Technische Universität Berlin
Fakultät I – Geistes- und Bildungswissenschaften
Institut für Sprache und Kommunikation
Fachgebiet Audiokommunikation

Master's Thesis

Analysis and Synthesis of Control Parameters in Note Transitions

Philipp Moritz Götz
Matrikelnummer: 355443
23. März 2018

Erstgutachter: Prof. Dr. Stefan Weinzierl
Zweitgutachter: Henrik von Coler

Fakultät |
Geisteswissenschaften

Prüfungsausschüsse

Studiendekan und Vorsitzender

Prof. Dr. Friedrich Steinle

Eidesstattliche Erklärung

Ist jeder an der TU Berlin verfassten schriftlichen Arbeit eigenhändig unterzeichnet beizufügen!

Hiermit erkläre ich an Eides statt gegenüber der Fakultät I der Technischen Universität Berlin, dass die vorliegende, dieser Erklärung angefügte Arbeit selbstständig und nur unter Zuhilfenahme der im Literaturverzeichnis genannten Quellen und Hilfsmittel angefertigt wurde. Alle Stellen der Arbeit, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen wurden, sind kenntlich gemacht. Ich reiche die Arbeit erstmals als Prüfungsleistung ein. Ich versichere, dass diese Arbeit oder wesentliche Teile dieser Arbeit nicht bereits dem Leistungserwerb in einer anderen Lehrveranstaltung zugrunde lagen.

Titel der schriftlichen Arbeit

VerfasserIn/VerfasserInnen*

Name | Vorname | Matr.-Nr.

Betreuende/r DozentIn

Name | Vorname

Mit meiner Unterschrift bestätige ich, dass ich über fachübliche Zitierregeln unterrichtet worden bin und verstanden habe. Die im betroffenen Fachgebiet üblichen Zitiervorschriften sind eingehalten worden.

Eine Überprüfung der Arbeit auf Plagiate mithilfe elektronischer Hilfsmittel darf vorgenommen werden.

Ort, Datum | Unterschrift**

*Bei Gruppenarbeiten sind die Unterschriften aller VerfasserInnen erforderlich.
**Durch die Unterschrift bürgen Sie für den vollumfänglichen Inhalt der Endversion dieser schriftlichen Arbeit.

Abstract At the audio communication group a library consisting of note transitions played by a violin was recorded. The volume and the fundamental frequency trajectories from recordings of two notes were extracted and different models for approximation of the trajectories were developed: a simple model with a hyperbolic tangent function, a model with natural splines and a model with beziér curves. The natural splines curves showed the best results when using a mean absolute error measurement. A synthesizer framework was developed in C++ on a Raspberry Pi to use the models in real-time. Further evaluation of the models was done in a user study. The question was if the models and their number of parameter have an influence on how precise the fundamental frequency of the analyzed transitions can be reproduced. The study showed that the models have a significant influence on the difference between the trajectories created by users and the original fundamental frequency trajectory. Both the beziér curves and the spline curves lead to significant better results than the hyperbolic tangent model when recreating fundamental frequency trajectories. Overall the spline curve model performed best. The approximation results had the lowest error and with the use of two parameters for curve adjustment the majority of users preferred this model in the study.

Zusammenfassung Am Fachgebiet Audiokommunikation wurden Aufnahmen von Notenübergängen, gespielt von einer Konzertviolinistin, aufgenommen. Kurven für die Lautstärke und die Grundfrequenz wurden aus Aufnahmen von zwei Tönen extrahiert und Modelle für die Approximation dieser Kurven entwickelt: ein Modell mit einer Tangens hyperbolicus Funktion, ein Modell mit natürlichen Splines und eines mit Beziér Kurven. Die Spline Kurven wiesen den geringsten mittleren absoluten Fehler bei Vergleich der modellierten Kurven mit den extrahierten Kurven auf. Auf einem Raspberry Pi wurde eine Framework zur Synthese in C++ entwickelt um die Modelle in Echtzeit zu spielen. Weiterhin wurde eine Studie zur Evaluation der Modelle durchgeführt, in der Teilnehmer gehörte Notenübergänge nachspielen sollten. Untersucht wurde der Einfluss der Modelle und deren Parameteranzahl auf die Genauigkeit der Reproduktion der Grundfrequenz. Ergebnisse zeigten, dass die Auswahl des Modells einen signifikanten Einfluss hat. Die natürlichen Spline und die Beziér Kurven zeigten dabei signifikant bessere Ergebnisse als die Tangens hyperbolicus Funktion. Insgesamt hatte das Modell mit natürlichen Splines die besten Ergebnisse: Die Approximation zeigte die geringsten Fehler und bei einer Nutzung von zwei Parametern bevorzugte die Mehrheit der Studienteilnehmer dieses Modell.

Acknowledgments

I would like to thank my professor Prof. Dr. Stefan Weinzierl and my supervisor Henrik von Coler for their effort. Special thanks to Henrik for the numerous meetings we had.

Advice given by Dr. Steffen Lepa has been a great help in the evaluation of the user study.

Thanks to all the students who took part in the user study.

Contents

1. Introduction

When synthesizing a melody or playing an instruments ornamentation plays an important role. Depending on the instrument there are a lot of different techniques that can be applied. One component of the ornamentation is how note transitions are played. Electronic instruments such as synthesizers are capable of producing continuous note transitions. One way implementing these transitions is a linear change of the fundamental frequency from one note to another. For acoustic instruments like a violin, transitions cannot be described in such an easy way. Developing mathematical models for synthesis of note transitions is the aim of this thesis.

Before finding models a deeper look into synthesizers and their way of producing continuous transitions is made. Russ [1] describes a so called portamento circuit which is used in synthesizers:

> The portamento circuits in analogue synthesizers work by restricting the rate at which a CV [control voltage] can change. Normally, the pitch CV from a keyboard will change rapidly when a new note is selected. A portamento circuit changes the slope of the transition between the two voltages. It thus takes time for the note to move from the existing pitch to the new pitch (p.171).

Control of portamento was often placed near the keyboard and depending on the synthesizer it could be switched on an off during play for expressive performance according to Jenkins [2]. The only additional parameter was the time the transition needs and no parameter about the curve form was described. Digital synthesizers on the other hand offer different curve forms, like the Ultranova by Novation Digital Music Systems Ltd. [3]: a linear and an exponential curve are selectable for a continuous note transition. But still no parameter which offers further control about the curve form is available.

Instead of recreating the function of analog synthesizer circuits the behavior of an professional musician on a instrument is analyzed. With this approach models can be developed which then offer a variety of parameters for curve manipulation. Therefore extracting control parameters of note transitions from recorded samples is done. In the next chapter the analysis of the recorded audio files and the modeling of the control parameters is presented. In chapter 3 the real-time synthesis of these models is described. Then the models are evaluated in a user study which is the topic of chapter 4. And finally in chapter 5 the overall results are discussed.

2. Analysis and Modeling

This chapter addresses the analysis and modeling of note transitions. First the audio material used for analysis is described and categorized into different transition types. The methods used to extract different control parameters out of the audio material are presented. After that results of the analysis are shown and discussed. Different models for the use of synthesizing transitions are then proposed and evaluated.

2.1. Analysis - Parameter Extraction

First a set of control parameters needs to be chosen which then can be extracted to provide a basis for the development of note transition models and their consecutive evaluation. The data set used here are sequences of two notes played on a violin. Details about the data set are described in the next section. For extraction a pure tone is taken as a basic model, because it is not intended to reproduce the complete timbre and behavior of a violin. Thus the characteristics of playing a violin like the movement of the bow and the physical aspects of the body and the resulting resonance are ignored. Therefore a single sinusoid can be used as the basic model for parameter extraction. The formula for a sinusoid is:

$$x(t) = a \sin(\omega t) \tag{2.1}$$

So there are two parameters: amplitude a and the angular frequency ω. According to Puckette [4] a good comparable measure for the signals amplitude is the root mean square (RMS) amplitude which is used as the first control parameter. The fundamental frequency in Hertz of the sinusoid f_0 is equal to $2\pi\omega$ and builds the second control parameter. The fundamental frequency is often used for a basic analysis of audio signals for example by Ikemiya et al. [5], Gómez et al. [6] and Dai et al. [7]. These two control parameters are then extracted from the recorded material.

2.1.1. Recorded Material

At the audio communication group a library of transitions was recorded. A concert violinist was instructed to play two notes with different articulation techniques for the transition. Overall the library consists of 344 note transitions which differ in the starting and ending note, the transition type, dynamic and if vibrato is used. In table 2.1 the different parameters are shown. The recordings were labeled after the scheme described

by von Coler and Lerch [8] with the help of the software Sonic Visualizer[1]. With this segmentation the audio files, each consisting of a sequence of two notes, have seven labels: rest, transition, note, transition, note, transition, rest. This thesis focuses on the transitions from one note to another.

Table 2.1.: Different parameter values in the violin library

Parameter	possible Values
Volume	*mp*, *ff*
Articulation	legato, glissando, detached
Vibrato	on, off
Semitones	0, 5, 7
Direction	up, down
Note 1	d', a', e", b"
Note 2	a, e', b', fis", g', d", a", e"', d', a', e", b", g

2.1.2. Transition Classification

Now a deeper look into what behavior is expected from the transitions is taken. The transitions are categorized here by the different articulations used, as shown in table 2.1. First the differences in playing for the articulations are described. After that spectrograms for each articulation are presented. With these information simplified models are described.

The three different articulations used in the library can be described in simple form with the following violin playing technique: for glissando the player is sliding the finger along the string from the position of the first note to the second note. During this slide the string is constantly excited by the movement of the bow. For legato the player first plays one note and then without any movement of this finger another finger pushes another note on a string. This also involves constant excitation of the strings. The last articulation used here is detached. This is basically the same as legato except there is pause in the excitation.

The glissando playing leads to a spectrogram shown in figure 2.1. In this case a transition from A3 (221.5 Hz) to D4 (295.66 Hz) is shown. For each tone the fundamental frequency and the harmonics are clearly recognizable. As expected from the sliding with the finger between the notes on the fingerboard a smooth change in frequency is visible. The behavior of the fundamental frequency can be translated into a simplified model. Also the power of the signal which is seen as volume can be translated into a simplified

[1]It is an application designed for viewing and analyzing audio files from the Centre for Digital Music at Queen Mary, University of London. Downloadable for free at `http://www.sonicvisualiser.org/` [20.03.2018]

model. It is shown in figure 2.2. The smooth change of the fundamental frequency and a constant volume is presented. At time t_1 the glide is starting.

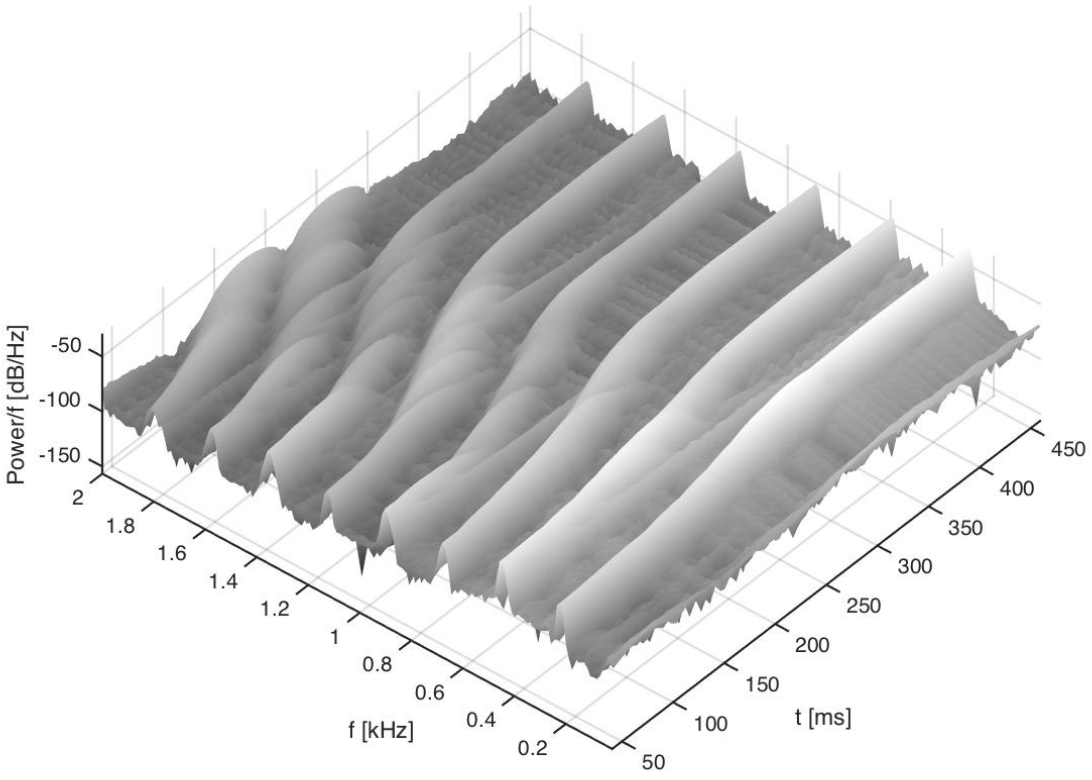

Figure 2.1.: Spectrogram of a transition, glissando

Analogous to the glissando articulation the spectrogram of a legato transition is shown in figure 2.3. This transition is going from A4 (443 Hz) to E4 (331.87 Hz). These frequencies and their harmonics are obviously noticeable. The attenuation in power of the first note and the rise in power of the second note are visible. In addition there is an overlap between the two notes. All these observations lead to the model presented in figure 2.4. The overlap in volume is reflected as well as a second frequency at t_1 when the second note starts and the first note is still active.

