5 Quantisation

Rhythm together with melody is one of the basic elements in music. According to Longuet-Higgins ([LH76]) human listeners are much more sensitive to the perception of rhythm than to the perception of melody or pitch related features. Usually it is easier for trained and untrained listeners as well as for musicians to transcribe a rhythm, than details about heard intervals, harmonic changes or absolute pitch.

> "The term rhythm refers to the general sense of movement in time that characterizes our experience of music (Apel, 1972). Rhythm often refers to the organization of events in time, such that they combine perceptually into groups or induce a sense of meter (Cooper & Meyer, 1960; Lerdahl & Jackendoff, 1983). In this sense, rhythm is not an objective property of music, it is an experience that has both objective and subjective components. The experience of rhythm arises from the interaction of the various materials of music – pitch, intensity, timbre, and so forth – within the individual listener." [LK94]

In general the task of transcribing the rhythm of a performance is different from the previously described beat tracking or tempo detection issue. For estimating a tempo profile it is sufficient to infer score time positions for a set of dedicated anchor notes (*i.e.,* beats respectively clicks), for the rhythm transcription task score time positions and durations for *all* performance notes must be inferred. Because the possible time positions in a score are specified as fractions using a rather small set of possible denominators, a score represents a grid of discrete time positions. The resolution of the grid is context and style dependent, common scores often uses only a resolution of 1/16th notes but higher resolutions (in most cases binary) or non-standard resolutions for arbitrary tuplets can always occur and might also be correct. In the context of computer aided transcription the process of rhythm transcription is called (musical) *quantisation*. It is equivalent to transferring timing information from a continuous domain (absolute performance timing in seconds) into a discrete domain (metrical score timing).

Depending on the way an input file was created (*e.g.,* free performance recording, recording synchronous to a MIDI click, mechanical performance) a tempo detection must be performed before the quantisation; for quantisation a given tempo profile is required. Because the tempo detection module might create a tempo profile that indicates only the score time positions for anchor notes (*e.g.,* start of a measure or down beats) the score time information for notes between these anchor points can still be imprecise because of slight tempo fluctuations between two beats. These errors need now to be removed by the quantisation module. For example, an unquantised onset time at a score time position of 191/192 needs to be quantised to a reasonable grid position (*e.g.,* quarter or eighth notes) to be displayable in conventional music notation.

By choosing a too high resolution for the conversion from performance timing into score timing (*e.g.,* 1/128 of a whole note), the resulting scores will be correct displayable but very hard to read because of complex note durations. If the resolution is chosen too low (*e.g.,* a quarter note) the resulting score will contain overlap errors (*i.e.,* notes that did not overlap in the performance have overlapping durations in the score) and it will be very inaccurate compared to the performance data. An human transcriber would here – if possible – prefer 'simple' transcriptions over complex solutions. But he would intuitively detect the point where a correct, 'simple' transcription is not possible and a more complex solution needs to be chosen. For example, five equal spaced notes cannot be transcribed as eighth notes if they were played all during a single half note, here a more complex quintuplet needs to be chosen.

Musical quantisation can be divided into onset time quantisation and duration quantisation. It depends on the specific approach, if the durations of notes are treated and quantised as a *duration* or if they become quantised indirectly by quantising the respective offset time positions. In commercial sequencer

applications often also a quantisation approach – usually called swing- or groove-quantisation – is implemented. This type of quantisation is used for creating 'performance-like' MIDI files from quantised (mechanical) scores or from very inaccurate performance data. This performance simulation approach is not in the scope of this thesis and will therefore not be discussed in the following.

Similar to the already discussed issue of tempo detection and the area of voice separation the basic problem for quantisation is indicated by the fact that the translation from score to performance and also the inverse task of transcription are highly ambiguous relations and no injective functions. A single score can result in different but 'correct' performances (including different tempo profiles), and a single (non-mechanical) performance can be transcribed correctly by different scores of different complexity.
If we, for example, assume that a rhythmically complex performance is a mechanical performance (*e.g.,* played by a notation software system), then its rhythmical complexity should be preserve in a transcription. If we would assume that this performance was created by an untrained, human performer, a much simpler transcription should be preferred, because the complexity of the performance has likely be caused only by inexact playing. An untrained performer would probably not be able to play a complex piece very precisely. An human transcriber would here intuitively evaluate the performance quality of the input data. If the whole piece was played with poor accuracy, he would allow more deviations between the written and the performed data as for a very exact played performance. This means that if a piece is played with low accuracy the grid size for the quantisation will be increased intuitively by the transcriber. As shown in Chapter 4 the quality or accurateness of a performance can be inferred algorithmically from the performance data itself before starting tempo detection or quantisation. Different from other known approaches our system uses this accuracy information and will adapt some resolution and search window parameters. As already shown in previous sections (*e.g.,* Chapter 4), the position of the onset time of a note is usually much closer to the timing as specified in the score than the timing of the performed duration of a note. In [DH89] citation of [Pov77]: "... the note durations in performance can deviate by up to 50 percent from their metrical value." Therefore the expected and allowed amount of inaccuracy (*i.e.,* deviation between observed performance and inferred score) is higher for the duration of a note than for its onset time.

As shown in [Cam00b] and in own experiments the output quality of a quantisation algorithm also depends on a correct voice separation (see Chapter 2). If because of an incorrect voice separation the correct local context of a note is not available (previous, or successive notes have been attached to other voice streams) the onset time position and even more the duration of this note might be quantised incorrect. Figure 5.1 shows a correct quantisation for the last note in the first measure (note c") and a wrong quantisation if this note is attached incorrectly to the voice stream of the upper voice.

