
\begin{aligned}
L(u) - R(u) &= [(l_3 - c_3) - (c_3 - r_3)]\Theta_3(u) \\
&\quad + [(l_2 - c_2) - (c_2 - r_2)]\Theta_2(u) \\
&\quad + [(l_1 - c_1) - (c_1 - r_1)]\Theta_1(u) \\
&\quad + [(l_0 - c_0) - (c_0 - r_0)]\Theta_0(u) \\
&= 2(\tfrac{l_0 + r_0}{2} - c_0)\Theta_0(u) \\
&= 2(M - c_0)\Theta_0(u).
\end{aligned} \tag{12}
$$

By substituting back into the original equation,

$$
\begin{aligned}
\Theta_0(u)[\quad & (M - c_0) \times (c_1 - c_0)\beta_0(u) \\
& + (M - c_0) \times (c_2 - c_1)\beta_1(u) \\
& + (M - c_0) \times (c_3 - c_2)\beta_2(u)] = 0.
\end{aligned} \tag{13}
$$

In short, this function indicates that in order to produce a G^1 continuous stitching of the boundary, the vector $(M - c_0)$ must align with $(c_1 - c_0)$, $(c_2 - c_1)$, and $(c_3 - c_2)$. When the vectors of $\gamma'(u)$ are not co-linear, as in areas of high curvature, it is impossible, without modifying the data points, to produce a G^1 continuous surface across the boundary.

Our approach, instead, minimizes the error by guaranteeing that $(M - c_0)$ aligns with $(c_1 - c_0)$. The basis function, $\beta_0(u)$, is the largest contributor to the error in G^1 continuity over the first knot interval; therefore, eliminating this term will yield good results. Because $\beta_1(u)$ also has a strong influence over the region, not all error will be removed, consequently producing small ridges as seen in the results.

Different tangent configurations around a corner yield different locations for the M point. While the M point is guaranteed to eliminate the first error term for all boundaries emanating from the corner, some locations will produce smaller errors from the second and third terms. Multiple methods to compute different tangent configurations were implemented, and analyzed. In practice no one solution guaranteed better performance over another; however, using the tangent plane of a cubic surface fit to the corner with a static parameterization most often experienced better results than other approaches. Using this approach, we recorded the aforementioned results.

CONCLUSION

The system accurately converts a molecular mesh into a smooth spline model. Leveraging the newly realized homogeneous environment, modeling software toolkits can combine the protein strand's spline model with its associated converted molecular spline surface via boolean operations, Figure 8. After obtaining a solid model of the molecule, our fabrication process is able to produce the desired physical visualizations, as shown in Figure 1.

Two main advantages of our algorithm are accuracy and smoothness. The converted spline models maintain the original data using point interpolation, thus producing the level of accuracy we desire. Additionally, our stitching methods enforce inter-surface continuity between the 10 bi-cubic spline surfaces used to encompass the input mesh. 99% of the boundary space is within 1° of G^1 continuity, and regions of potential ridges are confined to the corners of the surfaces, as shown in Figure 5. In the worst case, the angle between the normals of two surfaces at a common point is smaller on the new spline model than the corresponding boundary point on the original mesh. Thus, the new spline model is an accurate representation of the original molecular mesh with added smoothness.

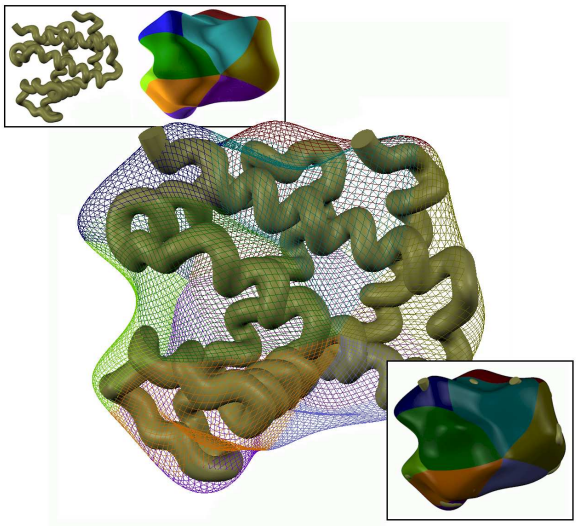

Figure 8. Within the homogeneous environment, the protein spline model is combined with the converted molecular spline model to produce the solid model needed for fabrication.