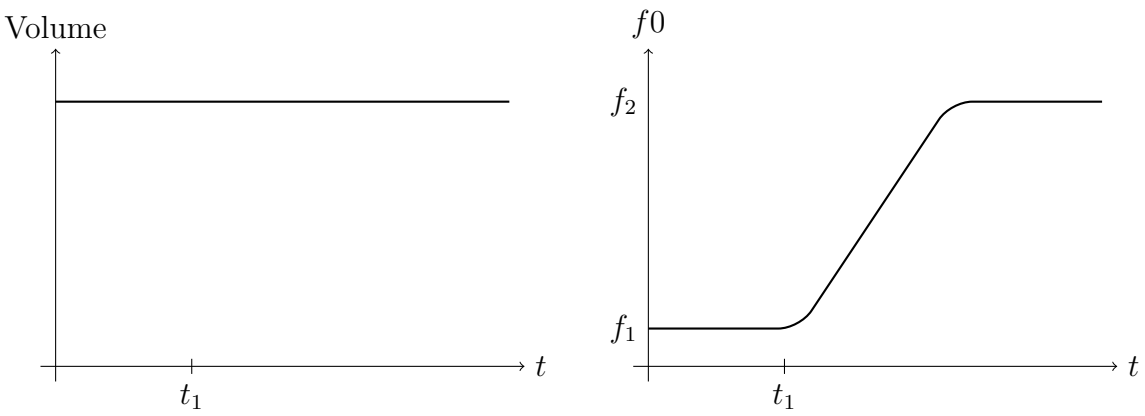

Figure 2.2.: Simplified transition model for glissando articulation

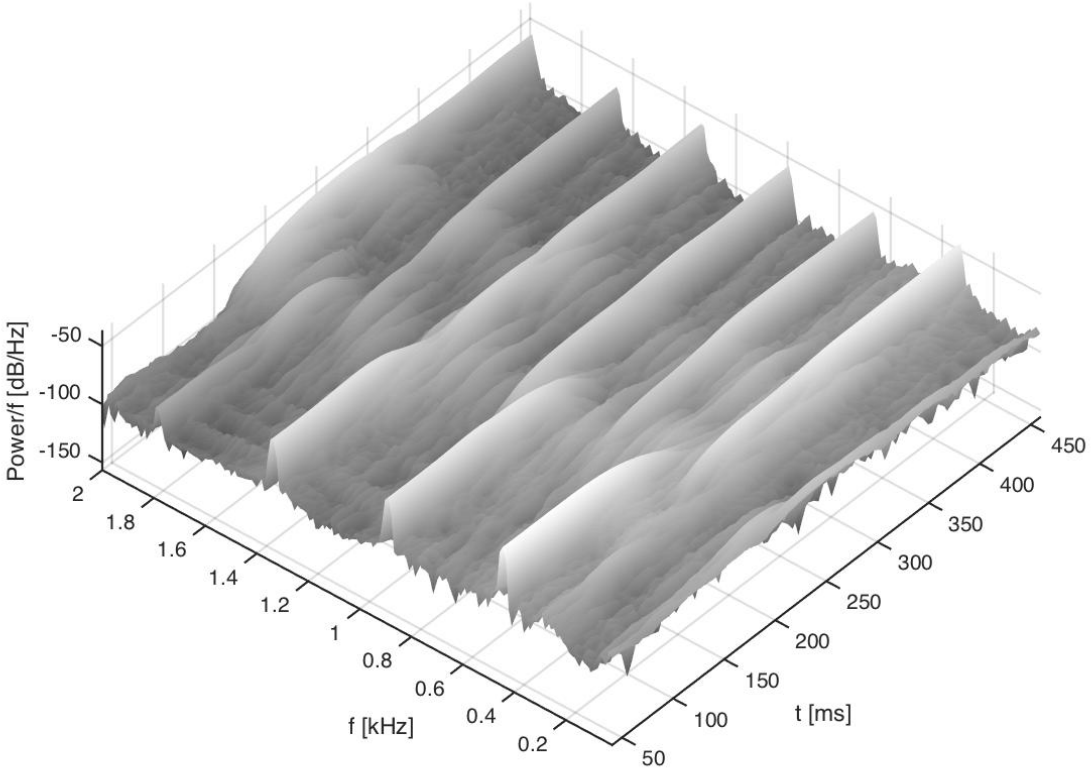

Figure 2.3.: Spectrogram of a transition, legato

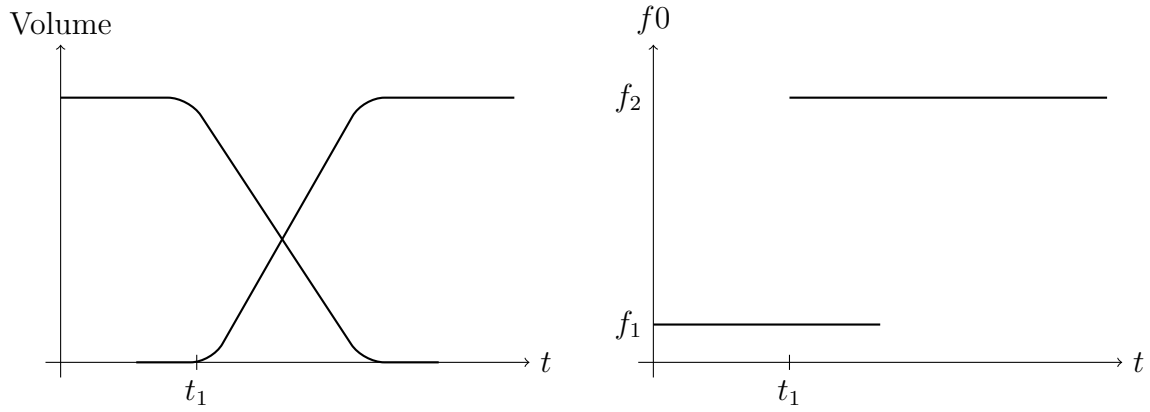

Figure 2.4.: Simplified transition model for legato articulation

For the last articulation used, a spectrogram is shown in figure 2.5. In this case the notes A3 (221.5 Hz) and D4 (295.66 Hz) were played. The fundamental frequency and the harmonics are clearly visible for both tones. Also visible are the volume attenuation of the first note and the volume rise of the second note. As expected no smooth connection between the two different tones is visible. This behavior can be translated into a simpler model, shown in figure 2.6. At t_1 the excitation of the first note ends and at t_2 the excitation of the second note starts.

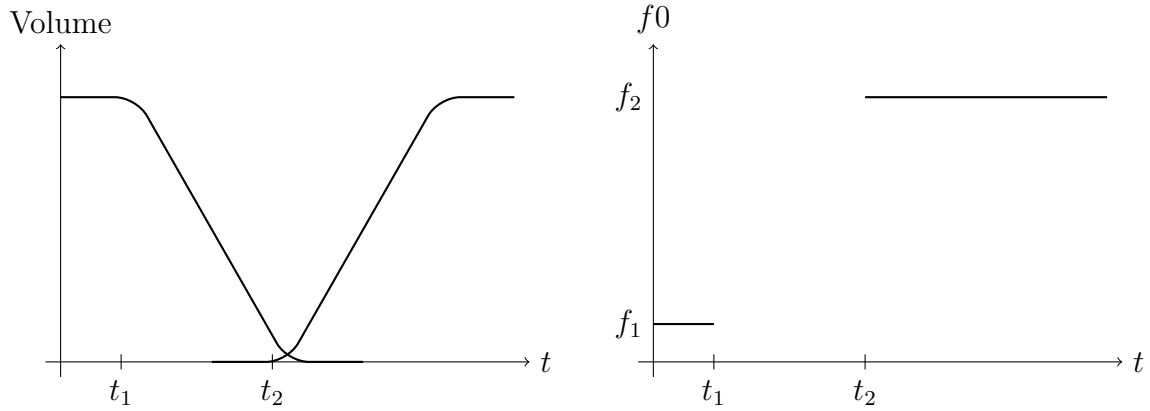

Figure 2.6.: Simplified transition model for detached articulation

With these models a decision for further analysis has to be made. Modeling of the fundamental frequency of transitions with detached and legato articulation is straightforward. During the transition the frequency first is constant, then there is a jump and after that it is constant again. Also the volume during these transition is comparable to modeling the attack and the decay of a tone without another note following. In conclusion modeling these transition can be achieved by modeling volume envelopes. By contrast, glissando

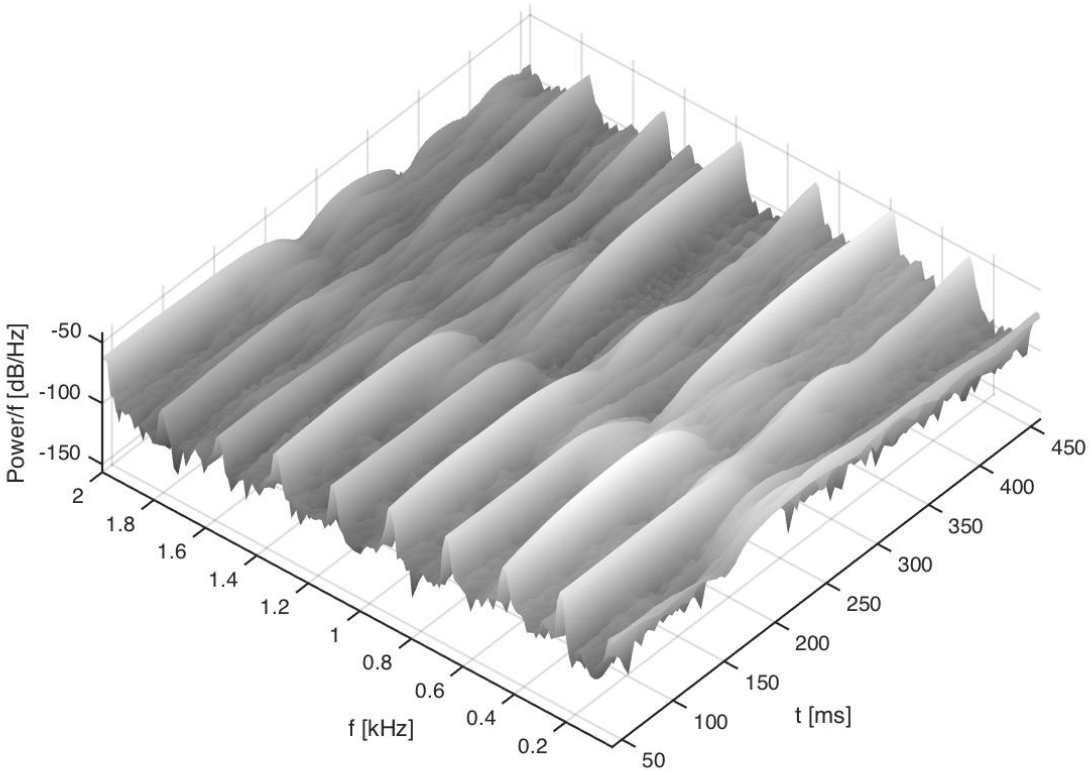

Figure 2.5.: Spectrogram of a transition, detached

transitions require a further analysis. A model for the continuous frequency change and a deeper look into the behavior of the volume of the signal is necessary.

2.1.3. Algorithms for Extraction

Glissando transitions are now in focus of the analysis and for the two control parameters proper algorithms are needed.

Fundamental Frequency

For the extraction of the fundamental frequency (f0) trajectory a suitable algorithm is a prerequisite. In this specific use-case the algorithm has certain requirements. With the knowledge of the starting and the ending note the algorithm only needs to detect frequencies in a specific range. Furthermore the algorithm only needs to deal with monophonic input data due to the fact that the library only consists of monophonic violin recordings. A ground truth is needed to test the trajectories calculated by the algorithms. For this purpose the spectrogram and the starting and ending notes are a

good basis for manual evaluation.

In the preexisting code written by Henrik von Coler he used a MATLAB implementation of the SWIPE Algorithm from Camacho [9] for f0 detection. This implementation of the Algorithm has some drawbacks. The Algorithm often detects false trajectories. Different sets of parameters did not solve the problem. Sometimes the algorithm calculates unhelpful trajectories when the first detected value is not within the interval consisting of starting and ending note of the analyzed transition or the f0 value is completely outside the range in which it is expected. Out of the overall 96 glissando transitions 56 trajectories were false.

With the information about the beginning and the ending note of a transition, a filter can be implemented before using the SWIPE algorithm. But it must be taken into account that filtering involves delaying the signal depending on the filter. To get the f0 trajectory and other parameter trajectories synchronous to the filter output the filter should be designed with constant group delay. In this case a FIR filter design is chosen, because in contrast to IIR filters they can be designed to fulfill this requirement.

With the help of a MATLAB FIR-Filter design function each audio sample is filtered with a low-pass and a high-pass and filter. The function uses the classical method of windowed linear-phase FIR digital filter design described by Digital Signal Processing Committee [10]. The audio data is processed by convolution with the respective filter coefficients. The group delay of the filter is used to truncate the output of the filter: the first n samples and the last n samples of the output are discarded. n is equal to the group delay. Then the filtered output has the same length as the input and the phases of the in- and output are identical inside the pass band.

By applying the filter prior to the f0 trajectory calculation, the SWIPE algorithm achieves slightly better results: 34 out of 96 transition were falsely detected.

Considering the analysis results the need for another algorithm is clearly recognizeable. The YIN algorithm proposed by De Cheveigné and Kawahara [11] is used. A MATLAB implementation by the authors of the algorithm is used and none of the 96 transitions was extracted with an obviously false trajectory.

Volume

As mentioned before the amplitude of a signal x is described by the root mean square (RMS) value. Equation 2.2 is mathematically identical to the one from Puckette [4] but with a different notation:

$$RMS = \sqrt{\frac{\sum_{i=0}^{N-1} x_i^2}{N}}. \tag{2.2}$$

In the analysis the RMS value is calculated for each audio block, which consists of N samples.

2.1.4. Results

All the glissando transitions are now processed with the two algorithms described before. The parameters for the algorithms are shown in the following table:

Table 2.2.: Parameters for control parameter extraction

	parameter	value
general		
	f_s	96 kHz
	blocksize	4096 samples
	hopsize	512 samples
YIN		
	$f0_{min}$	starting note +/- 150 cent
	$f0_{max}$	ending note +/- 150 cent

The expected range of f0 depends on if the transition is from a lower to higher note or vice versa. If the starting note is lower than the ending note $f0_{min}$ is the starting note minus 150 cent and $f0_{max}$ is the ending note + 150 cent. For one glissando transition the result of the volume and f0 extraction is shown in figure 2.7. Additionally the starting and the ending note is plotted as a dashed line in the graph of the f0 trajectory.

