Computer-Aided Design and Applications

http://www.cadanda.com

Localized Editing of Catmull-Rom Splines

Manolya Eyiyurekli[1] and David Breen[2]

[1]Drexel University, me52@drexel.edu
[2]Drexel University, david@cs.drexel.edu

ABSTRACT

Catmull-Rom (C-R) splines offer many useful modeling properties, such as affine invariance, global smoothness, and local control. They are therefore of great interest to Computer Aided Design (CAD) users. C-R splines are easily evaluated and are a good choice for interactive applications because they interpolate their control points and therefore provide an intuitive way to represent and edit curves in these applications. We have developed an interactive free-form surface-editing framework that uses Catmull-Rom splines for sketch-based editing. Mouse strokes are translated into splines that are used to define free-form surface deformations. Direct control point manipulation is provided for the modification of the splines. Unfortunately, the definition of C-R splines dictates that moving one control point only affects a small, fixed portion of the curve. In order to provide greater flexibility, control and expressiveness, we have developed techniques that expand and generalize the result of modifying one C-R control point. The techniques allow the user to define the range and type of influence that manipulation of a single control point may produce on a C-R curve, providing a versatile and powerful localized curve editing capability.

Keywords: interactive curve editing, Catmull-Rom splines, localized curve editing
DOI: 10.3722/cadaps.2009.xxx-yyy

1. INTRODUCTION

There is a large body of research on curves from the 1960s to the present time [3,7]. Curves are widely used in every aspect of computer graphics, especially splines which are piecewise polynomial parametric curves. Splines are popular in CAD because of the simplicity of their construction, their ease and accuracy of evaluation, and their capacity to define complex shapes during interactive design. Catmull-Rom (C-R) splines [2] are a family of cubic interpolating curves formulated such that the tangent at each control point is calculated using the previous and next control points on the spline. Catmull-Rom splines have C^1 continuity, local control, and interpolate their control points, but do not lie within the convex hull of their control points. We utilize this type of spline in an interactive surface modeler because of its ability to interpolate every control point. Direct control point manipulation has been recognized as a powerful computer graphics tool, and yet the strictly defined local influence of C-R control points, while useful from many perspectives, is limiting and may be a drawback for many freeform editing applications. In a single control point manipulation operation, one is unable to apply a modification to the spline shape that affects more than a small neighborhood on the spline, the span between neighboring control points.

We have recently developed an interactive free-form surface-editing framework [6] that uses Catmull-Rom splines for sketch-based editing. Mouse strokes are translated into splines that are used to define

free-form surface deformations. We employ Catmull-Rom splines in this setting in order to provide an interactive and easy-to-use method for curve editing. C-R spline's ability to interpolate control points is an important feature, one that allows us to accurately translate user input into a mathematical representation. In order to overcome the immediate neighborhood limitation when modifying Catmull-Rom splines, we describe an approach for the localized, interactive editing of C-R splines. To provide greater flexibility, control and expressiveness, we have developed techniques that expand and generalize the result of modifying one C-R control point. The techniques allow the user to define the range and type of influence that manipulation of a single control point may produce on a C-R curve, thus creating a versatile and powerful localized curve editing capability.

Localized editing gives the user more control over the shape of the spline when moving a single control point. An active window is defined around the control point selected by the user, and it limits the resulting modifications to a user-defined segment of the curve. The extent of the window can be changed by the user any time during editing. We provide a variable-resolution editing framework that lets the user specify the control point resolution within the active window. This enables the user to increase the density of the control points where more detail is needed, and makes that part of the curve more controllable. However, due to the locality of Catmull-Rom splines, any change to a single control point only changes the portion of the curve between the control point and its neighboring control points. The dense sampling of the control points provides the means for creating fine details, but it also makes it difficult to specify changes at varying spatial scales. The active window therefore gives the user control over the range of influence associated with a single control point editing stroke. The movement of this single control point is distributed to every control point within the window. The main issue to be addressed is how to transfer the modifications made by the user when dragging a single control point to the other control points within the active window. Our work proposes techniques for distributing the motion of a single C-R control point within an active window in order to produce an intuitive and expressive localized spline editing functionality. Several techniques are described and their results are compared in Section 3. A more detailed example is presented in Section 4.

