MUSIC ANALYSIS AND SPECTRAL MODELING BASED ON CUBIC B-SPLINES

Daniel Möhlmann, Otthein Herzog, Thomas Wagner
Center for Computing Technologies (TZI), Universität Bremen, Germany
dmoehl@tzi.de herzog@tzi.de twagner@tzi.de

ABSTRACT
Structured representations of audio offer interesting opportunities for analysis and manipulation: techniques like spectral modeling help to decompose digital sound recordings into sinusoidal elements, which may then be analyzed or transformed separately. In this paper, we present a novel method to perform high-level decomposition of musical signals to obtain an intuitive and compact structured representation. We present examples of our spline conversion method for both single instrument notes and complex music recordings. Our approach is based on sinusoidal analysis and uses cubic B-splines to encode sound elements in frequency and time. The proposed model preserves the relevant details of the spectral frequency distribution to a large degree and is well suited for additive synthesis.

1. INTRODUCTION
Digital audio can be processed and analyzed in various contexts, including music production and speech transmission. In most situations it is a challenging task to extract information from a mixed recording. What seems easy for a human listener — e.g. listening to voices in a crowded room or recognizing several instruments in an orchestra — is difficult to achieve in computer-based analysis. It is typically even more complicated to make changes to such mixed recordings, which would require methods to separate the sound of various sound sources perfectly in time and frequency.

The ways in which audio can be analyzed and manipulated depends strongly upon the form of representation and encoding. In [8], Vercoe et al. divide audio encoding methods into structured and unstructured representations. While in the unstructured case of a PCM waveform recording individual samples carry little information, structured representations of audio are based on a low-dimensional representation space, in which the individual dimensions control significant perceptual aspects of the signal, like pitch, volume or the temporal structure. An extreme example for structured audio is the MIDI protocol, which represents music as a collection of control signals, such as note onsets and offsets, instrument codes or vibrato intensity. While it is very convenient to manipulate MIDI data, its notation-based model cannot be applied to the analysis and transformation of arbitrary sound signals. There are various other examples for structured sound representations and corresponding synthesis methods, most of which are tailored for a specific purpose. For example, physical modeling describes a specific sound producing system and the interacting forces in it [7]. The main problem of physical models is that there is no easy way to build a model automatically from sound. For frameworks that need to store or manipulate arbitrary sound signals, an all-purpose representation model is needed.

Spectral models provide much of the desired versatility. In section 2, we discuss the standard spectral model and some of its drawbacks, including the disadvantage of redundancy and the lack of expressivity resulting from its frame-based model. To overcome these problems, we propose an extended spectral model based on cubic B-splines, which encodes time-dependent frequency and power information in a simple set of spline coefficients. We describe our model in section 3. In section 4, we show how a PCM waveform can be converted into a spline-based representation that preserves the perceptually relevant aspects of the sound. Methods for analysis and re-synthesis are presented in sections 4.1 and 4.2, and results of the conversion are presented for two different input signals in our evaluation in section 5.

2. SPECTRAL MODELING
Spectral modeling synthesis [6] combines sinusoidal components and noise to re-create arbitrary harmonic structures and stochastic elements. The frequency and volume parameters of these components can be controlled over time. Parameters for additive synthesis can be extracted automatically from a sampled input sound by measuring harmonic peaks in the frequency spectrum and tracking their continuity between consecutive analysis frames. Improvements of the tracking algorithm have been proposed, e.g. by Lagrange et al. [4]. The identified peak tracks can be synthesized through a combination of sinusoidal oscillators to obtain an approximation of the original sound.
Spectral models are able to describe a wide range of arbitrary sounds, both musical and non-musical, but they do not normally include temporal structures. While MIDI files specifically encode the onset and duration of notes, spectral models encode only local frequency distributions in a time frame and capture changes in time by concatenating frames and blending between them. Unlike MIDI notes, the spectral model does not offer an easy access to the “building blocks” of a sound. For stationary sounds, i.e. sounds that do not change significantly over time, the spectral model also contains a lot of redundancy by repeating identical values. We propose to integrate the temporal evolution of peak power and frequency into a parametric model. Instead of representing a tone with slowly changing pitch through a sequence of different frequency values, we use uniform B-splines to encode its time-varying properties, thus reducing redundancy and obtaining more useful representations.