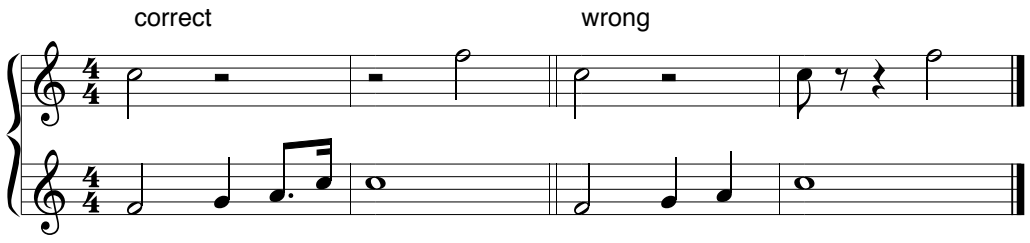


Figure 5.1: Possible effect of wrong voice separation to quantisation.

In the remainder of this chapter we first give an overview on existing quantisation approaches and then describe an approach developed in the context of this thesis that consists of a pattern-based part and a quantisation model which evaluates the local context of notes. All described approaches work on a single voice stream (including chords, but no overlapping notes) as input data.

5.1 Existing Approaches

In the following we give an introduction to musical quantisation. First we show and formally define the principles of the simplest type of quantisation called *grid quantisation*. Then we discuss the details of some more advanced approaches known from literature.

5.1.1 Grid Quantisation

This simplest form of musical quantisation maps (shifts) all performance time information to the closest time position of a fixed, equidistant grid of quantised score positions. For a performance time position t always the closest grid time position is chosen as quantised time position. This method – also called *hard* quantisation – is easy to calculate and implemented in most commercial sequencer and notation software products. Given an arbitrary, unquantised score time position t and an ordered set of equidistant score time grid positions G with

$$G = \{\ g_1, \ldots, g_{|G|} \mid g_i = (i-1) \cdot r, i \in \mathbb{N}\ \}, \tag{5.1}$$

where r is the grid resolution given as a beat duration in score time units. For an arbitrary unquantised time position t and a given grid G (with $g_{|G|} \geq t$), a quantisation function t_{qpos} returning the quantised time position for t can be defined as

$$t_{qpos}(t, G) = \arg\min_{g \in G}\{|g - t|\}. \tag{5.2}$$

If $|g_i - t| = |g_{i+1} - t|$ (because $t = (n + 1/2) \cdot r$) the earlier grid position g_i should be returned as result of $t_{qpos}(t, G)$. In general, depending on the actual implementation, also other selection strategies are possible (*e.g.,* latest grid position, avoid collisions between notes).
This approach requires that the grid size r needs to be selected in advance (*e.g.,* quavers) and kept fixed for the range of notes to which t_{qpos} should be applied. If the correct score position of an onset time or a duration is not an integer multiple of the grid resolution r, quantisation errors, such as overlapping notes or large shifts will occur, even if the resolution r is very small. Figure 5.2 shows the resulting errors if a group of eighth triplets (left) is quantised to an eighth note grid (centre) or to a smaller sixteenth note grid (right).

The worst case input for grid quantisation are pieces where binary and ternary (or higher prime number) subdivisions of note durations are mixed. Such files cannot be quantised correctly to an equidistant grid until all subdivisions are an integer multiple of a very small grid resolution, *i.e.,* the grid resolution must then be the greatest common divisor (gcd) of all types of durations of the performance. Unfortunately the resulting small grid duration increases the number of quantisation errors and therefore decrease the readability of the resulting score. Another general issue of this *hard* grid quantisation – beside the resolution issues – results from the fact that it is totally independent from note neighbourhood relations or context/history information of the quantisation process. The mathematical description of the single grid quantisation are a trivial case of the multi-grid quantisation described in Section 5.2.

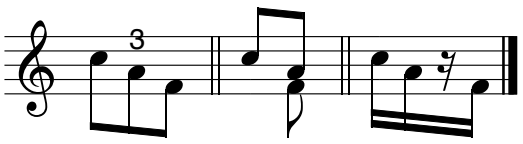


Figure 5.2: Quantisation errors will occur if, for example, a group of eighth triplets (left) is quantised to an eighth note grid (centre) or to a smaller sixteenth note grid (right).

5.1.2 A Rule-Based Approach

One of the first systematic approach to quantisation has been proposed by Longuet-Higgins in [LH76]. the author shows a rule-based approach for the transcription (including beat detection, quantisation and pitch spelling) of monophonic melodic lines of classical Western tonal music, using an hierarchical tree representation for the relations between durations. The tree hierarchy approach uses assumptions about cognitive effects of musical pattern. The model is based on three main assumptions:

- A listener collects rhythmical groups of notes as entities and builds up an hierarchical structure of this groups. The authors assume that the hierarchy can be represented in a tree structure using binary and ternary nodes only. Detected beats will be subdivided into two or three parts where each part might be subdivided recursively again.
- An initial tempo is available and needs not be inferred by the system.
- For human listeners the perception of rhythm is independent from the perception of melody.

The system tries to infer (bottom up) the perceived hierarchical structure of a performance. The approach does not evaluate any intensity information of the notes for tempo detection. This is justified by the assumption that an human listener is able to perceive beat and rhythm also from performances of instruments, such as organ or harpsichord, which do not allow the performer to change the intensity of notes because of the physical constraints of the instrument. We agree in the assumption that this is possible, but if intensity information is available it will stabilise and support the perception of rhythm by human listeners and suggest to evaluate this information if it is available. An elaborated version of the approach in [LH76] has been proposed by Longuet-Higgins and Lee in [LHL82]. Instead of the strong hierarchical model they here are using an approach based an the analysis of *relative durations* which are equivalent to IOI ratios as used in this thesis. Because this approach focusses more on tempo detection in general it is described in detail in Section 4.2.1.