2. PREVIOUS WORK

Catmull-Rom (C-R) splines [2] offer many useful modeling properties, such as affine invariance, global smoothness, and local control. They are therefore of great interest to Computer Aided Design (CAD) users. C-R splines are easily evaluated and are a good choice for interactive applications because they interpolate their control points and therefore provide an intuitive way to represent and edit curves in these applications. In general it is more natural for points drawn by a user to end up on the curve, rather than defining control points that lie away from the actual curve. The related curve editing work in this section that can be placed in two categories, multi-resolution curves and sketch-based modeling.

Finkelstein and Salesin [8] present the theory and methods for multi-resolution curves, which are detail-preserving, end-point-interpolating cubic B-splines that may be modified at different spatial resolutions. They describe a robust mathematical foundation based on wavelets that supports smoothing, editing and approximating these splines. Although this type of curve has desirable properties, e.g. multi-resolution support, it is not easy to apply to an interactive application where a novice user would need to provide a set of parameters to create the complex framework. Elber and Gotsman [4] extend this approach to non-uniform B-splines and provide the mechanism for local refinement and adaptive local curve manipulation. However, this method uses least-squares approximation for multi-resolution decomposition of the free-form curve and is incapable of providing continuous resolution control. Later work [5] presents a scheme that combines multi-resolution control with linear constraints into one framework, allowing one to perform multi-resolution manipulation of non-uniform B-spline curves, while specifying and satisfying various linear constraints on the curves. The multi-resolution control combined with linear constraints prescribes a precise freeform geometry, and creates a framework for interactive editing of non-uniform B-spline curves.

FiberMesh [9] is a sketch-based modeling framework that uses a set of 3D curves to define surfaces. For a given set of curves, the system automatically constructs a smooth surface by applying functional optimization. These curves also serve as handles to deform the model during the editing process. The

user interface for curve deformation is a direct manipulation method, the user grabs and drags a point on a curve, and the curve deforms smoothly within a region of influence. The geometry of the curve is represented using differential coordinates, and the final result is obtained by solving a sequence of linear least-squares problems, which satisfy the positional constraints given by the user. ShapeShop [10] is another sketch-based modeling tool that describes a new technique for inflating 2D contours into rounded three-dimensional implicit volumes. The system fits a smooth 2D variational implicit curve to the discrete samples collected from user input. Variational curves provide many benefits, such as automatic smoothing and gap-closing with minimal curvature. ShapeShop supports sketch-based editing of the set of point samples, but not the final variational curve. The user modifies these point samples from the current 2D sketch and the variational curve is re-computed using the new samples.

Wires [13] is a deformation technique which uses curves (wires) that are placed in close proximity to the surface and act as handles that deform the surface locally. Wires are defined as free-form parametric curves with a set of parameters and an implicit function that defines a volume around the curve. The curves can be modified with a direct manipulation method, and the related parameters can be set by the user to specify deformations in the underlying implicit surface. Physically inspired methods, such as PriMo [1], provide a precise way to simulate physically plausible deformation, but are not suitable for interactive modeling due to undesirable changes when stretching and compressing the overall curves, e.g. buckling and detail loss due to the length preservation inherent to all physically inspired curve deformation algorithms.

Compared to previous work, our approach provides a novel technique for direct manipulation of an interpolating spline that allows a user to easily modify a curve with an adjustable span of influence; thus overcoming the locality restriction of Catmull-Rom splines. The relative simplicity of the method ensures that the modification will occur at interactive rates.