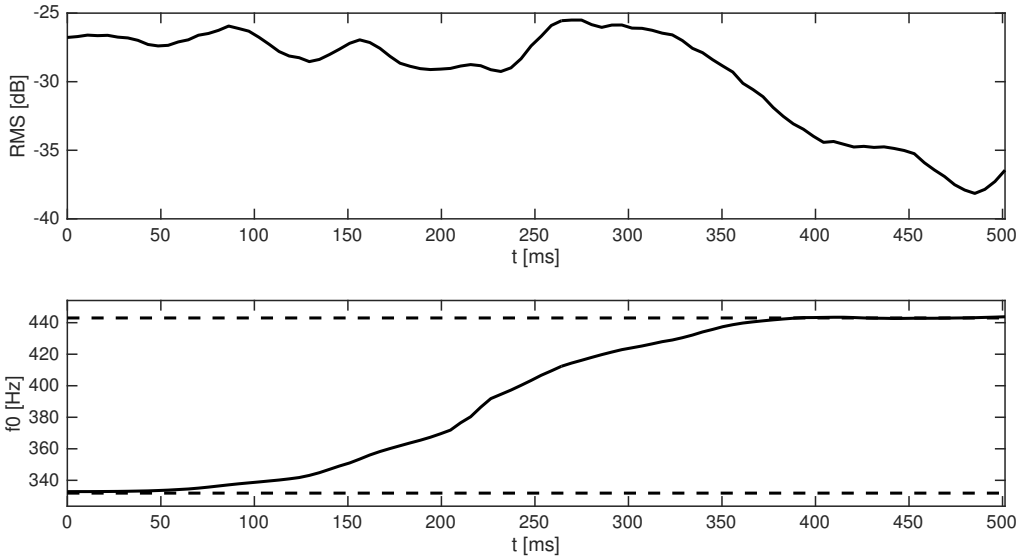

Figure 2.7.: RMS and f0 trajectory of one glissando transition

2.2. Modeling

The calculated trajectories from previous section should be described in a more general form. The goal is finding formulas that can be used for real-time synthesis. Different algorithms for approximation are described and evaluated.

In the previous section we found curves for two different parameters: f0 and volume. One topic discussed here is if these two parameters need different interpolation algorithms. One main requirement is the parameterization of the resulting curves. The number of parameters per curve depends on the algorithm.

2.2.1. Real-Time Requirements

The term real-time often arose in previous sections. This term needs some explanation in the context of audio and synthesis. Scholz [12] uses the definition from the Oxford Dictionary of Computing: real-time means a system can always provide correct results in a defined period of time. It means that even if the system provides the correct results after this period of time the system fails. In the audio context this means that the calculation of a block of audio samples must be finished after a certain amount of time. This time period is defined by the length of the processed audio block and the sample rate. An application with a block size of 64 samples and a sampling rate of 48000 Hz for example has $\frac{64}{48000Hz} = 1.33ms$ to calculate each block. On the target system this time period can be converted to processor cycles and every function then can be measured. To assure this behavior we can propose requirements to the algorithms used for audio calculation:

- Every part of the audio calculation should be done in a definite amount of time. Some algorithms have an undefined amount of iterations to get their result, for example recursive newton interpolation described by Schwarz and Köckler [13].
- The overall amount of time needed for calculation is measured by taking the longest calculation path for each algorithm. For example if some parameters need to be recalculated only every second this should be taken into account.
- There should never be an algorithm in the calculation that has to wait for an undefined amount of time to start the calculation.

With these requirements in mind we can describe and evaluate algorithms for the use of real-time audio synthesis in the next section.

2.2.2. Algorithms

In this section different algorithms for approximation of the trajectories are described and evaluated. First an error measurement is introduced and then the algorithms are described. For the algorithms there are different approaches. The first three are with widely known functions, linear, exponential and hyperbolic tangent. Their parameters

can be used to improve the result of the approximation. For the next approach points from a trajectory are used for the calculation of a cubic spline curve between them. The last approach is a bézier curve, which is using basic functions and control points. These points do not lie on the curve itself most of the time.

Linear Transition

Before using algorithms for trajectory approximation a look at ways of creating a transition between starting and ending notes without the use of other data points is done. The most obvious way for a transition between two points is a linear connection. For this the general form of the equation of a straight line by Papula [14] is used:

$$y(x) = mx + b. \tag{2.3}$$

Exponential Smoothing

Another way to create a smooth curve between data points is stated by Brown [15]. The definition of the smoothed function is:

$$S_t(x) = \alpha x_t + (1 - \alpha) S_{t-1}(x). \tag{2.4}$$

For an audio related use case we adapt the definition of the exponential smoothing factor α from Zölzer et al. [16] to:

$$\alpha = 1 - \exp\left(\frac{-\Delta t}{\tau}\right). \tag{2.5}$$

τ is called the time constant and is defined as the amount of time in which the function reaches $1 - e^{-1} (\approx 0,63)$ of the original signal after excitation with a unit set function. Δt is the time between the x - values of the function. For example the hopsize can be used. Figure 2.8 shows examples for exponential smoothing. In this case the we used two points $y_0 = 0$, $y_1 = 1$ and different time constants τ. The red dashed line shows $\sim 63\%$ of y_1. This curve was created using the following formula:

$$\begin{aligned} S(n) &= y_0, \quad n = 0, \\ S(n) &= \alpha y_1 + (1 - \alpha) S(n-1), \quad n > 0, n \in \mathbb{N} \end{aligned} \tag{2.6}$$

Hyperbolic Tangent

The third way we consider is a transition with the help of a hyperbolic tangent function defined by Papula [14]. The functions needs some improvements to fit our needs:

$$T(x) = c + d \tanh\left(\frac{x - a}{b}\right), \quad x, a, b, c, d \in \mathbb{R} \tag{2.7}$$

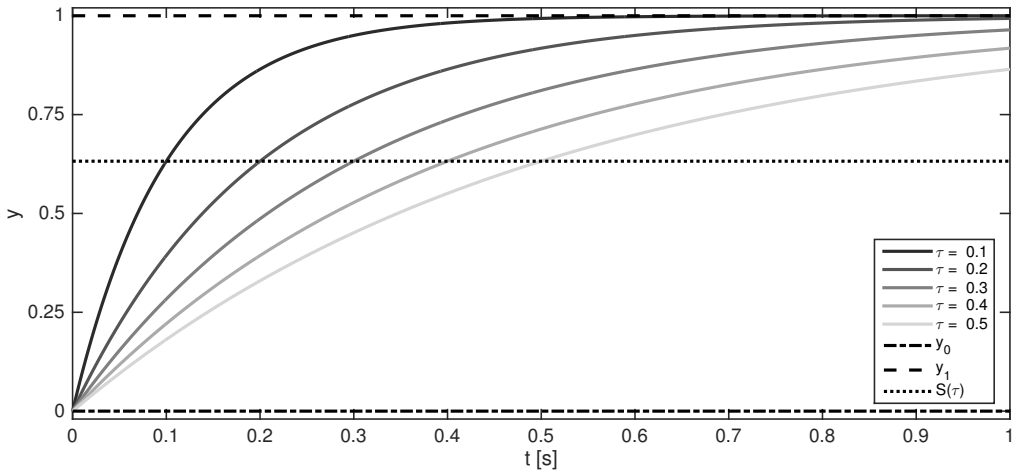

Figure 2.8.: Exponential smoothing with different values for τ

For a transition between two values y_0 and y_1 the parameters c and d must be:

$$\begin{aligned} d &= \frac{|y_0 - y_1|}{2}, \\ c &= min(y_0, y_1) + d. \end{aligned} \tag{2.8}$$

Parameter a is depending on the x - values. For a transition between the first value x_0 and the last value x_1 parameter a must be:

$$a = \frac{|x_0 - x_1|}{2}. \tag{2.9}$$

The last parameter b then controls the slope of the function. In figure 2.9 a hyperbolic tangent curve is plotted with different values for b.

Cubic Splines

Splines are a set of functions for piecewise interpolation where every piece is defined by a polynomial.
According to Unser [17] and Schweizer [18] splines find regular use in signal processing as well as data interpolation and seem to be a good choice for our modeling task, since they are robust against data fluctuation whilst having low processing costs. Here cubic splines are used, which means every piece is described by a cubic polynomial, because higher order polynomials could cause unwanted oscillation at the edges. Splines and mostly B-Splines are widely used for f0-interpolation, by Ardaillon et al. [19], Barbot et al. [20], Hahn et al. [21], Hahn and Röbel [22], Lolive et al. [23], Lolive et al. [24], Lolive et al. [25] and Robel [26]. Here the natural cubic splines are used instead of B-Splines, because for interpolation with B-Splines the control points used do not lie on the curve itself. This

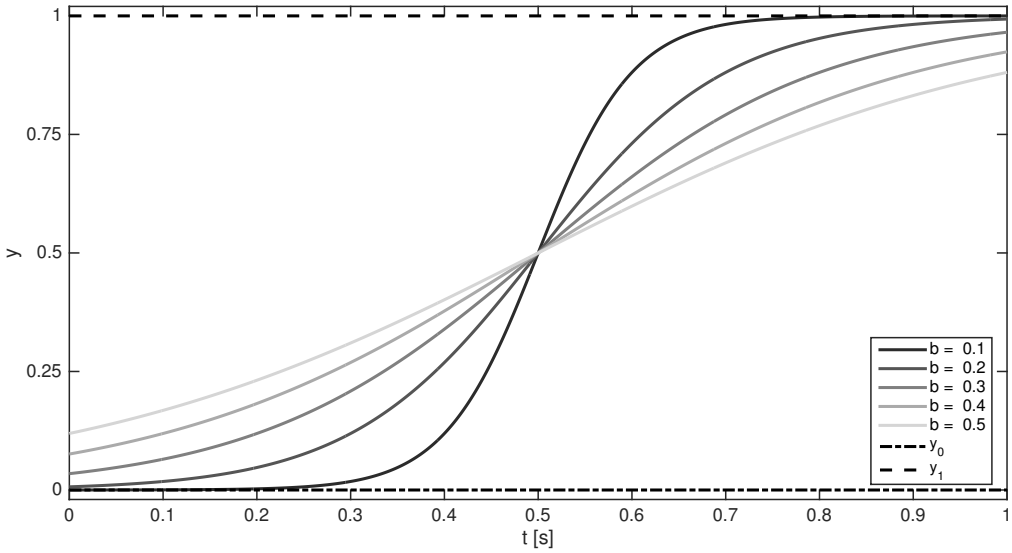

Figure 2.9.: Hyperbolic tangent smoothing with different values for b

curve fitting approach is used with bézier curves in the next section. Another reason why natural cubic splines are chosen is that there is no curvature at the starting and the ending point of the curve, which means the second and the third derivatives at these points are zero, and the curve can continue as a straight line. The following formulas used for the calculation of natural cubic splines are taken from Engeln-Müllges et al. [27].

There are n points $P_i = (x_i, y_i)$ and the resulting graph containing all the points is called S. Between the points P_i and P_{i+1} the segments S_i are represented by a cubic polynomial:

$$\begin{aligned} S_i(x) &:= a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \\ & x \in [x_i, x_{i+1}], a_i, b_i, c_i.d_i \in \mathbb{R}, i = 0, 1, ..., n - 1. \end{aligned} \tag{2.10}$$

The goal is having a formula for a graph which parameters are depending on the points P_i. In a general form this leads to a set of equations. The equations are simplified by using the following substitution:

$$h_i = x_{i+1} - x_i, \quad i = 0, 1, ..., n - 1. \tag{2.11}$$

Equations have to be formulated for the coefficients a_i,b_i,c_i and c_i. The coefficients a_i are:

$$a_i = y_i, \quad i = 0, 1, ..., n - 1. \tag{2.12}$$

Next step is buildings an equation System $Ac = a$. The first and the last coefficient c_0

and c_n are set to 0 due to the boundary conditions of natural cubic splines. The Matrix A is:

$$A = \begin{pmatrix} 2(h_0+h_1) & h_1 & & & & \\ h_1 & 2(h_1+h_2) & h_2 & & & \\ & h_2 & 2(h_2+h_3) & h_3 & & \\ & \ddots & \ddots & \ddots & & \\ & & h_{n-3} & 2(h_{n-3}+h_{n-2}) & h_{n-2} & \\ & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{pmatrix} \tag{2.13}$$

Vectors c and a are:

$$c = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{pmatrix}, \quad a = \begin{pmatrix} 3\frac{y_2-y_1}{h_1} - 3\frac{y_1-y_0}{h_0} \\ 3\frac{y_3-y_2}{h_2} - 3\frac{y_2-y_1}{h_1} \\ \vdots \\ 3\frac{y_n-y_{n-1}}{h_{n-1}} - 3\frac{y_{n-1}-y_{n-2}}{h_{n-2}} \end{pmatrix} \tag{2.14}$$

With the resulting c vector the remaining b_i and d_i are:

$$b_i = \frac{y_{i+1}-y_i}{h_i} - \frac{h_i}{3}(c_{i+1}+2c_i), \quad i = 0, 1, ..., n-1 \tag{2.15}$$

and

$$d_i = \frac{1}{3h_i}(c_{i+1}-c_i), \quad i = 0, 1, ..., n-1. \tag{2.16}$$

With this set of equations several points out of extracted f0-trajectories can be used to get a polynomial representation of the curve. In figure 2.10 a natural cubic spline curve is plotted with the four points $P_i = (0,0), (\frac{1}{3}, 0.8), (\frac{2}{3}, 0.5), (1,1)$. For a large set of points the computation time needed for calculation can rise depending on the algorithm used for matrix solving. Also the choice of points (e.g equidistant) needs evaluation. This is done in the next section.