In general hierarchical approaches (*e.g.,* [LH76, LHL82, LJ83]) are somehow limited to styles of music which are build along a strong hierarchical subdivision of metrical levels. They cannot be applied directly to styles of music that include dissonant rhythmic structures [Yes76, LK94], such as for example, contemporary Western Art music or Jazz where there exist non-hierarchical subdivisions of metrical levels (*e.g.,* binary and ternary subdivisions of crotches). Instead of a strong hierarchical view here a representation in form of layers or strata [Yes76, Mor99] might be more adequate. An own approach for quantisation by using a non-hierarchical grid is discussed Section 5.2.

5.1.3 The *Transcribe* System

In [PL93] Pressing and Lawrence propose a rule-based transcription system called *Transcribe* (see also Section 1.3). The input data is split into segments – the authors give no information about the size or positions of break points – and grid templates are compared to each section. For the segmentation into sections and the comparison to the grid positions the tempo must be inferred before. Pressing and Lawrence propose the usage of an autocorrelation approach, where it stays unclear if this can work for performances including tempo fluctuations. The grid templates are defined by the number of subdivisions in the range of 2^{-k} beats, with $-1 \leq k \leq 6$. Therefore a range from a double whole note down to a 64th note can be selected. The grid includes the directly specified time positions, their equivalent dotted positions ($1.5 \cdot 2^{-k}$), and a set of allowed tuplets up to 10:9. Each segment is then partitioned in a set of non-overlapping, user definable time windows and for each time window two error functions are calculated:

1. The absolute error E_{abs} given by a root mean square:

$$E_{abs} = \sqrt{\frac{1}{M}\sum_{j=1}^{M}\left(t_j - I_{N^*_{abs}}(t_j)\right)^2}, \tag{5.3}$$

   where the window time w consists of the time positions $t_1, \ldots, t_M$, N^*_{abs} is the best fitting grid template minimising E_{abs}, and $I_N(t_j)$ is the closest grid positions of N to t_j.

2. The inter-onset error E_{IO} given by

$$E_{IO} = \sqrt{\frac{1}{M-1}\sum_{j=2}^{M}\left(IOI(t_j) - I_{N^*_{IO}}(IOI(t_j))\right)^2}, \tag{5.4}$$

where $IOI(t_j) = t_j - t_{j-1}$ denotes the inter-onset interval for t_j, N^*_{IO} is the best fitting grid template minimizing E_{IO}, and $I_N(IOI(t_j))$ is the inter-onset interval of the closest grid positions to t_j in N.

If $N^*_{abs} \neq N^*_{IO}$ the template with the smallest error will be selected as total best pattern N^*. The overall error is defined as $E(N) = \min\{E_{IO}(N), E_{abs}(N)\}$. If the selection is $N^* = N^*_{abs}$ all time positions t_j will be moved to their closest grid position. If N^*_{IO} was chosen as the total best template three different strategies can be selected:

1. The first note of the time window is moved to the nearest grid position of N^*_{IO} and the inter-onset intervals of the remaining notes are quantised to the nearest integer multiple of the grid size of N^*_{IO}. The final score time positions then are the result of this inter-onset quantisation.

2. Equation 5.4 is modified to

$$E_{IO} = \sqrt{\frac{1}{M}\sum_{j=1}^{M}\left(IOI(t_j) - I_{N^*_{IO}}(IOI(t_j))\right)^2}, \tag{5.5}$$

   with $IOI(t_1) = t_1 - t'_M$ and t'_M is the latest time position of the previous time window. This can be done for every time window except the very first one and leads to connected windows and a "running inter-onset rms error". The time positions t_j will be quantised to their nearest grid position in N^*_{abs}.

3. The first time position (the starting time of the first note in the time window) will be shifted to the closest allowed tuplet position independent from N^*_{IO}. The notes $t_2, \ldots, t_M$ are processed as in the second option which preserves the inter-onset relations.

A special feature of this approach is the definition of four different pattern types with different rhythmical complexity:

- F filled – a performed note on every grid position
- R run – performed notes on all first k grid positions and and no performed notes on grid positions greater than k.
- U upbeat – no performed notes on grid positions $1, 2, \ldots, k$, performed notes on all remaing grid positions of pattern i, and a performed note on the first grid position of pattern $i + 1$.
- S syncopated – no performed note at the first and last grid position and performer notes on all other grid positions.

Because a grid position might be a nominal duration d, the dotted duration $1.5 \cdot d$, or a tuplet subdivision $(a/b) \cdot d$, this four types of pattern are enough to categories all possible selections.
By defining different classes of rhythmical complexity:

Class 1: F patterns only	*Class 2:* F & R patterns only
Class 3: F & R & U patterns only	*Class 4:* All pattern types

and selecting a single class for the quantisation process the output of the system can be limited to a certain rhythmical complexity. The authors also propose a parameter x (given in percent) for adjusting the influence of local context to the template selection process. If a time window i has selected a template N then a different template N' for the next window $i + 1$ will only be selected if

$$\frac{E(N) - E(N')}{E(N)} > \frac{x}{100}. \tag{5.6}$$

We assume that in an actual implementation this constraint must be refined to cover cases where $E(N)$ might be zero already.

