# EDITING STRATEGIES FOR BÉZIER-MODELED CONTINUOUS EXPRESSION CURVES

*Bret Battey*
De Montfort University
Music, Technology &
Innovation Research Centre

## ABSTRACT

PICACS (Pitch Curve Analysis and Composition System) is a software prototype inspired in part by study of Indian classical music. The primary motivation of the system is to facilitate analysis, composition, and convincing computer rendering of music that is characterized by detailed, expressive shaping of the continuum between scale steps. The system creates models of pitch, amplitude, and spectral-centroid expression envelopes by utilizing nonlinear least-squares fitting of constrained Bézier spline curves. Details of the modeling system have been published elsewhere [2]; this paper focuses on the unique challenges that arise in designing an editing system for such models to support compositional purposes. While some of the challenges are specific to the Bézier model, it is clear that many of the challenges would apply to any system that seeks to support modelling and editing of continuous expression using sets of linked curves and multiple layers of expression data.

## 1. INTRODUCTION

Computer music arose during a period in which some composers were strongly challenging the primacy of the pitch lattice, focusing instead on treating pitch-space as a continuum. In our own time, this is reflected in a current compositional focus on spectromorphology and gesture [1].

However, numerous other musical traditions around the world have also placed great emphasis on the shaping of the pitch continuum. Indian classical music is one of these traditions. Though its pitch organization is based on a scalar framework, the subtle control and specific shaping of the space between scalar steps is often fundamental to the expressive power of performance. Indeed, both the curvilinear shaping of pitch gesture and the accompanying hand and body gestures employed by many North Indian classical vocalists suggests an abstract but potentially powerful affinity between the ancient traditions of Indian musical expression with the gestural language of some contemporary electroacoustic music.

This intuition of commonality has led the author to pursue research into computer modelling and expressive rendering of pitch-continuum melodies such as those found in Indian classical music. It seems possible that such a pursuit could inform the integration of aspects of traditional pitch language with the gestural shaping of found-sound utilized in some contemporary electronic music.

In this pursuit, it has become clear that existing computer music tools are not well aligned with the goal of defining and rendering expressively convincing pitch-continuum melodies. Some tools, such as those that are MIDI-based, rely excessively on a discrete-event paradigm. Others may better support pitch continuums but do not provide means to create continuous curves between specific scale steps utilizing the characteristic shapes produced by vocalists and physical instruments.

Further, the tools often do not readily support the crafting of pitch, amplitude, and spectrum simultaneously. Indian classical music performance can demonstrate a rich complexity of pitch, amplitude, and spectral interrelationships. Given this complexity, analysis and extraction of expression data from human performance seems like a reasonable approach to the problem. Having captured such data, editing tools could be developed that would allow the composer to repurpose the expression data towards the convincing rendering of custom pitch gestures.

## 2. PICACS

PICACS (Pitch Curve Analysis and Composition System) is a software prototype developed to address these issues [2]. Pitch, amplitude, and spectral centroid curves are extracted from a recorded performance. Critical inflection points on these curves are identified utilizing psychoacoustic criteria. Constrained Bézier splines are fit to the data between those points using a nonlinear optimization technique. Details of this Bézier modelling process, the nonlinear fitting techniques required to execute it, and examples of its use for expressive computer rendering have been published in the above-referenced article and are not provided here. PICACS operates under Mac OSX OpenMCL using Bill Schottstaedt's Common LISP Music (CLM) and Rick Taube's Common Music (CM).

There are two primary means by which PICACS can support compositional activity and convincing expressive rendering of pitch-continuous melodies and textures. One approach is to use the PICACS analysis functions to make generalized models of various aspects of continuous expression in recorded performances. One might, for example, collect a wide variety of instances of an Indian singer executing a particular type of ornament. Using the quantitative characterizations of the ornaments that PICACS provides, one could then create a statistically informed model of the ornament and an

editing function that implements this model. The composer could then sketch a pitch-continuous melody and apply the ornament model at a point in that melody, resulting in a fully expressive ornament that reflects the style of the source artist(s).

The second approach is to provide editing tools that allow a composer to repurpose the nuances of human performance(s) towards expression of a new melody or texture by editing and rearranging the PICACS model of that performance.

To date, tools have been developed to support the second approach, allowing for editing and rearranging of the segments of the Bézier model. In the process, it became clear that editing of continuous expression models brings up distinct challenges, some specific to the use of Bézier curves, but others seemingly common to any system that would use a series of connected curves and would have multiple layers of expression data. These challenges also reveal conceptual differences between continuum thinking and discrete-note thinking. It is these challenges that we will explore here.