3. CUBIC B-SPLINES

A B-spline [1] uses polynomial weighting curves to interpolate a path between control points \( P_{0..m} \). For the case of a single spline segment, the curve trajectory passes through the end points and is influenced by additional control points in between. Using splines, a time-varying parameter of a sinusoidal track can be described by a handful of spline curve coefficients. The control parameter \( u \) varies from 0 to 1 along the trajectory of the curve. The influence of each control point \( P_i \) along the path is determined by the local value of its basis function \( B_i \).

\[
Q(u) = \sum_{i=0}^{m} P_i B_i(u) \tag{1}
\]

B-splines with 4 control points (or knots) and basis functions of polynomial order 3 are called cubic B-splines by convention. For uniform B-splines, the placement of the knots is equidistant along the time axis. The optimal power and frequency coefficients for the control points \( P_{0..3} \) can be efficiently obtained through least-squares fitting to a linear model \( y = X \cdot c \), where \( y \) is a vector of \( n \) observations, i.e., points in time-frequency space, and \( X \) is a matrix of predictor variables.

Green et al. [3] have proposed the use of B-splines to model the time-varying power trajectory for individual bins in a spectrogram. However, in their model, a tone that rises slowly in frequency will be represented by a set of seemingly disconnected power splines, one for each frequency within the range of the tone. We argue that the spline model can be made much more expressive if changes in frequency are encoded by spline curves as well. In theory, one time-varying sinusoid can be modeled as a pair of coupled splines: one for the power trajectory of the sinusoid and one for its frequency trajectory.

Given that the control points of the spline are equidistant for uniform B-splines and their number is fixed, a spline in our model in the time range of a partial will always be described by a fixed number of coefficients. Setting the system to allow higher-level polynomials or more breakpoints is possible, however, we have found that higher degrees of freedom can lead to unexpected results: since the fitting algorithm has no concept of simplicity and does not prefer smooth or simple curves over complex curves, it may propose “best fits” that contradict human interpretation, unless constrained otherwise. We use the implementation of the GNU Scientific Library for linear multi-parameter fitting\(^1\).

A sinusoid track has a frequency and a power component, each of which can change independently over time. Therefore, we store the variables of two independent splines (2-4 coefficients) in each track, along with the start time and length. In total, the track is thus described by 10 floating point values — regardless of its length. This representation is inherently lossy: it sacrifices some of the details for the benefit of a more expressive model. The degree of “smoothing” can be specified, so that a trade-off between the preservation of details and the model simplicity can be made.

4. ANALYSIS AND SYNTHESIS APPROACH

Our implementation uses a sinusoidal tracking method similar to the McAulay-Quatieri (MQ) algorithm [5]. Instead of using a standard short-time Fourier transform (STFT) spectrogram, we implement the time corrected instantaneous frequency (TCIF) spectrogram described in [2] to obtain increased frequency resolution.

4.1. Peak Tracking

The audio is processed one frame at a time. Samples are windowed using a Hann window function. Two time-shifted fast Fourier transforms (FFTs) are computed from the input data and a list of frequencies, powers and phases is obtained using the TCIF method. We reduce the number of peaks by deleting frequencies that are below some power threshold.

\(^1\)http://www.gnu.org/software/gsl

![Figure 1. Comparison of a single partial in a spectrogram (magnified) and the corresponding cubic B-spline, illustrated through its spline cage.](image-url)
value. After that, we combine frequencies that are closer to each other than a frequency threshold value. Since the TCIF method will shift the energy of side-lobes\(^2\) to the actual center frequency of a peak, a combination is possible without losing important information. Peaks with high power values are combined first, and the power of contributing peaks is added to them. A list of open tracks and another list of closed tracks is maintained during tracking. The following procedure is repeated at each frame:

1. **Predict track values**: At each frame, a prediction of the expected next frequency is computed for each track by computing a spline fit through the track’s existing points, but extending its time range to include the predicted time. The same is done for the power.

2. **Append peaks to tracks**: The peak that best fits the prediction for a track is appended to it (provided it lies within a specified tolerance range).

3. **Create new tracks**: Peaks that have not been assigned to a track start their own track.

4. **Close tracks**: A track is closed when it has been inactive for a specified number of frames, or when the standard deviation between the actual values and the smooth spline exceeds a specified tolerance range.

On the last frame of the audio file, remaining open tracks are closed. Very short tracks (i.e. shorter than a given number of frames) may be removed, since they are likely to contain either noise or barely noticeable sinusoids. The list of sinusoidal tracks can now be stored, manipulated or used for re-synthesis.