5.1.4 A Connectionist Approach

In [DH89] Desain and Honing propose a connectionist approach for quantisation where a sequence of notes is represented as a network of connected 'cells', known in other areas as *interactive activation and constraint satisfaction networks* (see Figure 5.4). The goal is to reach a state of convergence between performance timing and score timing by several iterations. Desain and Honing distinguish between three types of cells:

1. basic cell – representing a note
2. sum cell
3. interaction cell – connecting any combination of two basic or sum cells

The interaction cells try to establish integer duration ratios between their connected cells. Only if the observed ratio is close to an integer ratio the connection cell will change the durations of connected basic cells to fit the exact integer ratio. If the observed is in between two integers – unclear which direction should be selected – the connection cell would not change the ratio of the connected cells. For the so-called *basic model* it is assumed that interaction cells are directly connected to two basic cells. The sum cells are not used for the basic model described in the following paragraph. The behaviour of interaction cells can be defined as a function $F(r)$ taking an observed IOI ratio as input and giving a 'desired' ratio as output:[1]

$$F(r) = ([r] - r) \cdot |2(r - \lfloor r \rfloor - 0.5)|^p \cdot [r]^d \tag{5.7}$$

$F(r)$ should be interpreted as the change of ratio for an observed ratio r. The parameter p is called the *peak* parameter – typically in range 2 to 12 – which adjust the width and height of the function peaks which result in the size of adjustment during a single iteration process. The decay parameter, d – typically in range -1 to -3 – adjusts the decay of the influence of higher ratios. See Figure 5.3 for a sample output of $F(r)$. During a single iteration the change of ratio $F(a/b)$ for two intervals a, b is

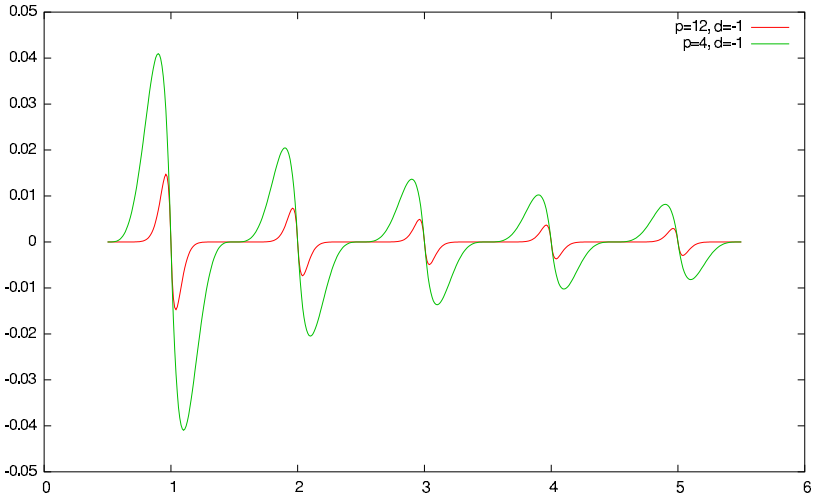


Figure 5.3: Sample output for $F(r)$ with different settings for the peak parameter p.

calculated and new inter-onset intervals $a' = a + \Delta$ and $b' = b - \Delta$ are calculated so that the interval sum is preserved ($a + b = a' + b'$):

$$\frac{a + \Delta}{b - \Delta} = \frac{a}{b} + F(\frac{a}{b}), \tag{5.8}$$

which results in

$$\Delta = \frac{b \cdot F(\frac{a}{b})}{1 + \frac{a}{b} + F(\frac{a}{b})}. \tag{5.9}$$

[1] The original text used a function called *entier* instead of the floor brackets.

The iteration is stopped if the change in duration for all basic cells is lower than a certain threshold l. It is obvious that not all successive notes of a piece will have integer multiple duration ratios. For example, an eighth note followed by a dotted eighth note results in an IOI ratio of 1.5. Therefore Desain and Honing introduce a *compound model*, for which the sum cells are used. Each sum cell is connected to two basic cells and an interaction cell. It sums the activation levels (inter-onset intervals) of the connected basic cells and interacts with the connected interaction cell. If the sum cell changes its value (triggered by an interaction cell) the change of value is given proportionally to the corresponding basic cells. The number of cells for a series of $n+1$ notes or n inter-onset intervals are n basic cells, $[(n+1)(n-2)/2]$ sum cells, and $[n(n^2-1)/6]$ interaction cells (see Figure 5.4). As also stated by the authors the behaviour

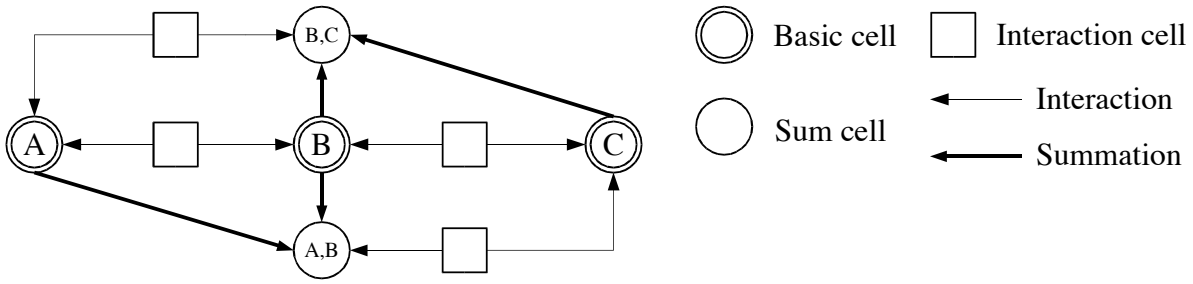


Figure 5.4: Compound connectionist net (copied from [DH89]).

of connectionist systems is difficult to study. They propose the usage of a *clamping* method where the states of all cells except one are fixed and the changes for that specific cell can be observed. By using this method it is possible calculate a function between an integer ratio of the 'free cell' and the so-called 'potential energy' of that ratio. If the potential energy is low the corresponding ratio fits the constraints given by the connected interaction cells more than a ratio with a high potential energy.