3. CURVE EDITING

The curves in our editing system are drawn by the user on a reference plane via mouse strokes; thus making them planar curves. As the user clicks and drags the cursor, several control points are captured that follow the cursor's movements. The sampling of the control points is directly related to the speed of the user's strokes. In other words, a slower more detailed input creates a denser sampling and a more accurate representation of the curve while a faster and free-handed drawing leads to a less detailed curve. A Catmull-Rom spline is fit to these control points once the mouse button is released. The first and last control points are inserted multiple times to force the curve to interpolate both end points. A discrete sampling of the points on the curve is displayed to the user as a polyline. The sampling rate used to display the curve is user-controlled and can be changed during the editing session. The control points and the boundary of the active window are highlighted while drawing the curve to aid interactive editing. The curve can be further modified by the user through our localized editing interface. The editing interface provides the user with several options for modifying Catmull-Rom splines. The user can set an active window size to ensure only a certain part of the curve is modified with each stroke. This window can also be interactively changed to fit the user's needs. Once the curve is drawn and a window size is set, editing is achieved by clicking on a control point and dragging it to a new position. The active window need not be symmetric. The user can shift either end of the window to change its extent. All the control points within the active window around the dragged control point are moved following one of the user-defined schemes described in Sections 3.3 and 3.4. The curve is fit to the new set of control points after every editing step, and the user is provided with immediate feedback of the overall shape of the curve while editing. The number of control points within the active window can also be changed by the user in order to provide more detailed, higher resolution editing. The number of control points may also be reduced within the window as well.

The movement of the control points within the active window can be described through a set of schemes that modify all the other control points within the window as a function of the displacement of the selected control point. Two alternate ideas, interpolating the displacement of the selected control point within the active window, and interpolating the displacement vector with a vector orthogonal to the curve, are described in the following sections.

3.1 Active Window

Defining an active window gives the user control over the range of control points that are to be affected by a single editing stroke. The movement of a single control point is distributed to every control point within the window. The window size cannot be larger than the curve itself. The active window is set to be the start and end points of the curve in case the window size is greater than the curve. The window is centered at the control point being dragged, but the extent need not be symmetric. The window size is defined as a number of control points n_R to the right and m_L to the left of the selected/modified control point. For example, the window can span 3 control points on the left and 4 control points on the right. This can either occur if the window size is set to four, but a control point that is 3 control points from a curve boundary is selected, or if the user adjusts the left or right end of the window interactively. At any time the boundary of the active window is highlighted and the user can adjust the window size on either side by moving these highlighted points. Figure 1 demonstrates editing with a symmetric active window, as well as the results from changing the window size.

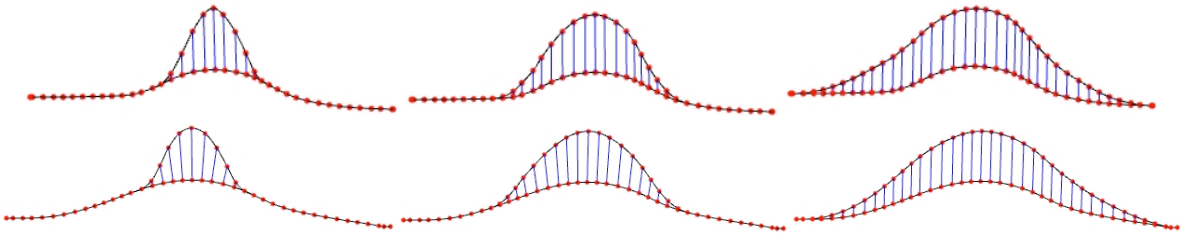

Fig. 1: Changing the active window size. Top: Only the displacement is interpolated. Bottom: Both the displacement and the normals are interpolated. Left-to-right: Editing with window size = 5 control points. Middle: Editing with window size = 10. Right: Active window spans the whole curve. The blue arrows show the movement of the control points within the active window.

3.2 Multi-resolution Control

The resolution of the control points within the active window is user-defined. The control points derived from user input are non-uniform and their sampling resolution is directly proportional to the speed of the input strokes. We found it useful to work with a uniform sampling of control points in most cases, so we have incorporated an option to redefine the curve with a new set of control points that are evenly distributed over the curve. Once the initial control points are inputted, a C-R spline is fit to these points. If the option to uniformly sampling the control points is chosen, a new set of control points is created by evenly sampling the curve, and a new curve is fit to these control points. The number of control points is kept the same during this operation. However, the user may also increase/decrease the number of control points immediately after the curve is resampled.