Bézier Curves

As we have seen in the previous section cubic splines offer a way to approximate curves. The next approach presented are bézier curves. These curves are often used in the context of computer graphics, since they create a smooth modifiable output. They are rarely used in a computer music context but for example Battey [28] also used them for f0 modeling. One major difference to natural cubic splines is that the control points for the polynomial are not part of the curve itself. Another major difference is that the curve definition is not piecewise. This leads to a modified approach for finding the polynomials for approximation. The general definition of bézier curves $K(x)$ from Schwarz and Köckler

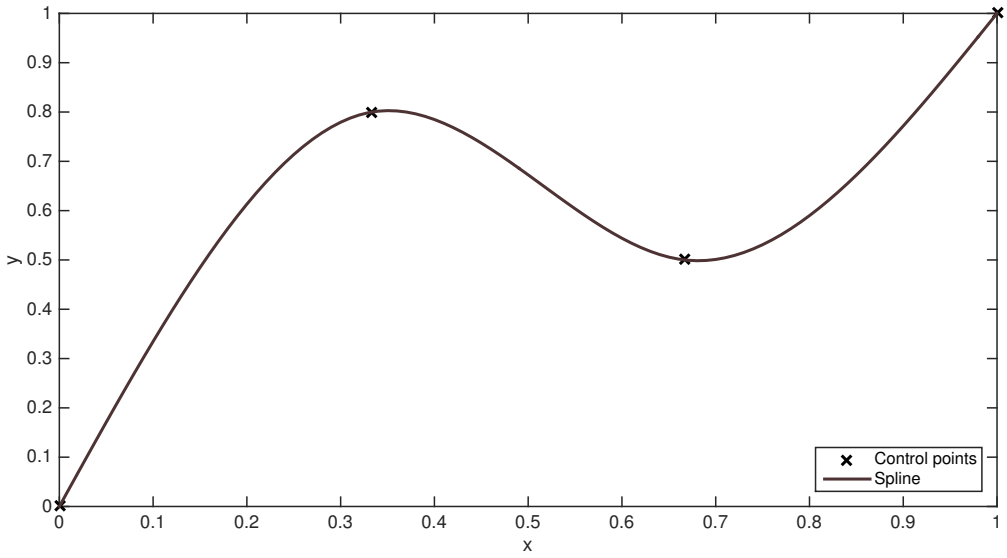

Figure 2.10.: Natural cubic spline

[13] is the following:

$$K(x) = \sum_{i=0}^{n} b_i B_i^n(x), n \in \mathbb{N} \tag{2.17}$$

where b_i are the y values of the bézier points C_i. The x values of C_i are:

$$C_{i,x} = \frac{i}{n}, \quad i = 0, 1, ..., n \tag{2.18}$$

This means the bézier points are equidistant on the x axis. $B_i^n(x)$ are the bernstein polynomials. The bernstein polynomials are defined as:

$$B_i^n(x) = \frac{1}{(b-a)^n} \binom{n}{i} (x-a)^i (b-x)^{n-i}, \quad x \in [a, b], a, b \in \mathbb{R}. \tag{2.19}$$

n is the polynomial degree. To use these polynomials we need another restriction: $a = 0$ and $b = 1$. For the bernstein polynomials this means:

$$B_i^n(x) = \binom{n}{i} x^i (1-x)^{n-i}, \quad x \in [0, 1]. \tag{2.20}$$

For a bézier curve that goes through of a set of points $P_i = (x_i, y_i)$ we need to find the corresponding C_i. Now an example for a set of four points is presented. The x values of the points P_i must be the same as the x values from C_i. So we need to choose four equidistant points. In this case the polynomial degree is $n = 3$. The bernstein polynomials

are:

Table 2.3.: Bernstein polynomials and control points

i	B_i^3	C_i
0	$(1-x)^3$	$(0, b_0)$
1	$3(1-x)^2x$	$(\frac{1}{3}, b_1)$
2	$3(1-x)x^2$	$(\frac{2}{3}, b_2)$
3	x^3	$(1, b_3)$

Then $K(x)$ is:

$$K(x) = b_0(1-x)^3 + b_1 3(1-x)^2 x + b_2 3(1-x)x^2 + b_3 x^3, \quad x \in [0,1]. \tag{2.21}$$

With the four points $P_i = (x_i, y_i)$ which lie on $K(x)$ we can write:

$$y_i = b_0(1-x_i)^3 + b_1 3(1-x_i)^2 x_i + b_2 3(1-x_i)x_i^2 + b_3 x_i^3, \quad i = 0,1,2,3 \tag{2.22}$$

Similar to the splines this can be transformed into an equation system $Ab = a$. The matrix A is then:

$$A = \begin{pmatrix} (1-x_0)^3 & (1-x_0)^2 x_0 & 3(1-x_0)x_0^2 & x_0^3 \\ (1-x_1)^3 & (1-x_1)^2 x_1 & 3(1-x_1)x_1^2 & x_1^3 \\ (1-x_2)^3 & (1-x_2)^2 x_2 & 3(1-x_2)x_2^2 & x_2^3 \\ (1-x_3)^3 & (1-x_3)^2 x_3 & 3(1-x_3)x_3^2 & x_3^3 \end{pmatrix}, \tag{2.23}$$

vectors b and a are:

$$b = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}, \quad a = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}. \tag{2.24}$$

This can be written in a more general form for $n-1$ points. The equation system $Ab = a$ is then:

$$A = \begin{pmatrix} B_0^n(x_0) & B_1^n(x_0) & \dots & B_{n-1}^n(x_0) & B_n^n(x_0) \\ B_0^n(x_1) & B_1^n(x_1) & \dots & B_{n-1}^n(x_1) & B_n^n(x_1) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ B_0^n(x_{n-1}) & B_1^n(x_{n-1}) & \dots & B_{n-1}^n(x_{n-1}) & B_n^n(x_{n-1}) \\ B_0^n(x_n) & B_1^n(x_n) & \dots & B_{n-1}^n(x_n) & B_n^n(x_n) \end{pmatrix}, \tag{2.25}$$

$$b = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix}, \quad a = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \\ y_n \end{pmatrix}. \tag{2.26}$$

In figure 2.11 a bézier curve is plotted with the four control points $P_i = (0,0), (\frac{1}{3}, 0.8), (\frac{2}{3}, 0.5), (1,1)$. The bézier points C_i are then $(0,0)$, $(\frac{1}{3}, 1.983)$, $(\frac{2}{3}, -0.8667)$ and $(1,1)$. They form the control polygon, which is the blue dashed line in the figure. For synthesis the y-values of these points, here also referred as b_i, can be used to modify the curve without causing a recalculation of the equation system.

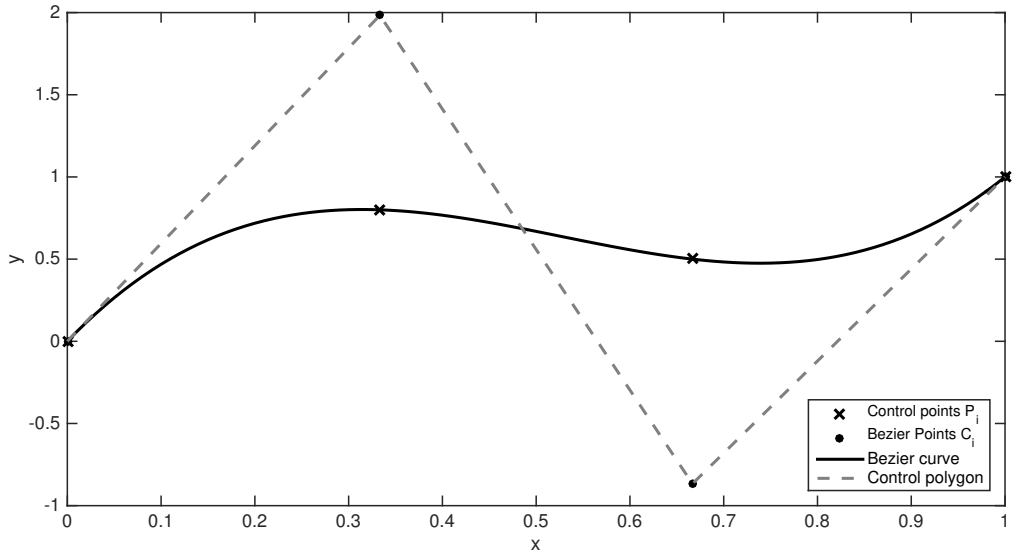

Figure 2.11.: Bézier Curve

2.2.3. Extracting the parameters

In the previous section the theory for all the used algorithms was presented. Before going into more detail about the implementation of the algorithms the input data was modified. One goal here is to find a general form for a glissando transition from one note to another. To achieve that, a comparable form of the extracted trajectories was needed. It was made by normalizing the transitions. Each transition is normalized individually, so that $y = 0$ corresponds to the starting point and $y = 1$ to the ending point of the transition. For the frequency the starting note corresponds to 0 and the ending note to 1.

All of them were implemented as a function in MATLAB. For the exponential smoothing and the hyperbolic tangent theory does not give a straightforward solution which parameters might lead to good results. For the exponential smoothing a set of curves were created with a different α for each curve. Then the mean absolute error was calculated and the α with the lowest error was picked. The same procedure was made for hyperbolic tangent curve. The parameter varied was b. For a linear transition there is no parameter to influence the curve if the line goes from the starting note to the ending note.

2.2.4. Analysis of the modeling parameters

Now that every model provides a parameter set for each note transition we can try to find correlation between the attributes and the extracted modeling parameters, for example a correlation between the dynamic of the transition and the parameters used for a spline modeling. The goal is finding a parameter set for the different attributes that a potential user can pick when synthesizing a transition.

Clustering

A general overview about clustering algorithms is provided by Xu and Wunsch [29]. According to them clustering has no straightforward solution and needs trials and repetitions. The goal of the clustering is to find a correlation between the clusters and the attributes of the transitions, shown in table 2.4. With the extracted parameters for the proposed models the transitions can be categorized into clusters. The k-means algorithm, initially proposed by Lloyd [30], widely used in statistics, is used here in a modified version by Arthur and Vassilvitskii [31], known as k-means++. The algorithm tries to find k centers for n data points in $\mathbb{R}^d$ and minimize the squared distance between each point and its closest center.

One problem is finding the right number of clusters. A graphical visualization for finding a suitable number of clusters are silhouettes. First described by Rousseeuw [32]. This algorithm is used to evaluate the number of cluster centers found by a clustering algorithm. The data set where clusters can be found contains for example all extracted parameters for all f0 trajectories for one model. After finding clusters for k different centers a correlation between attributes of the transitions and the clusters is analyzed. Attributes of the transitions are:

Table 2.4.: Attributes of the transitions

attribute	value 1	value 2	value 3
dynamic	*mp*: mezzopiano	*ff*: fortissimo	
Δf	0 semitones	5 semitones	7 semitones
transition type	detached	glissando	legato

These attributes and their values would indicate two or three clusters, because of their number of possible values. At best all transitions which are grouped to one cluster share the same attribute value. Now the transition attribute values and their assigned clusters are compared.

First the clustering is done for the amplitude trajectories. In the following table one result is shown for a k-means++ clustering with $k = 2$ and the extracted parameters for a four point interpolation with spline curves:

Table 2.5.: Number of transitions for one attribute value and their cluster

dynamic	transitions in cluster 1	transitions in cluster 2
mp	117	55
ff	97	75

The table shows that 117 transitions with the dynamic *mp* are assigned to cluster 1 and 55 to the second cluster. This means that the clustering of the amplitude trajectories into two clusters does not reflect the separation from the attribute dynamic. Ideally the table should be diagonal, which means all transitions with the dynamic *mp* would be in cluster 1 and all transitions with the dynamic *ff* would be in cluster 2 to indicate this assumption. The clustering was also done with different models and different numbers of interpolation points. However for other attributes and numbers of clusters there is also no evidence that the clustering is similar to the separation of the transitions by their attributes.

Second clustering is done for f0 trajectories. The beziér curve and the spline parameters indicate a useful number of clusters from two to four clusters with the k-means algorithm. For the dynamic and Δf there is no evidence that these attributes could be associated to clusters, but transitions with type glissando could be assigned to one specific cluster. Table 2.6 shows the results for a k-means++ clustering with $k = 3$ clusters. The used data set was beziér curve parameters extracted out of all transitions with an upward direction from note 1 to note 2 and four interpolation points. $83,3\%$ of transitions with type glissando are in cluster 2, and only 5% with type legato and only 4% of type detached. The center of this cluster can be seen as an average glissando transition. In contrast to that, the two other clusters can not be separated by the transition type. This could be caused by overfitting. The algorithm used for the f0 trajectory does not detect gaps, so the trajectories for detached transitions can show a continuous trajectory even when there is no such behavior.

Table 2.6.: Three clusters and the number of associated transitions by type

transition type	transitions in cluster 1	transitions in cluster 2	transitions in cluster 3
legato	25	3	32
glissando	0	40	8
detached	15	2	32

The cluster centers itself are a set of parameters and a trajectory can be created with them. In figure 2.12 the trajectories of cluster centers for clusters which can be associated to the articulation glissando are presented. The beziér and the spline curves where results for a clustering with a data set created with a four point interpolation from transitions with an upward direction from note 1 to note 2.

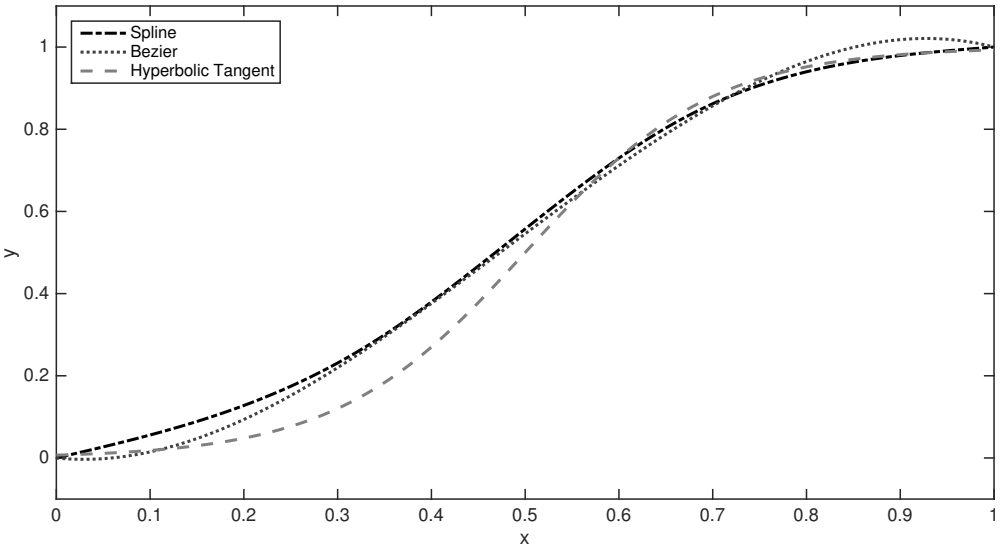

Figure 2.12.: Trajectories of cluster centers associated to glissando

The clustering showed that some attributes are correlated to the modeling parameters and some are not. The last example showed that for the search for a general description of a glissando transition modeling parameter sets can be found. But further evaluation is needed for finding a general model.