The approach can be implemented in two ways: with an asynchronous update strategy or with a synchronous update strategy. For the asynchronous implementation the authors propose to chose to use a random order for the evaluation of cells. Desain and Honing implemented a version with synchronous update in Common Lisp using a vector of inter-onset intervals as input. Unfortunately the authors give only some short outline about further research and potential of their approach. They did not evaluate the approach with 'real' performance data. In [Row01] a window-based real-time implementation (C++) of the approach is available and discussed. This implementation demonstrates how the connectionist net works for incoming events, but it cannot be used directly for processing complete MIDI files and comparing the output to other quantisation approaches.

The connectionist system can be somehow compared to a spring and rod model where the basic cells could be interpreted as the rods and the interaction cells the as springs (with discrete elongation positions). The elongation of the springs could be interpreted as the change between performance IOI ratio and *quantised IOI ratio*. Different from other approaches which focus on the quantisation of time positions and durations this connectionist approach quantises actually the inter-onset ratios (IOI ratios). The advantage here is that the quantisation of IOI ratios can be done without knowledge about a tempo profile. In best case the tempo profile can be induced by a quantisation of IOI ratios.

Roberts gives in [Rob96] some criticism to this connectionist approach because it includes some non connectionist features, such as hard-wired connections and cells with a complex functionality.

5.1.5 Vector Quantiser

Cemgil, Desain, and Kappen presented in [CDK00] the so-called vector quantiser. The proposed framework is based on Bayesian statistics and operates on short groups of onsets (code vectors). Each group of notes represents the sequences of notes between two successive 'beats' which must be inferred separately by a tempo tracking module before the quantisation can start. The 'beats' should be equal to the tactus level that an human listener perceives when listening to the input data. For their model the real score duration or distance between two beats is not evaluated because the approach works only on hierarchical

subdivisions of beats.[2] The complete series of 'beats' at time positions $t_i, \ldots, t_n$ here is called a *tempo track*.
The model does not explicitly evaluate or quantise the note durations, only the onset times (onsets) are investigated. Their model is based on the assumption that "... in western music tradition, notations are generated by recursive subdivisions of a whole note ...". These subdivisions can be represented in an hierarchical form using prime numbers for the division on each level. For example, a subdivision scheme $\mathcal{S} = [2]$ would divide the duration of a beat into two equal parts. A subdivision into four equal parts would be encoded as $\mathcal{S} = [2, 2]$. Because of the regular distance for the hierarchically defined grid, a subdivision scheme for dividing quarter beats into eighths **and** eighth triplets can only be defined by $\mathcal{S} = [3, 2]$ creating a regular grid of $1/24$ note durations!

The segments – representing the notes between two beats – can be seen as rhythmic patterns which are not perceived as a sequence of isolated notes but as a perceptual entity made of onsets. Therefore Cemgil *et al.* propose the quantisation of these groups over the quantisation of the single onsets. In mathematical terms this means that there exists a correlation between the score position of notes which is high if their (performed) distance is small.
One issue here might be the strict division into segments between two successive beats. If these beats are at the quarter note level – which is a very common tactus level for human listeners – the actually perceived pattern is usually longer than a single beat duration. By dividing the piece into 'beat-segments' the correlation between notes cannot be evaluated if some note have been played closely before and others closely after a beat.

By defining a prior probability $p(c)$ for a quantisation c (*i.e.,* a rhythm pattern) and the likelihood $p(t|c)$ for quantisation c for a given performance t it is possible to calculate the posterior $p(c|t)$ using the Bayesian formula:

$$p(Score|Performance) \propto p(Performance|Score) \times p(Score) \tag{5.10}$$
$$posterior \propto likelihood \times prior \tag{5.11}$$

Resulting in

$$p(c|t) \propto p(t|c)p(c) \tag{5.12}$$

which this is equivalent to a MAP[3] estimation problem; Cemgil *et al.* convert this maximization task into a minimization problem:

$$-\log p(c|t) \propto -\log p(t|c) + \log p(c) \tag{5.13}$$

which can be calculated more easily. The term $-\log p(c)$ gives a measure for the complexity of the selected rhythm pattern c and $-\log p(t|c)$ the distance between the performed onset times of t and the quantised onset times of c. The minimum of $-\log p(c|t)$ will then result in a quantisation c where the complexity of c and the distance between c and t is balanced. For $p(c)$, the prior of a specific c they propose to use

$$p(c) = \sum_{\mathcal{S}} p(c|\mathcal{S})p(\mathcal{S}), \tag{5.14}$$

with

$$p(\mathcal{S}) \propto e^{-\xi \sum_i w(s_i)}, \tag{5.15}$$

where w is a weight function defined by the authors, preferring simple subdivisions as shown in Table 5.1, and parameter ξ adjusts the probabilities of the different subdivision schemes of w.

s_i	2	3	5	7	11	13	17	o/w
$w(s_i)$	0	1	2	3	4	5	6	∞

Table 5.1: $w(s_i)$ (source [CDK00] p. 14)

By adjusting the correlation parameters to an uncorrelated behaviour the proposed vector quantiser works exactly like a grid quantiser, shifting onset times to their absolute nearest grid position. With other

[2] In the following a beat should be equal to the distance between two successive beats.
[3] Maximum a-posteriori

correlation parameters the behaviour can be changed between a pure grid quantiser and a quantisation with full correlation.

For the evaluation the authors did some experiments where ten musically educated and experienced human listeners should transcribe 91 four-note rhythmical patterns t_k, where the first and last note where always exactly on the beat and the second and third note on 'inexact' positions between the beats. This perception task resulted in 125 different notations (57 used only once). Using this data they estimated the posterior $p(c_j|t_k)$ for a transcription c_j of a performance t_k as

$$p(c_j|t_k) = \frac{n_k(c_j)}{\sum_j n_k(c_j)}, \tag{5.16}$$

where n_k gives the number of times where t_k was transcribed as c_j. This means that $p(c_j|t_k)$ is high if only a few different solutions c_j have been chosen for the transcription of t_k. With a second experiment – a production task – where the participants should perform their own notated rhythm patterns Cemgil *et al.* could calculate the mean and variance parameters for deviation between score and performance timing.