In cases where a part of the curve is stretched or condensed, the resolution on the modified portion of the curve may need to be adjusted. The user can either chose to resample the curve within the active window with the same resolution as the rest of the curve or increase/decrease the resolution of the control points within the active window. This resampling is also uniform but the resolution can be higher or lower than the rest of the curve. Once the active window is placed over another portion of the curve, the resolution in the previous active window stays fixed until it is once again edited by the user. This kind of variable-resolution control combined with a flexible, i.e. asymmetric, active window enables us to define very specific parts of the curve to be edited. Rough sketches can be created by lowering the resolution and working with a larger active window. Then more control points can be added where needed to provide additional detail and control.

3.3 Interpolating the Control Point Displacement

The first approach taken to solving the problem of distributing control point displacement within the active window sets the direction of the displacement to be the same for all control points within the window. The magnitude of the displacement monotonically decreases from the control point being

modified to the boundaries of the window. There are two properties needed for the function that implements this kind of drop off in order to create an acceptable result. The function should be smooth and it should go to zero at the boundaries to avoid discontinuities. We have tested several such functions and found three that provide the desired smooth transitions (Equations 1, 2 & 3). These equations include a linear, Gaussian and a cosine function that all decrease from 1 to 0 within the window range. Equation (2) uses a second function (f(d)) to ensure proper boundary conditions, since the Gaussian does not go to zero in a finite range.

Linear: $$1-\frac{d}{W} \tag{1}$$

Gaussian: $$f(W-d)e^{\frac{-d^2}{2\sigma^2}} \tag{2}$$

$$\sigma = W/5$$

$$f(x)=\begin{cases} 1.0 & x > \varepsilon \\ (x/\varepsilon)^2 & x \le \varepsilon \end{cases}$$

Sinusoidal: $$\frac{1}{2}+\frac{1}{2}\cos^{\alpha}(\frac{d}{W}\pi) \tag{3}$$

d is the distance (over the curve) to the center of the window, i.e. the control point being modified by the user, and W is a window size defined for each side of the active window, either W_R or W_L.

f is a function that ensures the whole equation goes to zero smoothly. It is equal to 1 up to a small distance from the boundary of the window (ε=σ in our examples) and goes to zero smoothly at the boundary.

Figures 2 and 3 present curves produced from interpolating the control point displacement and using the three drop-off functions. Figure 2 applies the change to the entire curve and Figure 3 uses a symmetric active window of size 5. Equation 3 created the best results in terms of a smooth and intuitive interaction and we use this function in the final examples presented in Section 4. Figure 4 shows a case where the curve is edited twice, once pulled towards the left and once towards the right. Although the results are good, moving all control points in the same direction is not always satisfactory for our purposes. Section 3.4 describes another scheme for calculating the offset direction for all control points in the active window.

Fig. 2: Distributing the displacement of the control point on the entire curve. Left to right: Linearly decreasing function in Equation 1, exponentially decreasing function in Equation 2, sinusoidal function in Equation 3 (α=1.0). Three curves are drawn in each case: Initial curve, local effect of moving one control point, and the curve after distributing the movement to all control points.

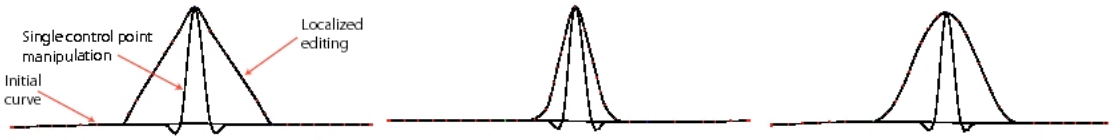

Fig. 3: Distributing the displacement of the control point within the active window. Window size=5. Left to right: Linearly decreasing function in Equation 1, exponentially decreasing function in Equation 2, sinusoidal function in Equation 3 (α=1.0). Three curves are drawn in each case: Initial curve, local effect of moving one control point, and the curve after distributing the movement to all control points.