2.2.5. Evaluation

All the algorithms presented in this section must be compared and in terms of numerical accuracy one can be picked. A numeric measurement will be introduced for the algorithms but for further evaluation a user study is necessary. We need to take the complexity of the algorithms into account and the benefits we get from increasing computational effort.

A lower error limit needs to be found for which we can say that there is no notifiable difference. According to Kollmeier et al. [33] the just notifiable difference (JND) is about 3 Hz for frequencies below 500 Hz and about 0,6 % for frequencies above 1000 Hz for sine waves. For more complex tones inside the range of the fundamental frequency of speech (80 Hz - 500 Hz) the JND is about 1 Hz. But these values are measured with stationary tones, so for a tone with changing frequency these values could be different. With these values in mind one of the algorithms can be picked.

Error Measurement

An error measurement is needed to compare the results of the algorithms. A relative error is defined by Schwarz and Köckler [13]: $x \in \mathbb{R}$ and $\tilde{x}$ is the approximate value for x. If $x \neq 0$ the relative error ε is defined as:

$$\varepsilon := \frac{x - \tilde{x}}{x}. \tag{2.27}$$

To make the error comparable by normalization, we use the relative error formula from Engeln-Müllges et al. [27]:

$$\varepsilon_x := \frac{|x - \tilde{x}|}{|x|}. \tag{2.28}$$

Another error is the absolute error, also described by Engeln-Müllges et al. [27]:

$$\delta_x = |x - \tilde{x}|. \tag{2.29}$$

In the scope of this thesis we are going to approximate vectors $x_i = (x_0, x_1, \cdots, x_{n-1}, x_n)$ of length $n, x_i \in \mathbb{R}, n \in \mathbb{N}$ with the approximate vector $\tilde{x}_i = (\tilde{x}_0, \tilde{x}_1, \cdots, \tilde{x}_{n-1}, \tilde{x}_n)$ of length $n, \tilde{x}_i \in \mathbb{R}, n \in \mathbb{N}$. We define a mean absolute error for vector approximation:

$$\bar{\delta}_x := \frac{1}{n} \sum_{i=0}^{n-1} |x_i - \tilde{x_i}|. \tag{2.30}$$

For more information about the error we use the standard deviation of the error:

$$\sigma := \sqrt{\frac{1}{n} \sum_{i=0}^{n-1} (\delta_x - \bar{\delta}_x)^2}. \tag{2.31}$$

All of these indicators can also be described in percent:

$$\delta_{\%} = \delta \cdot 100 \tag{2.32}$$

and

$$\sigma_{\%} = \sigma \cdot 100. \tag{2.33}$$

A relative error works good if x and $\tilde{x}$ have high values, but if the values are around 0, the relative error can quickly rise and consecutively fails as a comparable error measure. For example: $x_i = (345, 350, 500)$ and $\tilde{x}_i = (340, 345, 495)$. The absolute error is then 5 for each pair of x_i and $\tilde{x}_i$. The relative error is then $\varepsilon_i = (0.0145, 0.0143, 0.01)$. In this case the absolute error is the same for every pair but the relative error varies. In this example the variation is not that high, but for $x_i = (0.1, 0.2, 0.4)$ and $\tilde{x}_i = (0.05, 0.25, 0.35)$ the relative error is very high and varies even if the absolute error is the same: $\varepsilon_i = (0.5, 0.25, 0.125)$. The absolute error is chosen. In addition the error is comparable if the data set is normalized, which means x and y - values are transformed linear to values between 0 and 1.
With theses formulas for error description we have a tool to measure the accuracy of the interpolation algorithms.

Fundamental frequency

First the results of the interpolation of the f0 trajectories is analyzed. All the glissando transitions are used and the parameters of the different models are extracted. In figure 2.13 one trajectory and the corresponding curves for each model are shown. This is one example of a glissando transition. Apparently, the exponential curve is the worst approximation in this case with a mean absolute error $\bar{\delta}_x$ of 0.21. In this case the normalization from 0 to 1 equals a frequency interval of 97.7 Hz. The mean absolute error can also be describe in Hz by multiplying the value with the interval and in table 2.7 the errors for all four models are shown.

Table 2.7.: Mean absolute error for models for one transition

model	normalized $\bar{\delta}_x$	$\bar{\delta}_x$ [Hz]
exponential smoothing	0.21	21.27
hyperbolic tangent	0.041	3.99
beziér	0.052	5.01
spline	0.32	3.14

In this example the spline approximation performs best but only with a small difference compared to the hyperbolic tangent curve. Next this error measurement is done for all the f0 trajectories of the glissando transitions. In table 2.8 the results are listed. The two models exponential smoothing and hyperbolic tangent do not depend on the number of interpolation points so there is only one row for each model. The error for the exponential smoothing curve is too high to use it as a model, so this model will be disregarded.

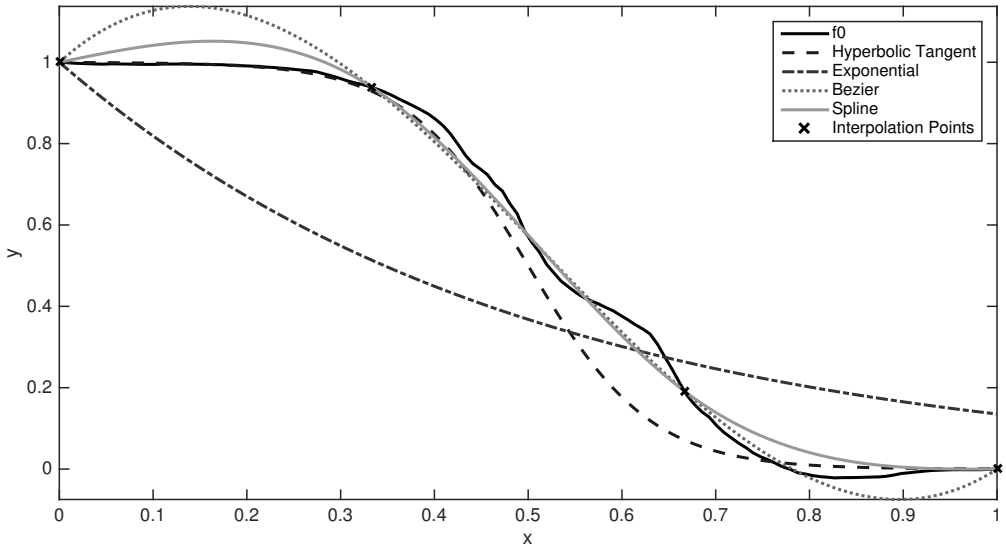

Figure 2.13.: f0 trajectory and interpolated curves for four interpolation points

Table 2.8.: Mean absolute error results for all glissando transitions

model	interpolation points	mean of normalized $\bar{\delta}_x$	mean of $\bar{\delta}_x$ [Hz]
exponential smoothing	-	0.185	34.60
hyperbolic tangent	-	0.083	15.38
beziér	4	0.0539	10.21
beziér	5	0.0394	7.26
beziér	6	0.0358	6.61
beziér	7	0.0311	5.39
beziér	8	0.0325	5.74
beziér	9	0.0377	6.63
beziér	10	0.0379	6.67
beziér	11	0.0594	10.22
beziér	12	0.0805	13.73
spline	4	0.0387	7.24
spline	5	0.0272	5.04
spline	6	0.0205	3.66
spline	7	0.0163	2.93
spline	8	0.0145	2.59
spline	9	0.0119	2.16
spline	10	0.0109	1.98
spline	11	0.0096	1.72
spline	12	0.0094	1.67

The results also confirm that the exponential smoothing is a bad choice for a transition model. The hyperbolic tangent does not give the best results in average but as we can see in figure 2.13, this curve could still be a good choice for synthesis due to its simple calculation. The spline curves lead to better results the he more points are used for interpolation. In contrast the beziér curves have an optimum number of interpolation points in terms of the mean value of $\bar{\delta}_x$. In this case seven points give the best results. In figure 2.14 the interpolation of a trajectory with 12 interpolation points is shown. The beziér curve in the graph shows the problem with increasing the number of points used for interpolation. By increasing the number of points the polynomial degree also increases, which leads to oscillation at the beginning and the end of the curve. This phenomenon was first described by Runge [34]. The spline curves do not show this behavior, because these curves are defined piece wise by third order polynomials and with an increasing number of points the number of pieces / polynomials increases and not the polynomial degree.

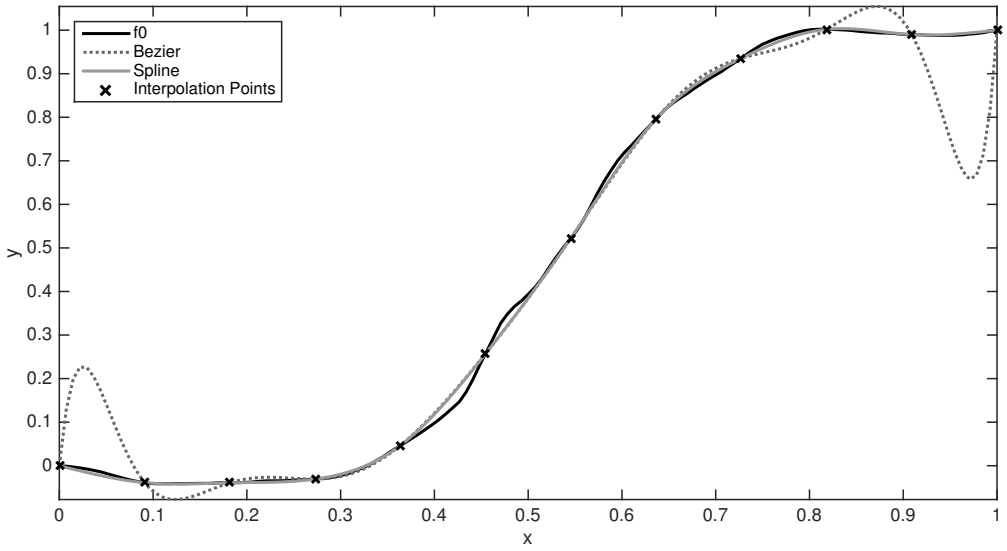

Figure 2.14.: f0 trajectory and interpolated curves for twelve interpolation points

The evaluation shows that with clustering a general description of glissando transitions can be found. The error evaluation suggests that exponential smoothing performs bad and should not be further regarded. The other models will be used in a real-time implementation, explained in the next chapter. While the hyperbolic tangent shows results that are not that promising the computation is rather simple compared to the other two models. The beziér curves show that polynomial interpolation with increasing polynomial degree is suboptimal, while the spline curves, which also use polynomials, offer the solution to this problem. The implementation and further testing in a real-time system can then show which model is usable for synthesis and an actually playable synthesizer.

Volume

The same error measurement was conducted with the volume trajectories of the glissando transition. The results show the same tendencies as for the f0 trajectories, except that the exponential smoothing model and the hyperbolic tangent model do not produce any curves that can be used for synthesis. This is due to the shape of the trajectories. There are additional maxima and minima, which can not be approximate by these two models. In contrast to the fundamental frequency no general description or an average behavior could be observed. In this case the advantages of the beziér and the spline curves are shown, which can approximate a lot of different curve forms, because for these models there is no previous assumption about the curve itself while the hyperbolic tangent function is limited to a small range of curves.

3. Synthesis

In this chapter the used tools and the framework developed for a MIDI controlled real-time synthesis are described, which is later deployed for a user study to evaluate the models.

3.1. Requirements

The different modeling algorithms described in the previous chapter should be playable and adjustable in real-time. Controlling the synthesizer should be realized with MIDI[1]. The developed models itself do not produce any audible output so a simple waveform synthesis and a volume control is needed, whose fundamental frequency is then controlled by the different algorithms. A sinusoidal waveform oscillator whose amplitude is controlled by an ADSR (attack, decay, sustain and release) envelope is suitable for the purpose of evaluation of the fundamental frequency change. The amplitude envelope is needed to avoid unwanted clicks in the output signal and is described by Puckette [4]. More complex synthesis could distract participants in a user study. Other features of a synthesizer like a filter or a low frequency oscillator for modulation are purposely left out to keep the synthesizer simple for the user study.

To ensure the real-time behavior of the synthesis the computational complexity should be kept as low as possible. The audio processing must be blockwise and triggered periodically to ensure an error free output. The time needed for the calculation must never exceed the time of one block. For example if the block size is 128 samples and the sampling frequency is 48 kHz the theoretic maximum time is 2,66 ms. If this time is over and the processing is not finished the output is interrupted and an audible noise is produced. Every block new incoming MIDI messages should be taken into account and change the calculation of the synthesis.

The synthesis should be monophonic to reproduce the behavior of the analyzed note transitions. Polyphony is not addressed here because the continuous frequency change from for example three different notes to the next three notes would involve an algorithm to pick which of the starting notes is linked to the ending notes to create three different trajectories for a smooth frequency transition. This is possible but in the context of the evaluation of the models and not an evaluation of an algorithm for polyphonic transitions

[1] Musical Instrument Digital Interface (MIDI) is a protocol widely used for control message for electronic instruments defined by The MIDI Manufacturers Association [35].

this should not be implemented here.

A model for the transition should control the frequency of the oscillator for every output sample. With this requirement first the oscillator must be capable of handling a sample wise frequency input and the trajectory must produce a frequency output every sample. The frequency control of the oscillator should be separate from the oscillator itself. There must be a module responsible for the frequency of the oscillator which is controlled by the incoming midi data. Additionally this module must then use one model for the calculation of a continuous frequency change. Incoming midi note values should change the frequency and midi control values change the parameters of the corresponding transition model. If a new midi note value is received a trajectory from the first to the second note is created. When the transition is over the frequency then is constant until another new midi note value is received. The currently used model should as well be controlled via midi. With this description of the oscillator and its frequency control there is a continuous audio output with a constant volume.