Unfortunately the authors provide no source code or running executable of their approach. Therefore it cannot be tested with 'real' performance data or other parameter settings. An disadvantage of the approach seems to be the strict hierarchical subdivision schemes resulting that can create only complete regular, equidistant grids. If only note durations of eighths, sixteenth, and eighth triplets should be used, this could only be represented by a scheme $\mathcal{S} = [2, 2, 3]$ which results in a regular grid ($\{n \cdot 1/48\}, n \in \mathbb{N}_0$) of note durations, much smaller than actually required ($\{n \cdot 1/12\} \cup \{n \cdot 1/8\}, n \in \mathbb{N}_0$). There also exist ambiguous schemes, such as $\mathcal{S}_1 = [2, 3]$ and $\mathcal{S}_2 = [3, 2]$ which could lead to different complexity measures for equal solutions.

An approach using explicit user definable or automatically learned patterns would overcome this issues. A pattern-based solution would also eliminate the problem with correlated note positions on different sides of beats and could also be used for an explicit quantisation of note durations. Different from the approach described in Section 5.3 the code vectors of Cemgil's approach are not given explicitly as note sequences. Instead they are defined in an hierarchical way with different numbers of subdivisions on each level. Therefore this approach might be seen as an implicit pattern quantisation model.

5.2 Multi-Grid Quantisation

For eliminating the triplet and binary note duration issue of the simple grid quantisation model shown in Section 5.1.1, a multi-grid quantisation can be used. For this type of quantisation the quantisation grid consists now of integer multiples of different resolutions r_i which might create a non-equidistant grid. Because of the typically higher density of grid points in this non-equidistant grid, inaccurate played onset times or durations would tend to be quantised to wrong grid position, if here just the absolute nearest grid position to an unquantised time position would be selected. We therefore propose a model for determining different attractions between unquantised time positions and the grid positions in their local neighbourhood. The basic principles and an implementation of the multi-grid model were proposed by us in [Kil96]. The following detailed, formal definition of this model and some distance measures (*e.g.,* Equation 5.21) are more elaborated than in [Kil96]. The functions shown in Section 5.2.4 (history-based quantisation) have been developed in the context of this thesis and were described in [Kil96].

As first step of this approach we define the multi-grid G_{multi} consisting of multiples of a limited set of note durations R:

$$R = \{\ r_1, \ldots, r_{|R|} \mid r_i > r_{i+1} > 0\ \}, \tag{5.17}$$

where r_i denotes any arbitrary quantised (*i.e.,* displayable) note duration. A multi grid G_{multi} can be defined by

$$G_{multi} = \bigcup_{i \in \mathbb{N}_0} i \cdot R \tag{5.18}$$

If R consists only of a single resolution entry ($|R| = 1$), the grid G_{multi} and all further calculations will be equal to the simple grid quantisation described in Section 5.1.1. The multi-grid set R can be seen as

related to the time position states $c_k \bmod 1$ proposed by Cemgil and Kappen in their tempo detection approach based on Monte Carlo sampling (see Section 4.2.2). If (and only if) there exists any $r_i \in R$ which is not an integer multiple of the smallest resolution $r_{|R|} \in R$, then G will be a non-equidistant grid. This is the case if, for example, the two smallest entries $r_{|R|}, r_{|R|-1} \in R$ are $1/12$ (eighth triplet) and $1/8$ (see Figure 5.5). It should be noted that the resulting non-equidistant grid is different from the strict hierarchical model for meter described by Lerdahl and Jackendoff in [LJ83] (see also Section 5.1.2).