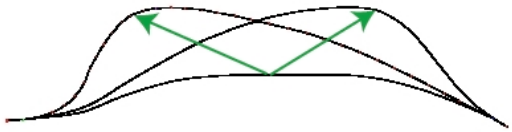

Fig. 4: The result of distributing the displacement of the control point. The same curve is modified twice by pulling the same control point in two different directions. Green arrows show displacement of one control point during editing.

The sinusoidal interpolant may be further adjusted to produce a variety of results. The parameter α is the exponent of the cosine in Equation 3. Setting the parameter to something other than 1 changes the shape of the curve in the active window. As seen in Figure 5, a value of α less than 1 begins to square off the modified section of the curve. Increasing α's value above 1 creates a faster drop-off in the displacement and a sharper hump in the curve.

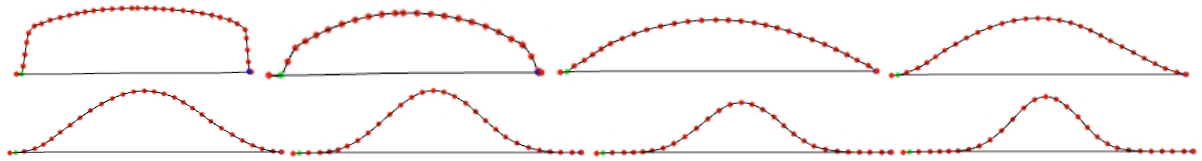

Fig. 5: The result of changing α in Equation 3. Left to right: α=0.1, 0.25, 0.5, 0.75, 1.0, 2.0, 3.0, 4.0.

3.4 Interpolating the Curve Normal

We further investigated distributing the direction of the control point movement within the active window, and tested two other solutions that led to another approach to localized editing of Catmull-Rom splines. One way to move the control points is to let the points move in the direction normal to the curve. Although this appears to be a natural way to move the control points, it did not produce acceptable results. This approach leads to unwanted deformations because it gradually expands/shrinks the portion of the curve within the window. The second approach interpolates the selected control point displacement with the curve's normal at the endpoints of the active window, while decreasing the magnitude of the offset using either Equation 1, 2 or 3. We use spherical linear interpolation (slerp) [11] to calculate intermediate vectors between the displacement of the selected control point and the curve normals at the boundaries of the window. All vectors are initially unit vectors that are then scaled according to functions described in Section 3.3 to create a set of displacement vectors for control points in the active window.

$$|D_0| * S(d) * slerp\left(\hat{D}_0, \hat{N}_n, d\right) \tag{4}$$

D_0 is the displacement of the selected control point. N_n is the curve normal at one of the boundaries of the active window, where n is the window size. The normals at n_R or n_L are used depending on which side of the window is being processed.

Both of these vectors are normalized before being interpolated. d is the distance along the curve to the center of the window. S(d) is one of the scaling functions in Equations 1 to 3. This approach created the best results in terms of smoothness and fairness of the resulting curve. Figures 6 and 7 contain results from this method using the three drop-off functions and two active window sizes.

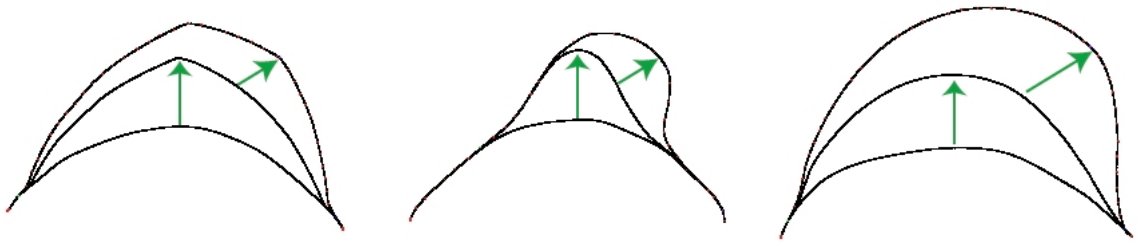

Fig. 6: Interpolating control point displacement with the curve normal at curve end points using spherical linear interpolation. The green arrow represents the editing stroke. Left to right: Linearly decreasing function in Equation 1, exponentially decreasing function in Equation 2, sinusoidal function in Equation3. The active window spans the whole curve.