## 3. OVERVIEW OF BÉZIER MODELING

In order to understand the editing issues and their resolutions, it is first necessary to understand the basic elements of the Bézier expression model.

The analysis process begins by identifying perceptually-critical inflection points in the pitch, amplitude, and fundamental-normalized spectral-centroid curves extracted from the target recording. PICACS then fits constrained Bézier splines between these inflection points.

A Bézier spline is a parametric spline curve. PICACS utilizes 3rd-order Bézier splines, in which two endpoints and two "control points" determine the shape of the curve. Such splines are widely used in computer graphics applications. As shown in the Figure 1, each control point determines the slope of the curve at its respective endpoint. It also determines the "weight" of the endpoint. That is, the further the control point is from its endpoint, the more the curve is weighted towards that endpoint.

However, PICACS utilizes constrained Bézier splines. The constraints were chosen in order to ensure that the curves always move forward in time, to reduce the number of variables required to define a curve, and to ensure that the critical-inflection points to which the curve is fit are the true maximum and minimum of the curve. Among the required constraints is the rule that each control point must lie in time between the two endpoints. This fact conveniently allows us to describe the position of each control point as a ratio of the distance between the endpoints. Figure 2 depicts the constraints and the use of ratios to describe the control point positions.

Fitting the constrained Bézier splines to the curve data between critical inflection points requires nonlinear least-squares fitting techniques. Once fitting is complete, the model of the original curve consists of a series of constrained Bézier splines connected at the critical inflection points. Figure 3 presents an example model.

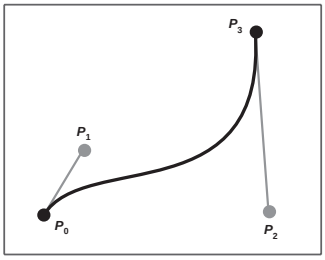


**Figure 1**. Bézier spline defined by two endpoints (P0 and P3) and two control points (P1 and P2).

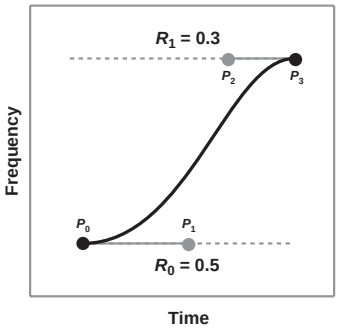


**Figure 2**. Bézier spline constrained such that 1) the control points must be at the same vertical level as the corresponding endpoint and 2) must lie in a position between the two endpoints. The grey and grey-dotted lines indicate the allowable range of positions for the control points. R0 and R1 define the position of the control points as ratios of the space between the endpoints.

Analysis of a whole recording is divided into phrases, defined by borders of silence. The analysis of a phrase is divided into pitch, amplitude, and spectral centroid layers. The set of splines that defines a phrase layer is called a bezlist. Joints between splines are called nodes. Each Bézier spline is called a Bézier segment or bseg for short.

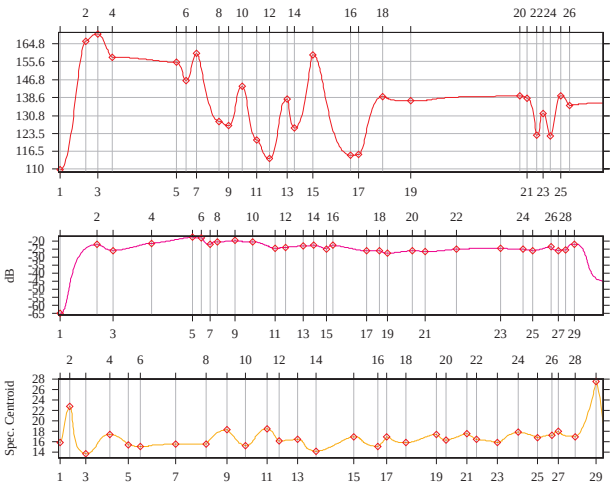


**Figure 3**. Bézier spline model of a Khyal phrase in raag yaman sung by Dr. Vikas Kashalkar.

## 4. FAMILIAR EDITING NEEDS

At one level, editing a PICACS phrase requires editing functions common to event-based musical editing environments in general. We want to execute queries (finding attributes of a given object such as the time or value of a node or the duration of a segment) and edits (changing the properties of objects). We want to be able to make edits to both time and value attributes in all three layers. It is helpful in some cases to have value functions specific to their domain, such as functions for the pitch layer allowing transposition or setting a node to a specific scale degree. The implementation of such functions is not problematic, so it does not need to be elaborated here.