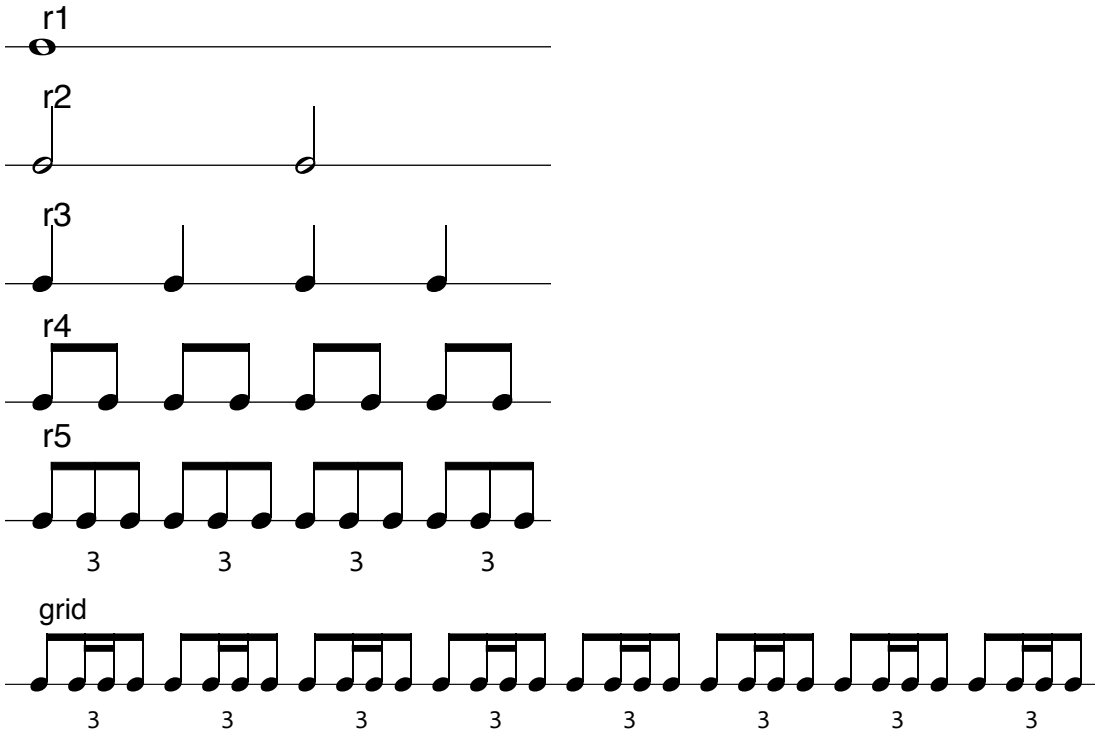


Figure 5.5: Set of resolutions ($R = \{1/1, 1/2, 1/4, 1/8, 1/12\}$) and the resulting non-equidistant grid. The same grid (with different weights, resulting in different attractions) could be obtained, for example, with $R = \{1/8, 1/12\}$

For the in the following described approach the quantisation of onset times and durations and can be processed in separate tasks. In the current implementation, first two most likely quantised positions for the onset time and then two most likely values for the quantised duration of all notes of the performance will be calculated. Both, the onset time and the duration module, work equally but can be used with different parameter settings *i.e.,* different grids for duration quantisation and onset times quantisation.

Now for each $r_i \in R$ a closest possible grid position $t_{cpos}(t, r_i)$ to an unquantised time position t – which can correspondent either to a note onset or a note duration – can be calculated by

$$t_{cpos}(t, r_i) = r_i \cdot (\arg \min_{a' \in \mathbb{N}_0} \{|t - a' \cdot r_i|\}), i = 1, 2, \ldots, |R|, r_i \in R. \tag{5.19}$$

If $|t - a \cdot r_i|$ is equal to $|t - (a+1) \cdot r_i|$ (because $t = (a + 1/2) \cdot r_i$) the earlier time position $a \cdot r_i$ will be selected as closest grid position for t and r_i. It is easy to see that $\forall\ r_i \in R : t_{qpos}(t, r_i) \in G_{multi}$. For each unquantised time position t and a grid resolutions set R there exists now a vector $T^t_{cpos} \subset G_{multi}$ of possible, closest quantised time positions with

$$T^t_{cpos} = \left(t_1, \ldots, t_{|R|} \mid t_i = t_{cpos}(t, r_i)\right). \tag{5.20}$$

In the following t_i should denote the i-th element of the vector T^t_{cpos}. For each quantised, grid score time position $t_i \in T^t_{cpos}$ the absolute distance $\delta(t, t_i) = t_i - t$ to the associated unquantised time position t can

be calculated. By using a Gaussian window function, the range of δ can be normalised to the interval $(0, 1]$:

$$p_\delta(t, t_i) = W_{Gauss}(t - t_i, \sigma) \tag{5.21}$$

According to Cemgil ([CKDH01]) a variance of $\sigma = 0.04s$ corresponds roughly to the spread of onsets from their mechanical means during performance of short rhythms.

All $t_i \in T^t_{cpos}$ with $\delta(t, t_i)$ outside a left or a right search window $lsWindow < 0 < rsWindow$ around t can be marked as *invalid* respective *valid* if inside the search window:

$$valid(t_i) = \begin{cases} 1, & \text{if } lsWindow \leq \delta(t, t_i) \leq rsWindow \\ 0, & \text{else} \end{cases} \tag{5.22}$$

If the size of a search windows w is larger than half of the smallest resolution $r_{|R|}$, it can be ensured that at least one $t_i \in T^t_{cpos}$ is a valid quantised time position for t:

$$w > 0.5 \cdot r_{|R|} \Longrightarrow t - w < t - t_{cpos}(t, t_{|R|}) < t + w \tag{5.23}$$

The adequate window size of the search windows and the parameter σ should be adapted to the overall *performance accuracy* which can be calculated in advance as shown in Section 4.3.5. For an high accuracy, σ and the search window size will be rather small, for low accuracy they might be increased. These parameter settings might also be different for the duration and onset time quantisation. Typically the search windows are larger and σ is higher for the duration quantisation than for onset time quantisation which gives respect to less accurate performance of durations compared to onset times. The size of the search windows for onset times and durations should be adjusted depending on the output of the accuracy analysis (see Section 4.3.5). If in the current implementation the accuracy is 100% (*i.e.,* a mechanical performance) all onset times and durations down to 1/64 notes will be converted exactly. If needed, here new small durations will be added to the duration and/or onset time grid. In the current implementation the default grids can be defined via initialisation files, see Section A.3 for details. If the accuracy is very low the algorithm might ask the user if existing small values should be removed from the two grids. To our knowledge such an adaptive, interactive strategy has not been proposed in the literature before.

In a simple implementation of the described multi-grid approach (including the trivial case of $|R| = 1$) now the time position $t_i \in T^t_{cpos}$ with the smallest absolute distance $\delta(t, t_i)$ would be chosen as quantised time position of t. It depends on the implementation if in ambiguous situations ($\delta(t, t_i) = \delta(t, t_j)$) t should be moved to the earlier or later grid position. Without a more advanced selection strategy as proposed in the following section, the multi-grid approach is still a hard quantisation to a closest grid position. This strategy of hard quantisation could still cause a high rate of errors – especially for non-mechanical files – because some grid positions in G_{multi} might have a very small distance. If assuming that the performance data is not machine generated and includes therefore inaccuracies, not always the closest grid position will be the correct quantised time position. As shown in the following it is possible to calculate some additional weight (or significance) information for each $t_i \in T^t_{cpos}$ which can be used for a more adequate selection strategy.