Fig. 7: Interpolating control point displacement with the curve normal at the window boundary using spherical linear interpolation. Window size=5. The green arrow represents the editing stroke. Left to right: Linearly decreasing function in Equation 1, exponentially decreasing function in Equation 2, sinusoidal function in Equation 3.

3.4 Editing Tangent Vectors At Control Points

We also explored the affects of editing tangent vectors at control points as an additional means for manipulating the curves. The tangents are initially calculated by averaging control points as explained in [2] for C-R curves. The user then can put the system in tangent editing mode and only the control points with their tangent vectors are displayed, instead of the curve itself. Clicking and pulling on the displayed lines representing the tangents can manipulate these vectors. Both the direction and size of the tangents can be changed. Figure 8 shows an example of changing a C-R curve's tangents to edit its shape.

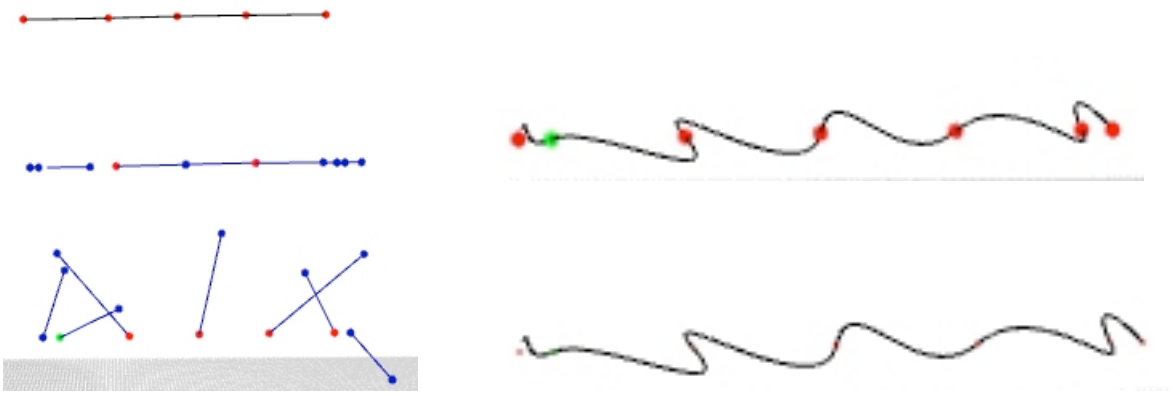

Fig. 8: Editing tangent vectors at control points. The control points are highlighted in red and the tangent vectors are drawn as blue lines. The vectors are drawn at each control point and the blue points at the end of each vector can be picked and modified resulting in new tangent vectors. Left: Top to bottom: The original C-R curve, original tangents, modified tangents. Right: The final curve after modifying the tangents, drawn with and without the control points highlighted.

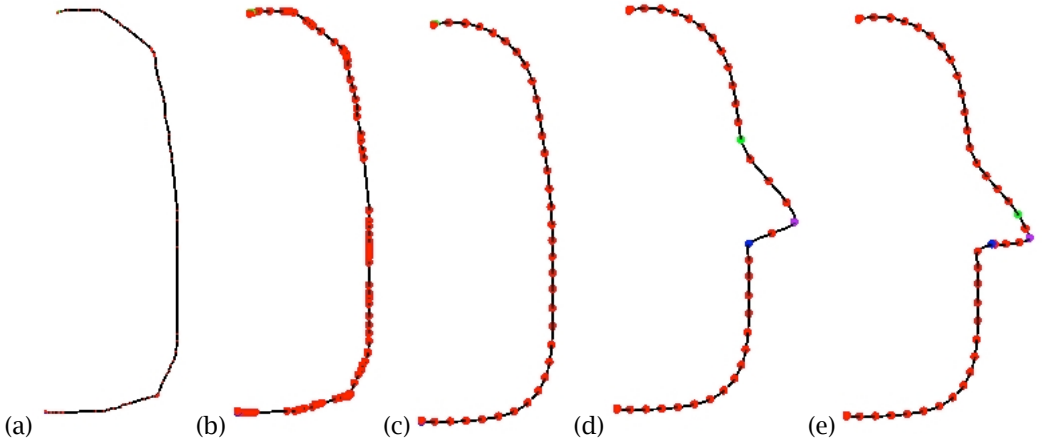

Fig. 9: The rough, non-uniform user input (a-b) is sampled uniformly and smoothed (c). Pulling the point highlighted in purple outwards creates the nose (d). The resolution in the active window is increased for further editing. The tip of the nose is pulled down to create the nose in (e). The active window boundaries are highlighted in green and blue. The editing is applied to the point highlighted in purple.