## 5. SPECIALIZED EDITING STRATEGIES

While the basic editing needs are conventional, the specialized nature of continuous expression curves and the Bézier model introduces distinct design challenges and complexities.

### 5.1. Connect, Merge and Split

There are two strategies we can utilize if we wish to replace a number of segments with a single segment.

The first of these is called NodeRange Connect in PICACS. In this case, a range of segments is replaced with a single segment running from the first node in the range to the last. The control point position ratios of the replacement segment are set to a default value such as 0.5 or to user-specified values. In this case, the shape of the replacing segment may bear little relationship to the combined shape of the segments being replaced.

The second strategy is called NodeRange Merge. In this case PICACS attempts to set the control point positions of the replacement segment to provide an approximation of the shape laid out by the replaced segments. This requires PICACS to render the curve defined by the target segments, then run its nonlinear curve-fitting procedure to optimally fit a single segment against that curve. This can effectively simplify a curve between two nodes when the intervening details of the curve are presumed to be unimportant.

Dividing a segment into two parts at some ratio-defined point across its length (the Segment Split operation) involves some particular challenges, as well. First, due to the mathematical nature of the Bézier spline it is not possible to find the value of the spline at a particular point in time (this is explained in greater detail in the aforementioned paper), hence we must estimate the value at the desired split point. Second, once we identify a split point, we have to determine the required control point positions needed to allow the two new splines to match the curve being replaced. At this point, PICACS simply linearly interpolates between the beginning and ending values of the spline to determine what value to use at the split point. This can produce a value that lies considerably off the original curve. An alternative strategy would be to render the curve and then look for the center value by linear interpolation between the nearest rendered points. In any case, the curve is split at the estimated center point, creating two new splines. Nonlinear optimization of the control points of the two new splines is required to establish optimal fit with the original curve.

### 5.2. Maintaining Layer Coordination

Difficult issues also arise when one wishes to alter time attributes, given that the analysis is comprised of three layers of data. A basic premise of the PICACS editor is that all three layers should always remain the same duration; any duration shift in one layer must be reflected in the others.

Single nodes (excluding end nodes of the phrase) can be shifted without impacting the overall duration. But should the change be reflected in other layers? For example, if one moves the time position of a single pitch node, should one alter the amplitude and spectral centroid layers as well? And, if so, how? There is no guarantee that there will be a time-matching node position in the other layers; indeed, there usually will not be. A function that would alter all three layers requires a Segment Split operation on the two complementary layers and a linked shift of the three resulting nodes. This can be problematic given that the context of the node may differ by layer. In one layer, the shift could actually land on or beyond an already existing node, or a segment may be created that is so short as to be either imperceptible or to be illegal for the output rendering algorithms. Such circumstances must be detected and handled appropriately.

In some cases, it makes sense to move a pitch node's time independently of the other layers, so this option is retained.

If one wishes to remove a segment or contiguous segments from a layer (the NodeRange Cut operation), one must remove the time gap from the other two layers. In order to do this, one must apply two Segment Splits to the other two layers at the beginning and end times of the target edit. The same risks mentioned in the above paragraph apply and code must detect and clean up any problems created by the split.

At that point, one is left with the question of how to heal the cuts — that is, how to regain data continuity at the point of the edit (Figure 4).

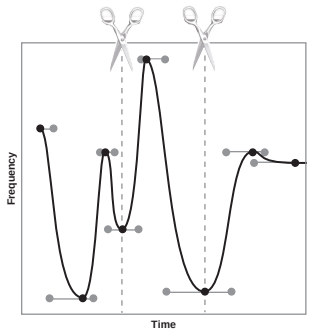


**Figure 4**. Portion of a larger bezlist in which the two segments are targeted for cutting. Gray dots indicate the segment control points.

To heal the discontinuity, we can either offset all segments that follow the cut (Figure 5), alter the endpoint of the spline preceding the cut, or alter the spline following the cut (Figure 6).

PICACS default behaviour is to alter the endpoint of the spline preceding the cut. Shifting up the entire bezlist after the cut is assumed to be a rarely-desired option, since regardless of whether one is operating on the pitch, amplitude, or brightness layer, the change is likely to represent a significant alteration of the perceptual character of the material.