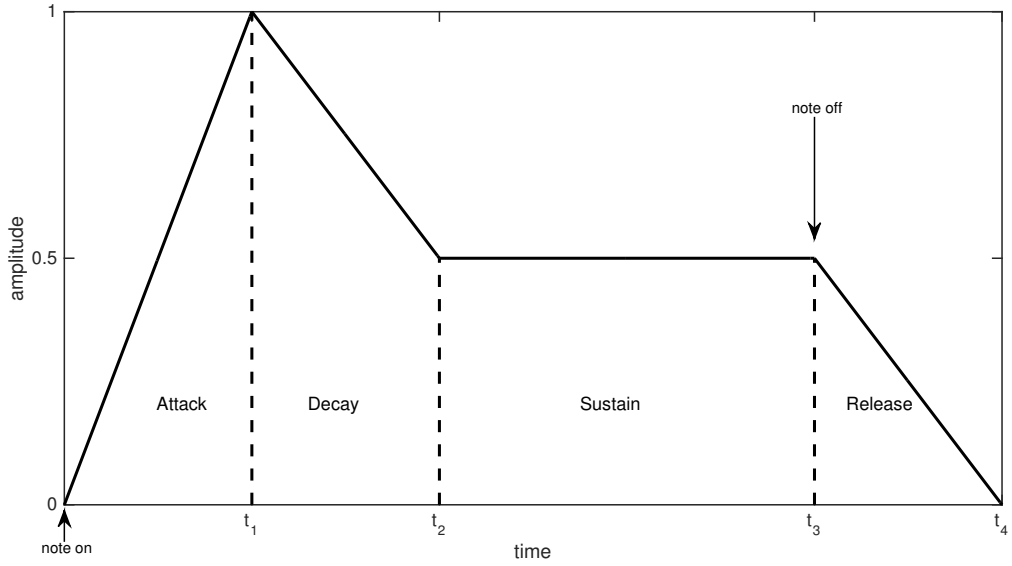

Figure 3.1.: ADSR envelope

The output volume should be controlled by the ADSR envelope which receives midi note on and note off messages. In figure 3.1 an ADSR envelope is shown. The parameters are the attack time (t_1), the decay time ($t_2 - t_1$), the release level and the release time ($t_4 - t_3$). When a note on message is received the attack phase starts and a linear ramp is created until the maximum value, which is 1, is reached. After that the decay phase starts and a linear transition from the maximum value to the sustain value, which is 0.5 in this example. The sustain value is constant until a note off message is received. Afterwards the release phase starts and the amplitude transitions linearly from the sustain value to

0. The envelope must be capable of receiving new midi note on messages even if a note off message is still missing or the release phase is not over yet. In this case the current value should be used as a starting point for the new attack phase. All the parameters described above should be controllable via midi. Most of the parameters described are part of the MIDI standard and the midi control values that should be used are shown in the following table:

Table 3.1.: MIDI control values for envelope parameters

parameter	midi control value
attack time	73
decay time	75
sustain level	70(not in standard)
release time	72

With these requirements a suitable hardware should be picked and a software should be developed.

3.2. Hardware

The Raspberry Pi by the Raspberry Pi Foundation [36] is a small computer of the size of a credit card. More precise the Raspberry Pi 3 Model B Rev 1.2 is used. The computational power with its 4 core 1.2 GHz Processor should be sufficient for the synthesis. One major advantage of the Raspberry Pi is that it is also capable of running in a headless setup which means, that there is no need for a connected screen.

Tests with the on board audio output of the Raspberry Pi revealed that it is not suitable for a low-latency setup. Therefore an external USB audio interface is used. The choice was the Behringer U-Control UCA222 due to its compact size and low price. With this interface a sampling frequency of 48 kHz was used and a low processing block size of 128 samples could be realized.

The prerequisite MIDI input is provided by an external USB interface. Here the LOGILINK USB to MIDI Adapter is used.

3.3. Software

In this section the developed software and additional dependencies are described.

3.3.1. Framework

On the Raspberry Pi the operating system used is Raspbian GNU/Linux 9.1 (stretch), kernel version 4.9.59-v7+. No additional tuning of the operating system towards real-time was done. Newmarch [37] describes that Linux gives the possibility for a real-time kernel but this is not necessarily needed here. This could be used later if the overall latency, which is the time between pressing a key on a MIDI input device and the audio interface producing output, is too high. The advanced Linux sound architecture by the ALSA project [38] provides audio functionality and is part of the main line kernel in Linux. The program developed here could directly use the ALSA API but then no further connection to other programs which then could control the synthesizer would be possible. This is needed for the user study described in chapter 4. The JACK audio connection kit by Knoth et al. [39] offers the functionality to connect audio and MIDI between programs and offers the possibility for low latency audio processing. JACK is written in C++ so the program itself should also be written in this language to use the provided API. According to the requirement from the previous section the JACK API offers a function that is called synchronous and provides buffers where the current audio block can be written to and a continuous audio stream is ensured.

The program is written in C++ and the C++11 standard described by Stroustrup [40] is used. To ensure correct functionality of the implemented models unit tests are implemented. The Boost unit test library by Rozental and Enficiaud [41] is used for this purpose.

3.3.2. Implementation

A program for the evaluation of the modeling algorithms is developed. Therefore the main goal of the software is producing a continuous audio output which is controllable in real-time. C++ offers object oriented programming and so the different functions and modules of the program are separated into classes. All classes involved in the audio processing manage blockwise audio samples. The primary class that holds all the other objects needed for processing audio and which operates as an interface for all parameter changes is the `SingleVoiceContainer` class. An overview of this class and its members is given in figure 3.2.

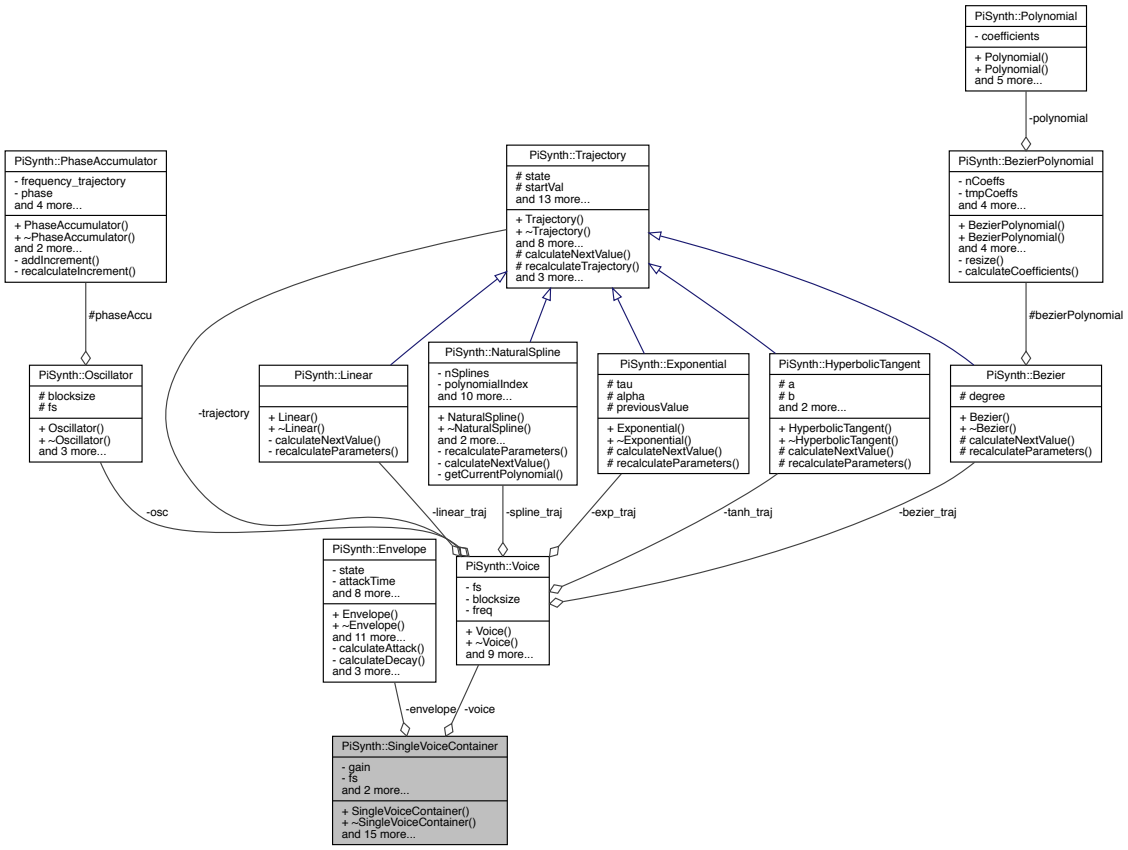

Figure 3.2.: Class diagram for SingleVoiceContainer class

Inside the class `Voice` first the `Trajectory` class creates a frequency vector. This vector depends on the currently selected model and its parameters as well as the current note value. With that vector the `Oscillator` class produces a sinusoidal output waveform. Then this output waveform is multiplied with the output of the `Envelope` class. All parameters for these classes can be controlled via MIDI. The handling and parsing of the MIDI messages is done in a separate class `MidiMapping`, which is not shown in the figure.

Inside the `MidiMapping` class the MIDI control values are assigned to the different parameters. For the trajectories the mapping of the MIDI control values used for the user study is shown in table 3.2. MIDI values, received as a number between 0 and 127 are mapped linearly to a value between the minimum and the maximum. These limits are chosen to fit best for the user study to reproduce resynthesized recordings out of the violin library. The limits can be extended if it is desired. One special case is the maximum of the hyperbolic tangent slope. If the value for the slope is greater than 0.23 the resulting trajectory has a jump in the beginning and the end, which can be seen in figure 2.9.

Table 3.2.: MIDI control values for trajectory parameters

parameter	midi control value	minimum value	maximum value
portamento / transition time	5	10 ms	4 s
model	75	0	127
hyperbolic tangent slope	52	0	0.23
Bézier parameter 1	53	0	1
Bézier parameter 2	54	0	1
Bézier parameter 3	55	0	1
Spline parameter 1	56	0	1
Spline parameter 2	61	0	1

Trajectories

Previously an overview about the synthesis was given without further implementation details. The formulas described in section 2.2.2 were implemented in MATLAB for testing and extracting parameters for the models. Solving the equation systems for the bézier and spline curves in MATLAB is straight forward. In the C++ standard library there is no function for matrix solving so a suitable library was needed. The Eigen library by Guennebaud et al. [42] was chosen for that purpose. With the help of unit tests it could be evaluated that the developed trajectories in C++ provide the same results as the MATLAB implementation. For the user mathematical functions like the hyperbolic tangent function the C++ standard library was used.

The developed program now offers a way of testing the different trajectory models in real-time. Further description of the C++ code is provided inside the code itself that is documented in the doxygen style developed by van Heesch [43].

3.3.3. Preparation for user study

For the user study a user interface is needed. For this purpose the programming language pure data (Pd) by Puckette [44] is used. The interface itself is later described in section 4.1. In Pd it should be possible to listen to audio files, receive and send midi files to the synthesizer to change for example the currently used transition model. In order to provide forwarding midi messages with Pd, the demon a2jmidid by Arnaudov [45] is used to connect the ALSA and JACK MIDI ports. Then the audio and MIDI signal flow can be set up according to figure 3.3.

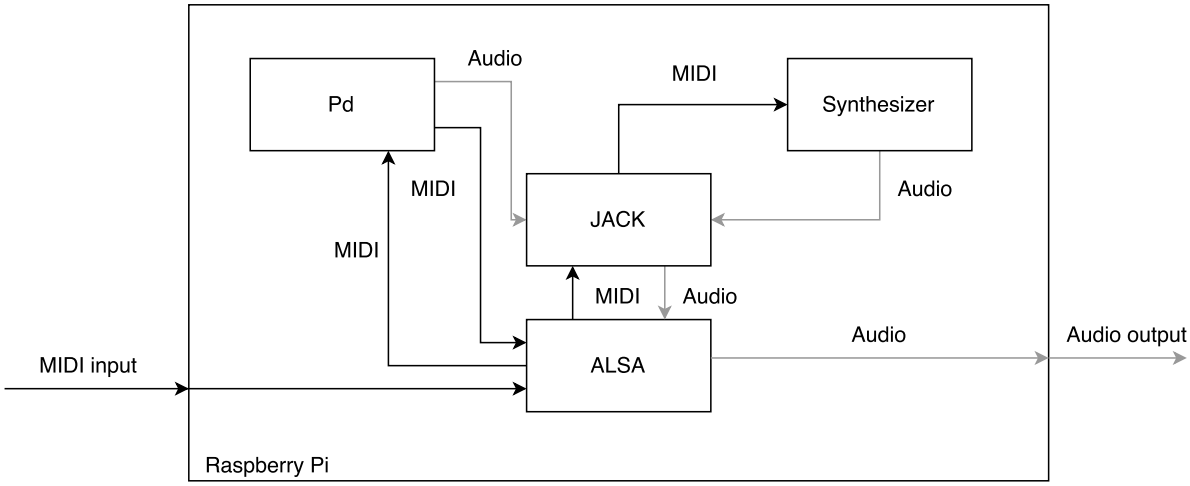

Figure 3.3.: Signal flow of audio and midi data

4. User Study

In this chapter the developed synthesis algorithms are evaluated in a user study. Questions examined could be: can listeners distinguish between different types of the nonlinear transitions? Are differences of transition parameters noticeable and do the parameters benefit the usage of the models in a synthesis context? However there is already a study done by Thyer and Mahar [46] that investigates the ability of listeners to discriminate different nonlinear frequency glides and the underlying physical mechanism in the ear. They showed that participants could distinguish between different nonlinear glides by their frequency change characteristics. This more general question should not be addressed in this study. The models developed here should be used in a real-time context for synthesis and therefore it would be interesting to know what models are useful for synthesis and how many parameters are effective in the context of reproducing the analyzed sounds. The question for this study is: can users reproduce the transition trajectories with the provided models?

There are many studies related to the study presented here. There are psychological studies focused on the perception of frequency glides like the ones by Thyer and Mahar [46], Madden and Fire [47] and Dooley and Moore [48]. However, these studies have a different approach because they do not propose a synthesis model and are focused on finding thresholds for the perception of frequency glides. Also related are studies which focus on the evaluation of digital music interfaces like the ones by Ghamsari et al. [49], Harrison and McPherson [50] and Konovalovs et al. [51]. But their focus is on the evaluation of digital music interfaces mostly developed by themselves and testing synthesis algorithms is not done in their studies. A task based study described by Wanderley and Orio [52] and O'modhrain [53] is often used for the evaluation of digital music interfaces can be adopted.