4. RESULTS

Here we demonstrate the variable-resolution, localized editing capabilities of the curve-editing approach. Figures 9-11 shows intermediate steps from an editing session that defines a facial profile. Figure 10-f presents the final result of the session. The initial user input (Figures 9a-b) is neither smooth nor uniform. First a uniform sampling of the control points at a user-defined resolution is produced (Figure 9c). Pulling the point highlighted in purple in Figure 9d outwards creates the nose. This operation stretches the area around the tip of the nose. The user chooses to resample this part of the curve to increase the resolution for further editing. Figures 10a-e show how variable-resolution

control and localized editing is utilized to create the mouth and chin. In order to create a rounder nose tip the user limits the active window to a small part of the curve around the nose and increases the resolution (Figure 11a). Figure 11b and 11c shows the editing of the nose. We added 3 more curves to roughly define the eye. Similar techniques are used to create and modify these additional curves.

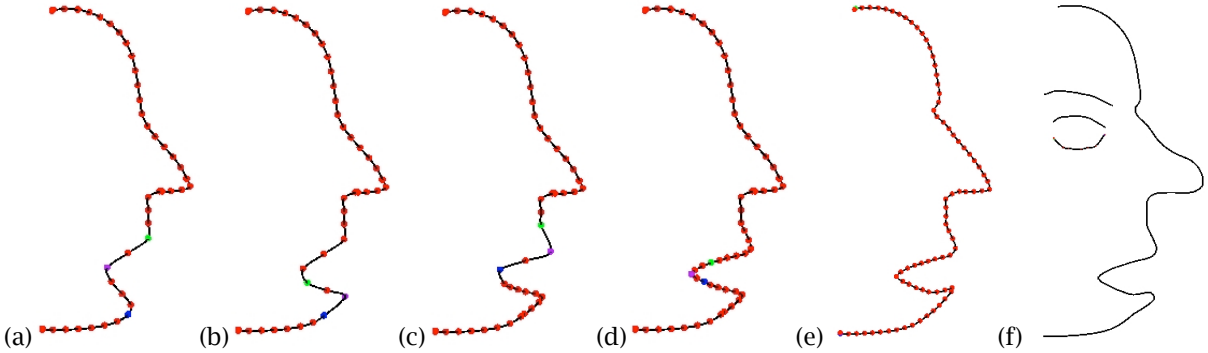

Fig. 10: The user works at different resolutions to create the mouth. The active window boundaries are highlighted in green and blue. The editing is applied to the point highlighted in purple. The entire curve is resampled in (e). The final result from sketching a profile is shown in (f.

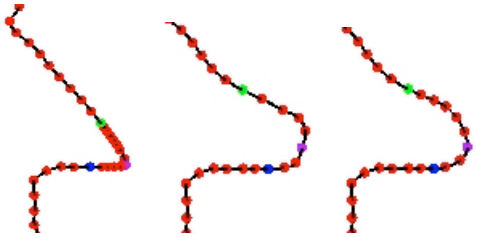

Fig. 11: The user works at different resolutions to edit the nose. The active window boundaries are highlighted in green and blue. The editing is applied to the point highlighted in purple.

We utilize our curve-editing tool in an interactive level-set surface-editing framework [6]. The surfaces are deformed using curves as outlines to the final shapes. We have included an example from this system in Figure 12. A sphere and a cross sectional curve is used to create a rubber duck.