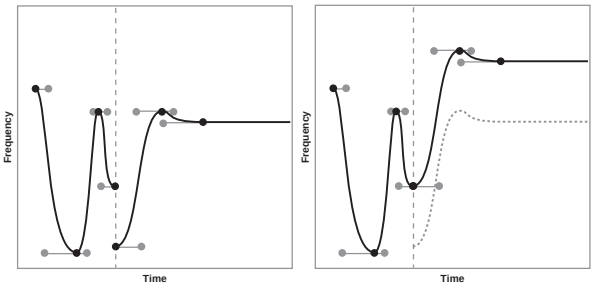


**Figure 5**. Now that the segments have been removed, there is a discontinuity in the bezlist. One response is to offset the segments following the cut to maintain continuity.

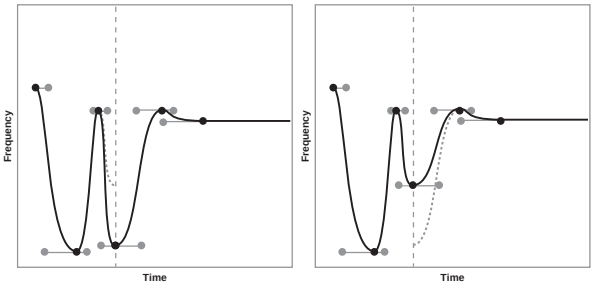


**Figure 6.** Altering the segment preceding a cut to maintain continuity versus altering the segment after a cut.

### 5.3. Other Considerations

In contrast to thinking of atomistic events with their own specific onset time, such as one tends to do in MIDI sequencing or multi-track soundfile editing, in bezlist models one tends to focus more on the time relationship between nodes. Hence it has been the author's experience that functions to set nodes to a specific time are not needed. Instead, one sets the timing of a node in relationship to other nodes. The primary function for doing this is Time Center, where one sets the time of a node to be in a position between the two surrounding nodes, with the position described by means of a ratio.

Given that one is editing a continuous pitch contour, where relative pitch positions may be more important than specific scale step positions, it is helpful to have editing tools that allow one to shape the continuum without reference to scale steps. Such tools include: Relative Scale, in which a group of nodes is scaled by a ratio between one node and some given value; Scale Towards, in which a group of nodes is scaled towards a given node; and Scale-Value-Deltas, in which the differences between subsequent values are scaled (for expanding or contracting the pitch contour).

Further, pitch editing is enhanced by the ability to request that a group of nodes be tuned to the nearest scale step of a given scale, or tuned to the nearest scale step and then randomly detuned.

### 5.4. Style/Ornament Operations

Another editing tool category allows for creation of ornamentation or elaborations of melodic contours. For example, a common pattern in Indian Khyal singing is to ornament a set of pitches by inserting lower-neighbour pitches between them. The lower neighbours are typically lower in amplitude. Further investigations could codify a likely set of spectral centroid variations associated with such ornaments. These observations could be combined to create a tool to apply the pattern. Investigations continue in this regard.

## 6. CONCLUSIONS

Bézier spline models provide a powerful tool for analysis, composition, and rendering of pitch-continuum expression on the computer. At the same time, the creation of editing tools to facilitate their manipulation for musical purposes raises distinct challenges, revealing conceptual differences between continuum thinking and discrete-event thinking in the musical realm.

In 2005, the author used PICACS in the creation of his composition *Autarkeia Aggregatum*. In this case he used fragments of South Indian vocal music as source data, recombining and editing the resultant PICACS model to create custom melodic material. The data was exported to *Praat* [3], enabling sample-based rendering using PSOLA techniques to avoid formant shifting [4].

At this time, PICACS research continues in a command-line environment. Editing of Bézier spline models invites a GUI approach, and development of such an interface is the logical next step in the evolution of PICACS.

## 7. REFERENCES

[1] Wishart, T. 1996. *On Sonic Art*. Harwood Academic Publishers.

[2] Battey, B. 2004. "Bézier Spline Modeling of Pitch-continuous Melodic Expression and Ornamentation". *Computer Music Journal.* 28:4, pp. 25-39, Winter 2004.

[3] Boersma, P. and D. Weenink. 2005. *Praat*. Version 4.3.04. http://www.fon.hum.uva.nl/praat/

[4] Miranda, E.R. 2002. *Computer Sound Design: Synthesis Techniques and Programming*. 2nd edition. London: Focal Press.