4.1. Design

In the previous chapter the synthesis of the trajectory models was described. Participants try to play with these models and test the usage of the parameters. To get the study into the context of the analyzed violin data set, participants try to replay the analyzed recordings. The Problem is that the developed models can only reproduce the trajectories and not the complete sound of the violin. For simplification the fundamental frequency trajectory is taken and a sinusoidal signal is created, further explanation of the stimuli is provided in chapter 4.2. These signal are used as a reference for the users to adjust the

parameters for each model.

One task consists of hearing a synthesized note transition and then tuning the parameters of one model and playing the two notes that the played sound is equally to the heard one. In each task only one model is tested and a suitable choice for the number of parameters per model is needed. The length of the transition is the only parameter which is the same for all of the models, but the reference note transition the participants are trying to reproduce defines this parameter. In the experiment the users do not change this parameter. The hyperbolic tangent model offers one additional parameter. With bézier curves and splines at least two parameters can be used. Theoretically a large number of parameters could be used for these two models, but testing with different numbers of parameters showed that too many parameters only complicate the task of reproducing the extracted trajectories. There are two parameters for the spline model and three parameters for the beziér curve presented to the participants. At least two parameters are needed for both models and more than three parameters do not lead to better approximation results.

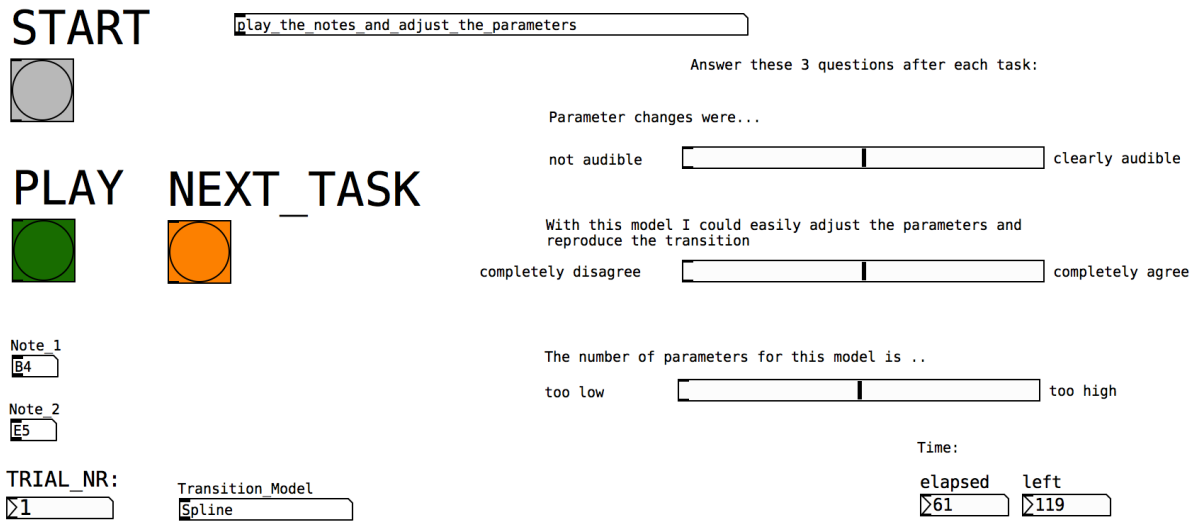

Figure 4.1.: PD user interface

Before the beginning of the task based study the participant gets a short description of the models and the effect of a parameter change, shown in appendix A.1, and five minutes to play with each model to get used to the parameters. In each task the user listens to a stimulus and information about the transition is shown: the model currently used, the starting and the ending note. Then they should set the parameters and play the two notes shown so that what they play sounds equal to the presented stimulus. They can listen to the stimuli as many times as they want. After solving the task they answer three questions about the model and the parameters. In figure 4.1 the user interface for the study is displayed. In the study the questions were in German, so here a translated version is shown. Overall the study consists of 21 tasks and after the participants are finished

they fill in a questionnaire with questions about their age, gender, education, their musical skills and general questions about the models, which can be found in appendix A.2.

4.2. Stimuli

As mentioned earlier using the recorded violin samples as a reference is not a good choice. With the extracted fundamental frequency trajectory a sinusoidal signal is created. Additionally a 500 ms sine wave with the frequency of the first note is added at the beginning and a 500 ms sine wave with the frequency of the second note is added at the end. Also a linear fade in at the beginning and a fade out at the end is used, each one with a length of 100 ms. The amplitude trajectories were not used for the stimuli for multiple reasons. Only one model should be tested at a time and so for each model the same seven stimuli were used which makes overall 21 tasks. Each task takes about two minutes so the overall experiment is around 40 minutes. Including the initial instructions and the final questionnaire the length of the study is at its maximum considering the average participants' ability to concentrate.

From the violin library seven samples where used which represent a good overall span of the attributes of the transitions. The fundamental frequency trajectories were extracted with the algorithm described in section 2.1.3. In following table the stimuli used for the study and their attributes are listed.

Table 4.1.: User study stimuli

filename	note 1	note 2	length	direction
TwoNote_DPA_18	A3	D4	380 ms	up
TwoNote_DPA_19	D4	A3	320 ms	down
TwoNote_DPA_65	E5	B4	400 ms	down
TwoNote_DPA_66	B4	E5	485 ms	up
TwoNote_DPA_113	D4	G4	300 ms	up
TwoNote_DPA_137	A4	D5	700 ms	up
TwoNote_DPA_186	E6	B5	550 ms	down

The optimal parameters for the three models for the seven presented stimuli were extracted. The absolute difference between these reference values and the parameters values configured by the participants in the study is used as an error measure. For the models with more than one parameter the mean value of the absolute difference is used to compare the error between the models. Normalization of the parameter value was also done to ensure better comparability of the error. This error is called parameter error in the discussion of the results. Additionally another error measurement is used. The curves created by the models with the user adjusted parameters are compared to the

extracted f0 trajectory with the mean absolute error as a measure. This error is called trajectory error in the discussion of the results.

4.3. Participants

The participants were recruited through the mailing list for students of the audio communication group. There were 15 participants in total and 14 of them were male and one female Their mean age was 27.4 years with a standard deviation of 5.6 years. They were all highly educated and had an university degree. The majority of the participants were musically skilled: 60 % played an instrument on a regular basis for more than 6 years and also 66.67 % had aural training for more than one year.

4.4. Results

The design of the study gives two independent variables, the model and the stimulus and several dependent variables: the three questions which are answered each task, the parameter error and the trajectory error. For the analysis a repeated measures analysis of variance (rmANOVA), described by Seltman [54] and Hair et al. [55] was used with the help of the software IBM SPSS. First the parameter error is analyzed.

Mauchly's test of sphericity indicated that the assumption of sphericity has been violated, $\chi^2 = 7.18, p = 0.028$, and therefore, a Greenhouse-geisser correction was used. Still there was a significant influence from the models on the parameter error, $F(1.40, 19.66) = 4.172$, $p < 0.05$, partial $\eta^2 = 0.23$. The results of the post-hoc test are shown in table 4.2.

Table 4.2.: Pairwise comparison of the models for parameter error

(I) Model	(J) Model	Mean difference (I-J)	Std. error	sig.	95% CI lower bound	95% CI upper bound
Hyperbolic tangent	Spline	-.066	.026	.024	-.122	-.101
Hyperbolic tangent	Beziér	-.026	.027	.346	-.084	.032
Spline	Beziér	.040	.014	.011	.011	.069

The pairwise comparison shows that there is a significant difference between the hyperbolic tangent and the spline model, as well as between the spline and the beziér model. In Table 4.3 the hyperbolic tangent model has the lowest mean value, but the difference

to the beziér model is not significant, therefore a general assumption about the model with the lowest parameter error can not be made.

Table 4.3.: Estimates of parameter error

Model	Mean	Std. error	95% CI lower bound	95% CI upper bound
Hyperbolic tangent	.126	.037	.048	.205
Spline	.193	.026	.137	.248
Beziér	.153	.018	.113	.192

The results of the parameter error showed that there is a difference between the models and how good the participants could adjust the parameters of the models. One drawback of this error is, that the mean absolute error of the resulting curve and the initially used f0 trajectory is not taken into account. For that reason the results of the trajectory error are now reviewed. This time the Mauchly's test of sphericity shows no significance, $\chi^2 = 2.883, p = 0.237$ and the model has a significant effect on the trajectory error as well, $F(2, 28) = 227.616$, $p < 0.05$, partial $\eta^2 = 0.942$. The pairwise comparison, shown in table 4.4, reveals a significant difference from the hyperbolic tangent model to other two models. Their comparison on the other hand shows no significant difference.

Table 4.4.: Pairwise comparison of the models for trajectory error

(I) Model	(J) Model	Mean difference (I-J)	Std. error	sig.	95% CI lower bound	95% CI upper bound
Hyperbolic tangent	Spline	.208	.014	.000	.179	.237
Hyperbolic tangent	Beziér	.218	.012	.000	.193	.242
Spline	Beziér	.010	.009	.281	-.009	.029

The estimated means in table 4.6 suggest that the hyperbolic tangent model is not a good choice compared to the other models for the recreation of the original f0 trajectories: the mean value is more than twice as much as the other two values.

Table 4.5.: Estimates of trajectory error

Model	Mean	Std. error	95% CI lower bound	95% CI upper bound
Hyperbolic tangent	.348	.008	.330	.366
Spline	.140	.018	.102	.179
Beziér	.130	.016	.096	.165

The results discussed so far are a measurement of the performance of the participants, but now the answers to the three questions asked during the study are analyzed. The answers could be adjusted on a slider which were then saved as a value between 0 and 100. The same analysis was done with these values as with the previously discussed errors. The first question was if the parameter changes were audible. Mauchly's test of sphericity shows no significance, $\chi^2 = 2.380, p = 0.304$ and the model has a significant effect on the answer, $F(2, 28) = 4.176$, $p < 0.05$, partial $\eta^2 = 0.230$. Pairwise comparison shows that there was no significant difference between the spline and the beziér model, while the comparison between spline and hyperbolic tangent as well as the comparison between beziér and hyperbolic tangent has been significant. The estimated means are shown in the following table:

Table 4.6.: Estimates for the answer of the first question: parameter changes were: 0 = not audible, 100 = clearly audible

Model	Mean	Std. error	95% CI lower bound	95% CI upper bound
Hyperbolic tangent	82.030	5.257	70.754	93.305
Spline	70.287	5.043	59.471	81.104
Beziér	70.780	5.324	59.361	81.198

The next question was how good the parameters could be adjusted to recreate the transition. For this question there was no significant difference between the models and the means were close to each other as shown in table 4.7. The answers to this questions show that the difficulty of the task was equally rated for all models.

Table 4.7.: Estimates for the answer of the second question: with this model I could easily adjust the parameters and reproduce the transition. 0 = completely disagree, 100 = completely agree

Model	Mean	Std. error	95% CI lower bound	95% CI upper bound
Hyperbolic tangent	70.224	5.686	58.029	82.420
Spline	69.843	5.694	57.630	82.055
Beziér	64.026	5.148	52.985	75.067

The last question addresses the number of parameters. Mauchly's test of sphericity shows no significance, $\chi^2 = 5.237, p = 0.073$. The model has a significant effect on the answer to this question, $F(2, 28) = 38.471$, $p < 0.05$, partial $\eta^2 = 0.733$. This time the pairwise comparisons of the models were significant for each combination. The estimated means are shown in table 4.8.

Table 4.8.: Estimates for the answer of the third question: the number of parameter for this model is: 0 = too low, 100 = to high

Model	Mean	Std. error	95% CI lower bound	95% CI upper bound
Hyperbolic tangent	32.855	3.522	25.300	40.410
Spline	49.750	1.323	46.913	52.587
Beziér	63.169	2.326	58.181	68.157

These results indicate that one parameter for adjustment is rated as insufficient, three parameters seem to be too many whilst two parameters are the best choice. In the final questionnaire the participants were asked which model they preferred. 53.5 % preferred the spline curves, 33.3 % hyperbolic tangent, 6.67 % beziér and 6.67 % had no preference. Overall the spline model was the preferred model by the users.

5. Conclusion

In the previous chapters different models were developed and evaluated with different measures. First a mean absolute error was used to examine which model an the number of points used for the interpolation performs best. Subsequently the models were implemented in a real-time synthesis framework before they were tested in a user study.

The spline curves lead to the lowest error with increasing number of interpolation points. The number of interpolation points is not necessarily restricted, but a reasonable number needs to be chosen. One problem with a large numbers can be overfitting. The validity of the curve which should be interpolated is depending on the algorithm used for the extraction of the curve. For the fundamental frequency extraction for example the algorithm used can introduce errors like detecting false frequencies or too much smoothing applied to the output. Also the possible use case of the models should be considered. More points for interpolation also lead to a higher computational complexity which should be considered in a real-time synthesis context.

In the user study the hyperbolic tangent curve with one parameter, the spline curve with two parameters and the beziér curves with three parameters were tested. Results showed that the error of the spline and the beziér curves compared to the reference fundamental frequency trajectory were the lowest. The difference between the two models was not significant. From that point of view no recommendation for an optimal model can be made. The participants preferred having two parameters for the adjustment of the curve so in a real-time synthesis context this would be a good choice. A parameter directly controls a point on the curve which then is interpolated by a model. The number of parameters is equal to the number of interpolation points minus the starting and the ending point of the curve. For other implementations instead of one parameter controlling one interpolation point, a parameter could also control more than one point.

Still the hyperbolic tangent function could be used for implementation especially when there is no additional parameter for the slope of the curve desired. The clustering showed that the average fundamental frequency curve is rather simple and a hyperbolic tangent curve with a fixed slope can be a good approximation. With the time of the transitions left as the only parameter this can be enough in a lot of use cases for the synthesis of glissando transitions.

Another aspect of the beziér and the spline curves that was not mentioned before is that they are capable of producing more complex curves. The extracted fundamental

frequency trajectories of the violin recordings were rather simple in contrast to what the models offer for synthesis. This opens a lot of possibility for sound design and the usage of the curves outside the context of re-synthesizing violin recordings. These curves could also be used in an oscillator when the starting and the beginning points of the curve are fixed to the same y-value. Then the resulting curve could be used as a controllable waveform in a wavetable oscillator.