### 4.2. Sinusoidal Synthesis

During synthesis, all spline tracks are converted back into sinusoids separately. A sinusoidal oscillator is used to produce a time-varying signal with the frequency and volume information obtained from the corresponding spline coefficients. Since the values along the spline may be read at arbitrary resolution, the output can be very smooth and is not limited to the original analysis frame rate. In our implementation, local frequency and power values are obtained from the evaluated spline for each synthesized sample\(^3\).

Since the spline is allowed to deviate from the original samples slightly, it cannot be guaranteed that the phase of the original sound is maintained for the whole duration of the sound. We have found that good re-synthesis results can be obtained even if the sinusoids are started with random phases or zero phases. For partial tracks that belong to the same harmonic group, it may be reasonable to synchronize their phases and frequency evolution. We use a phase unwrapping technique to calculate the phase increment of the sinusoid between two subsequent samples. The phase at the beginning of each track is set to 0. Let \( r \) be the samplerate of the signal, \( p_t \) be the instantaneous power of the power spline at sample \( t \) and \( f_t \) the instantaneous frequency. The local phase increment \( d_t \) and the value \( v(t) \) can be calculated as:

\[
d_t = \frac{f_t \cdot 2\pi}{r} \quad (2)
\]

\[
v_t = p_t \cdot \sin(\varphi_{t-1} + d_t) \quad (3)
\]

We use a samplerate of 44100 Hz throughout our system. Tracks are faded in and out for 0.5 milliseconds to avoid clicks in the audio output. Each track is synthesized into a buffer, which is additively mixed into the output buffer at the start time of the track.

### 5. RESULTS

We have tested our system with different music and sound recordings to test the behavior of the analysis and synthesis procedure. We were especially interested in the trade-off between the perceptual quality of the output and the compactness of the representation.

Figure 2 shows the conversion result of a piece of music containing a singing chorus (“Take a chance on me” by ABBA). The signal contains some challenges arising from the fluent change between voiced and unvoiced portions in human speech, as well as the proximity of the individual voices in the chorus. As the figure shows, the spline-based tracking is able to form coherent splines from the singing

\[^2\]Side lobes are caused by leakage effects that shift some of the energy of a peak to analysis bins in the neighborhood.

\[^3\]The resolution of the frequency and power changes could likely be reduced and samples in between could be interpolated, leading to fewer spline evaluations.
voices, though the splines tend to be less coherent in the higher frequencies. This is likely a result of the lower volume and of the increased steepness of the slopes: since the harmonic partials are multiples of the fundamental pitch of the singing voice, the slopes of quickly rising and falling pitches are scaled by the same factor and thus appear less stable to the tracking algorithm. Still, the re-synthesized output of the splines resembles the input remarkably. The combined synthesis gives the impression of a chorus with several voices, and the lyrics are clearly understandable.

Figure 3 shows the results of the conversion for a single piano sound. The straight lines of the harmonic partials are very well modeled by a very small set of splines, yet the re-synthesized sound captures the properties of the piano quite well. Since the examples shown here represent two extremes — a complex song on the one, a single note on the other — we claim that our method can be applied to a wide range of input signals in between\(^4\).

6. CONCLUSION

In this paper, we have introduced a novel method for sinusoidal modeling using cubic B-splines. We have demonstrated the ability of our system to encode complex musical signals into splines, preserving the relevant perceptual aspects of the input signal. Furthermore, we have shown that this form of structured representation produces a more compact model than conventional spectral modeling and is based on parameters that correspond to meaningful aspects of sound that are immediately relevant for manipulation, because they include the temporal evolution of the sinusoidal elements.

We plan to extend our model to include a stochastic component for encoding the residual signal. This should help to capture phenomena like drum sounds, wind, rain or unvoiced portions in speech, and give a more natural quality to the synthesized audio.

The spline model is meant to facilitate complex manipulations of the audio stream. We plan to implement such manipulations, including the separation of components, time stretching, frequency shifting or arbitrary warping of the splines.

So far, we have not yet taken the relation between different sinusoids into consideration. It seems reasonable to combine sinusoidal tracks from the same sound source into one parametric sound model to remove redundancies further and make the model yet more expressive.

Finally, we wish to thank the anonymous reviewers who have helped to improve the quality of this paper through their valuable feedback.

7. REFERENCES


\[^4\] We provide more examples and samples of our re-synthesized clips online: http://www.informatik.uni-bremen.de/~dmoehl/audio