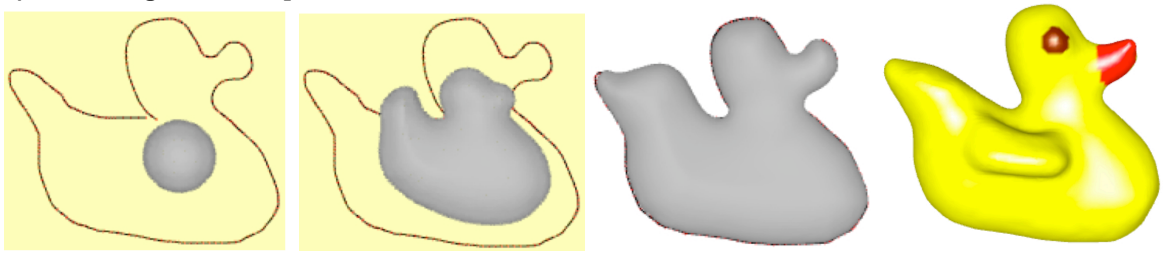

Fig. 12: A C-R curve is drawn and edited to look like a rubber duck. A level-set sphere model is evolved to fit this curve using sketch-based editing techniques. The final image is further edited using free-form editing operators [6].

5. CONCLUSIONS

We describe an approach for localized editing of Catmull-Rom (C-R) splines. We prefer to use C-R splines for their ability to interpolate every control point, which is important for accurately translating user input into a mathematical representation in our sketch-based surface editing system. Localized editing gives the user more control over the scale of editing to be performed, and the range of influence of a single editing operation. An active window is placed around the selected control point to limit the modifications to a sub-region of the curve. The user can change the size of the window and the control point resolution within the window any time during editing. The offset of the control points within the active window can be described through a set of schemes that interpolates the displacement of a selected control point. We discuss two alternative ideas, interpolating the displacement directly, and interpolating the displacement vector with the normal to the curve at the boundaries of the active window. These schemes provide a versatile, expressive and powerful localized curve editing capability for Catmull-Rom splines.

6. REFERENCES

[1] Botsch M.; Pauly M.; Gross M.; Kobbelt L.: PriMo: coupled prisms for intuitive surface modeling, Proc. 4th Eurographics Symposium on Geometry Processing, 2006, 11-20.

[2] Catmull, E.; Rom, R.: A class of local interpolating splines, In Computer Aided Geometric Design, R. E. Barnhill and R. F. Reisenfeld, Eds. Academic Press, New York, 1974, pp. 317–326.

[3] Cohen, E.; Riesenfeld, R. F.; Elber, G.: Geometric Modeling with Splines, A.K. Peters, 2001.

[4] Elber, G.; Gotsman, C: Multiresolution Control for Nonuniform B-spline Curve Editing, 3rd Pacific Graphics Conference on Computer Graphics and Applications, Seoul, Korea, 1995, 267-278.

[5] Elber, G: Multiresolution curve editing with linear constraints, Symposium on Solid Modeling and Applications, 2001, 109-119.

[6] Eyiyurekli M.; Breen D.: Interactive Free-Form Level-Set Surface-Editing Operators, in preparation.

[7] Farin, G.: Curves and Surfaces for CAGD: A Practical Guide, 5th edition, Morgan-Kaufmann, 2002.

[8] Finkelstein, A; Salesin, D.: Multiresolution Curves, Proc. SIGGRAPH '94, 261.

[9] Nealen, A; Igarashi, T; Sorkine, O; Alexa, M: FiberMesh: Designing Freeform Surfaces with 3D Curves, ACM Transactions on Graphics, ACM SIGGRAPH 2007, San Diego, USA, 2007, 41.

[10] Schmidt, R.; Wyvill, B.; Sousa, M. C.; Jorge, J. A.: ShapeShop: Sketch-Based Solid Modeling with BlobTrees, 2nd Eurographics Workshop on Sketch-Based Interfaces and Modeling, 2002, 53-62.

[11] Shoemake, K.: Animating rotation with quaternion curves, Proc. SIGGRAPH, July 1985, 245-254.

[12] Singh, K; Fiume, E.: Wires: a geometric deformation technique, Proc. SIGGRAPH, July 1998, 405–414.