Bibliography

[1] Russ, Martin (2009): *Sound Synthesis and Sampling*. Music Technology. Oxford: Focal Press.

[2] Jenkins, Mark (2007): *Analog Synthesizers*. Elsevier Ltd.

[3] Novation Digital Music Systems Ltd. (2010): "Ultranova User guide." URL `https://global.novationmusic.com/support/downloads/ultranova-user-guide`. [21.03.2018].

[4] Puckette, Miller (2007): *The theory and technique of electronic music*. World Scientific.

[5] Ikemiya, Y.; K. Yoshii; and K. Itoyama (2015): "Singing voice analysis and editing based on mutually dependent F0 estimation and source separation." In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pp. 574–578.

[6] Gómez, Emilia; Maarten Grachten; Xavier Amatriain; and Josep Llus Arcos (2003): "Melodic characterization of monophonic recordings for expressive tempo transformations." In: *Proceedings of Stockholm Music Acoustics Conference*.

[7] Dai, Jiajie; Matthias Mauch; and Simon Dixon (2015): "Analysis of intonation trajectories in solo singing." In: *Proceedings of the 16th ISMIR Conference*, vol. 421.

[8] von Coler, Henrik and Alexander Lerch (2014): "CMMSD: A Data Set for Note-Level Segmentation of Monophonic Music." In: *Audio Engineering Society Conference: 53rd International Conference: Semantic Audio*.

[9] Camacho, Arturo (2007): *SWIPE: A sawtooth waveform inspired pitch estimator for speech and music*. Ph.D. thesis, University of Florida.

[10] Digital Signal Processing Committee (1979): *Programs for Digital Signal Processing*. Piscataway, NJ, USA: IEEE Press.

[11] De Cheveigné, Alain and Hideki Kawahara (2002): "YIN, a fundamental frequency estimator for speech and music." In: *The Journal of the Acoustical Society of America*, 111(4), pp. 1917–1930.

[12] Scholz, Peter (2006): *Softwareentwicklung Eingebetteter Systeme*. Xpert. press Series. Physica-Verlag.

[13] Schwarz, H.R. and N. Köckler (2011): *Numerische Mathematik*. Vieweg+Teubner Verlag.

[14] Papula, L. (2009): *Mathematische Formelsammlung: für Ingenieure und Naturwissenschaftler*. Lothar Papula. Vieweg+Teubner Verlag.

[15] Brown, Robert Goodell (1963): *Smoothing, forecasting and prediction of discrete time series*. Prentice-Hall international series in management. Prentice-Hall.

[16] Zölzer, U.; M. Bossert; and N. Fliege (2005): *Digitale Audiosignalverarbeitung*. Informationstechnik. Vieweg+Teubner Verlag.

[17] Unser, Michael (1999): "Splines: A perfect fit for signal and image processing." In: *IEEE Signal processing magazine*, 16(6), pp. 22–38.

[18] Schweizer, Wolfgang (2016): *MATLAB kompakt*. De Gruyter Studium. De Gruyter.

[19] Ardaillon, Luc; Gilles Degottex; and Axel Roebel (2015): "A multi-layer F0 model for singing voice synthesis using a B-spline representation with intuitive controls." In: *Interspeech 2015*.

[20] Barbot, Nelly; Olivier Boëffard; and Damien Lolive (2005): "F0 stylisation with a free-knot b-spline model and simulated-annealing optimization." In: *Proceedings of the 9th European Conference on Speech Communication and Technology (Eurospeech)*.

[21] Hahn, Henrik; Axel Röbel; Juan José Burred; and Stefan Weinzierl (2010): "Source-filter model for quasi-harmonic instruments." In: *Digital Audio Effects (DAFx)*. pp. 1–1.

[22] Hahn, Henrik and Axel Röbel (2013): "Extended Source-Filter Model for Harmonic Instruments for Expressive Control of Sound Synthesis and Transformation." In: *Proc. of the 16th Int. Conference on Digital Audio Effects (DAFx-13)*. Maynooth, Ireland, p. 1.

[23] Lolive, Damien; Nelly Barbot; and Olivier Boëffard (2006): "Comparing b-spline and spline models for f0 modelling." In: *Lecture notes in computer science*, 4188, p. 423.

[24] Lolive, Damien; Nelly Barbot; and Olivier Boeffard (2006): "Melodic contour estimation with B-spline models using a MDL criterion." In: *Proceedings of the 11th International Conference on Speech and Computer (SPECOM)*. pp. 333–338.

[25] Lolive, Damien; Nelly Barbot; and Olivier Boeffard (2010): "B-spline model order selection with optimal MDL criterion applied to speech fundamental frequency stylization." In: *IEEE Journal of Selected Topics in Signal Processing*, 4(3), pp. 571–581.

[26] Robel, A (2006): "Adaptive additive modeling with continuous parameter trajectories." In: *IEEE Transactions on Audio, Speech, and Language Processing*, 14(4), pp. 1440–1453.

[27] Engeln-Müllges, Gisela; Klaus Niederdrenk; and Reinhard Wodicka (2011): *Numerik-Algorithmen*. Xpert. press Series. Springer Berlin Heidelberg.

[28] Battey, Bret (2004): "Bézier Spline Modeling of Pitch-Continuous Melodic Expression and Ornamentation." In: *Computer Music Journal*, 28, pp. 25–39.

[29] Xu, Rui and Donald Wunsch (2005): "Survey of clustering algorithms." In: *IEEE Transactions on neural networks*, 16(3), pp. 645–678.

[30] Lloyd, Stuart (1982): "Least squares quantization in PCM." In: *IEEE transactions on information theory*, 28(2), pp. 129–137.

[31] Arthur, David and Sergei Vassilvitskii (2007): "k-means++: The advantages of careful seeding." In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, pp. 1027–1035.

[32] Rousseeuw, Peter J. (1987): "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis." In: *Journal of Computational and Applied Mathematics*, 20, pp. 53 – 65.

[33] Kollmeier, Birger; Thomas Brand; and Bernd Meyer (2008): "Perception of Speech and Sound." In: *Springer Handbook of Speech Processing*. Springer Berlin Heidelberg, pp. 61–82.

[34] Runge, Carl (1901): "Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten." In: *Zeitschrift für Mathematik und Physik*, (Bd. 46), pp. 224–243.

[35] The MIDI Manufacturers Association (1996): "The Complete MIDI 1.0 Detailed Specification."

[36] Raspberry Pi Foundation (2018): "Raspberry Pi Documentation." URL `http://www.raspberrypi.org/documentation/`. [21.03.2018].

[37] Newmarch, Jan (2017): *Linux Sound Programming*. Springer.

[38] ALSA project (2018): "Advanced Linux Sound Architecture." URL `http://www.alsa-project.org`. [21.03.2018].

[39] Knoth, Adrian; Filipe Coelho; Nedko Arnaudov; Stephane Letz; and Robin Gareus (2018): "JACK Audio Connection Kit." URL `http://www.http://jackaudio.org/`. [21.03.2018].

[40] Stroustrup, Bjarne (2013): *The C++ programming language*. Pearson Education.

[41] Rozental, Gennadiy and Raffi Enficiaud (2017): "Boost Unit Test Library." URL `http://www.boost.org/libs/test`. [21.03.2018].

[42] Guennebaud, Gaël; Benoît Jacob; et al. (2010): "Eigen v3." URL `http://eigen.tuxfamily.org`. [21.03.2018].

[43] van Heesch, Dimitri (2016): "Doxygen." URL `http://www.doxygen.org`. [21.03.2018].

[44] Puckette, Miller (2017): "Pure data." URL `http://msp.ucsd.edu/software.html`. [21.03.2018].

[45] Arnaudov, Nedko (2012): "a2jmidid." URL `https://packages.ubuntu.com/de/trusty/a2jmidid`. [21.03.2018].

[46] Thyer, Nick and Doug Mahar (2006): "Discrimination of nonlinear frequency glides." In: *The Journal of the Acoustical Society of America*, 119(5), pp. 2929–2936.

[47] Madden, John P and Kevin M Fire (1997): "Detection and discrimination of frequency glides as a function of direction, duration, frequency span, and center frequency." In: *The Journal of the Acoustical Society of America*, 102(5), pp. 2920–2924.

[48] Dooley, Gary J and Brian CJ Moore (1988): "Detection of linear frequency glides as a function of frequency and duration." In: *The Journal of the Acoustical Society of America*, 84(6), pp. 2045–2057.

[49] Ghamsari, Mahtab; Amandine Pras; and MM Wanderley (2013): "Combining musical tasks and improvisation in evaluating novel digital musical instruments." In: *Proceedings of the International Symposium on Computer Music Multidisciplinary Research*. pp. 506–515.

[50] Harrison, Jacob and Andrew McPherson (2017): "An Adapted Bass Guitar for One-Handed Playing." In: *Proceedings of the International Conference on New Interfaces for Musical Expression*. Copenhagen, Denmark: Aalborg University Copenhagen, pp. 507–508.

[51] Konovalovs, Kristians; Jelizaveta Zovnercuka; Ali Adjorlu; and Daniel Overholt (2017): "A Wearable Foot-mounted / Instrument-mounted Effect Controller: Design and Evaluation." In: *Proceedings of the International Conference on New Interfaces for Musical Expression*. Copenhagen, Denmark: Aalborg University Copenhagen, pp. 354–357.

[52] Wanderley, Marcelo Mortensen and Nicola Orio (2002): "Evaluation of input devices for musical expression: Borrowing tools from hci." In: *Computer Music Journal*, 26(3), pp. 62–76.

[53] O'modhrain, Sile (2011): "A framework for the evaluation of digital musical instruments." In: *Computer Music Journal*, 35(1), pp. 28–42.

[54] Seltman, Howard J (2012): *Experimental design and analysis*. Carnegie Mellon University, Pittsburgh.

[55] Hair, Joseph F; et al. (2010): *Multivariate Data Analysis*. Prentice Hall.

List of Figures

List of Tables

A. User Study

The user study was made in german and therefore the instructions and the questionnaire were in german.

A.1. Instructions

Infotext für Teilnehmer

In diesem Versuch sollen unterschiedliche Algorithmen zur Erzeugung von kontinuierlichen Notenübergängen von zwei Noten untersucht werden. Zwischen zwei aufeinanderfolgenden Tönen wird ein kontinuierlicher Verlauf der Tonhöhe modelliert. Dafür gibt es 3 unterschiedliche Modelle mit jeweils 1-3 Parametern. Mit diesen Parametern wird der Kurvenverlauf des Grundtons von einer Note zur Anderen verändert. Zuerst könnt ihr euch für 5 Minuten im freien Spiel mit dem Keyboard und den Modellen sowie deren Parametern vertraut machen. Um kontinuierliche Übergange zu spielen, muss die erste Taste auf dem Keyboard noch gehalten sein, wenn ihr eine zweite Taste drückt.

Versuchsanweisung

Ihr werdet eine Tonfolge von 2 Tönen mit einem kontinuierlichen Übergang hören. Für jede Tonfolge wird euch ein Übergangsmodell zur Verfügung stehen. Dann sollt ihr die Parameter des Modells mit Hilfe der Fader auf dem Midikeyboard so einstellen, dass es euch möglich ist, den Übergang identisch zum abgespielten Übergang selbst zu spielen. Den gehörten Übergang könnt ihr so oft ihr wollt wieder hören indem ihr *Play* drückt. Ihr müsst selbst die 2 Noten auf dem Keyboard anschlagen. Wenn ihr der Meinung seid, dass ihr den Übergang so gespielt habt, dass er dem Gehörten gleicht, drückt auf den *Next_Task* Button. (Um kontinuierliche Übergange spielen zu können, muss der zweite Ton gespielt werden, während der erste noch gedrückt ist.)

A.2. Questionnaire

Fragen zu den Modellen

Ich hatte Schwierigkeiten beim Einstellen der Parameter.

	-3	-2	-1	0	1	2	3	
Ablehnung	☐	☐	☐	☐	☐	☐	☐	Zustimmung

Ich konnte einen deutlichen klanglichen Unterschied zwischen den Modellen feststellen.

	-3	-2	-1	0	1	2	3	
Ablehnung	☐	☐	☐	☐	☐	☐	☐	Zustimmung

Welches Modell hat dir am besten gefallen?

TanH (1 Parameter)	Bezier (3 Parameter)	Spline (2 Parameter)	Keines
☐	☐	☐	☐

Wieso?:

_____

_____

Personenbezogene Fragen

Alter: _____ Geschlecht: ☐ männlich ☐ weiblich ☐ keine Angabe

Beruf / Studiengang / Ausbildung: _____

Seit ... Jahren übe ich regelmäßig ein Instrument:

0	1	2	3	4-5	6-9	10 oder mehr
☐	☐	☐	☐	☐	☐	☐

Das Instrument, das ich am besten spiele ist _____.

Ich hatte Unterricht in Musiktheorie für

0	1	2	3	4-5	6-9	10 oder mehr	
☐	☐	☐	☐	☐	☐	☐	Jahre

_____

Ich hatte Unterricht in Gehörbildung für

0	1	2	3	4-5	6-9	10 oder mehr	
☐	☐	☐	☐	☐	☐	☐	Jahre

Die Fähigkeiten meines Gehörs sind:

	-3	-2	-1	0	1	2	3	
ich kann Intervalle nicht unterscheiden	☐	☐	☐	☐	☐	☐	☐	absolutes Gehör

---

B. Digital Ressource

This thesis comes with a CD. In the following listing a brief description of the folders and the content is made:

`Analysis/` This folder contains MATLAB code used for the analysis and the modeling, further description of the code can be found inside the readme.md file in the folder.

`Literature/` The used literature in .pdf format are inside this folder.

`Study/` This folder contains two subfolders:

- `PD/` the Software for the user study,
- `evaluation/` the Resulting dataset and scripts for evaluation. In readme.md instructions are described.

`Synthesis/` This folder contains the source code for the synthesizer, written in C++. The readme.md file in this folder describes setting up and running the program.

`Thesis/` The .bibtex, .tex file, figures and .pdf file for the document are inside this folder.