FUTURE WORK

Our conversion system ignores the data segmentation problem in order to tackle the challenges of smoothly stitching multiple spline surfaces. Because our work is motivated by a specific input data structure, we are able to make assumptions concerning the connectivity of the triangular mesh. Data segmentation of arbitrary meshes remains an interesting problem, that, when solved, may be coupled with our algorithms to the convert any triangular mesh into a smooth spline model.

ACKNOWLEDGMENTS

This work was supported in part by NSF(EIA0121533) and NIH(573996). All opinions, findings, conclusions or recommendations expressed in this document are those of the author and do not necessarily reflect the views of the sponsoring agencies. Our thanks to the researchers of the molecular graphics laboratory at

The Scripps Research Institute, our collaborators on the physical visualization project, for the molecular data and their insights on this paper.

REFERENCES

[1] Sanner, M., 1999. "Python: A programming language for software integration and development". *J. Mol. Graphics Mod.*, **17**, February, pp. 57–61.

[2] M. Sanner, B. Duncan, C. C., and Olson, A., 1999. "Integrating computation and visualization for biomolecular analysis: An example using python and avs.". *Proceedings Pacific Symposium in Biocomputing*, pp. 401–412.

[3] M. Sanner, D. S., and Olson, A., 2002. "Viper a visual programming environment for python". *10th International Python Conference*, February.

[4] Catmull, E., and Clark, J., 1978. "Recursively generated b-spline surfaces on arbitrary topological meshes". *Computer-Aided Design*, **10**(6), November, pp. 350–355.

[5] Doo, D., and Sabin, M., 1978. "Behavior of recursive division surfaces near extraordinary points". *Computer Aided-Design*.

[6] Loop, C., 1987. "Smooth subdivision surfaces based on triangles". *Master's Thesis, University of Utah, Department of Mathematics*.

[7] Xie, H., Wang, J., Hua, J., Qin, H., and Kaufman, A., 2003. "Piecewise c^1 continuous surface reconstruction of noisy point clouds via local implicit quadric regression". In IEEE Visualization '03 Conference Proceedings, pp. 91–98.

[8] Cheng, K., Wang, W., Qin, H., Wong, K., Yang, H., and Liu, Y., 2004. "Fitting subdivision surfaces to unorganized point data using sdm". In 12th Pacific Conference on Computer Graphics and Applications '04 Proceedings, pp. 16–24.

[9] Hoppe, H., DeRose, T., Duchamp, T., and Halstead, M., 1994. "Piecewise smooth surface reconstruction". *Proceeding of the 21st Annual Conference on Computer Graphics*.

[10] Cohen, E., Riesenfeld, R., and Elber, G., 2001. *Geometric Modeling with Splines: An Introduction*. AK Peters, Ltd.

[11] Krishnamurthy, and Levoy, M., 1996. "Fitting smooth surfaces to dense polygon meshes". In ACM Transaction on Computer Graphics (SIGGRAPH 1996 Proceedings).

[12] Grimm, C., Crisco, J., and Laidlaw, D., 2002. "Fitting manifold surface to 3d point clouds". *Journal of BioMechanical Engineering*, February.

[13] Loop, C., 1994. "Smooth spline surfaces over irregular meshes". In Proceedings of the 21st annual conference on Computer graphics and interactive techniques (SIGGRAPH 1994), pp. 303–310.

[14] Livingston, J. B., 1990. "Intersurface continuity of solid models". *Master's Thesis, University of Utah, Department of Computer Science*.

[15] Duncan, B., and Olson, A., 1993. "Approximation and characterization of molecular surfaces". *Biopolymers*.