5.2.1 A Weighting Strategy

The selection strategy of the previously described *hard* quantisation can be improved by calculating a weight (or significance) for each time position t_i of a vector T^t_{cpos}. This weight should depend on the number of non-equal possible time positions in T^t_{cpos}. Because each element $t_i \in T^t_{cpos}$ represents a quantised time position $t_i = a_i \cdot r_i, r_i \in R$, some time positions t_i, t_j might be equal because $a_i \cdot r_i = a_j \cdot r_j, r_i, r_j \in R, i \neq j$. If a specific time position occurs more often in T^t_{cpos} it should get a higher weight than a time position that occurs less often. It should be noted that if the unquantised time position t is very close to a common integer multiple of all $r_i \in R$, then all $t_i \in T^t_{cpos}$ will be equal time positions. The number of different possible time positions in T^t_{cpos} can be evaluated mathematically by defining some utility functions:

$$equal(T^t_{cpos}, i, j) = \begin{cases} 1, & \text{if } t_i = t_j \wedge j \geq i \wedge \nexists\ i' < i \text{ with } t_{i'} = t_i \\ 0, & \text{else} \end{cases} \tag{5.24}$$

The *hard* constraint for the 1 case ensures that each possibility is counted only once so that

$$\sum_{i=1}^{|R|}\sum_{j=1}^{|R|} equal(T^t_{cpos}, i, j) = \sum_{i=1}^{|R|}\sum_{j=i}^{|R|} equal(T^t_{cpos}, i, j) = |R|. \tag{5.25}$$

(Please note that the start index of the inner sum is different on left ($j = 1$) and right side ($j = i$).) Because only the valid time positions of T^t_{cpos} inside the search window around t should be evaluated we define:

$$count(T^t_{cpos}, i) = valid(t_i) \cdot \sum_{j=i}^{|R|} equal(T^t_{cpos}, i, j) \tag{5.26}$$

$$cValid(T^t_{cpos}) = \sum_{i=1}^{|R|} valid(t_i) \tag{5.27}$$

It follows that

$$\sum_{i=1}^{|R|} count(T^t_{cpos}, i) = cValid(T^t_{cpos}). \tag{5.28}$$

The *equal* and the *count* function perform a grouping operation on the possible quantised time positions $t_i \in T^t_{cpos}$. For each group of equal possible quantised time positions *count* retrieves the number of resolution entries r_i, attached to the group. Using the *count* and *valid* function a distribution vector $Q^t(T^t_{cpos})$ can be defined:

$$Q^t(T^t_{cpos}) = \left(q_1, \ldots, q_{|R|} \mid q_i = \frac{count(T^t_{cpos}, i)}{cValid(T^t_{cpos})} \right) \tag{5.29}$$

Analogous to T^t_{cpos}, in the following q_i will denote the i-th element of Q^t. From Equation 5.28 follows that the sum of all elements in Q^t is equal to one:

$$\sum_{i=1}^{|R|} q_i = 1,\ q_i \in Q^t \tag{5.30}$$

It also follows:

$$\forall\ i \neq j :\ q_i > 0 \wedge q_j > 0 \Longleftrightarrow t_i \neq t_j \wedge valid(t_i) = 1 \wedge valid(t_j) = 1 \tag{5.31}$$

If R contains more than one entry, the vector $Q^t(T^t_{cpos})$ gives different weights (or salience) to the valid, closest grid positions created by $t_i = a_i \cdot r_i$. This can be compared to the different strength of metrical grid positions in scores as proposed by Lerdahl and Jackendoff [LJ83]. Different from their strict hierarchical approach – where certain poly-rhythmic structures cannot be expressed (see [LK94]) – the multi-grid approach represents a merge of (arbitrary) layers or strata of a certain beat duration (see also [Mor99, Yes76]). For the quantisation process an entry $q_i \in Q^t$ can be interpreted as *attraction* of the associated quantised time position $t_i = a_i \cdot r_i$ to the unquantised time position t.

For the final quantisation – which is a selection of one final closest grid positions out of T^t_{cpos} – now two measures are available: the attraction (or metrical strength) q_i and the absolute distance between t_i and the unquantised time position t, *i.e.*, $\delta(t, t_i)$ respectively $p_\delta(t, t_i)$. Table 5.2 shows a small example for a given resolution set R and an unquantised time position t. For $t = 13/64$ here the time position with the highest attraction for is $t_1 = 1/4$ but the absolute closest time position is $t_3 = 3/16$.

5.2.2 Selection of n Best Solutions

As shown in the previous subsection two measures are now available for choosing a final quantised onset time $t_i \in T^t_{cpos}$ for an unquantised time position t: the absolute distance $\delta(t, t_i)$ and the attraction q_i.

	i		
	1	2	3
r_i	1/4	1/8	1/16
t_i	1/4	1/4	3/16
$\delta(t, t_i)$	+3/64	+3/64	-1/64
$count(T^t_{cpos}, i)$	2	0	1
$Q^t(T^t_{cpos}, i)$	2/3	0	1/3

Table 5.2: Example for multi-grid quantisation of an unquantised time position $t = 13/64$ using a set of resolutions $R = \{1/4, 1/8, 1/16\}$ and search windows $lsWindow = -1/8$, and $rsWindow = +1/8$.

For balancing this two possible different time positions these two measures are now combined to a single quality (or probability) measure. We define the grid probability $p_g(t, q_i, t_i)$ as

$$p_g(t, q_i, t_i) = q_i \cdot p_\delta(t, t_i). \tag{5.32}$$

The function p_δ has been defined in Equation 5.21 as the normalised version of the distance function $\delta(t, t_i)$. Given the vector T^t_{cpos} and Q^t, a grid probability vector P^t_g can then be defined as

$$P^t_g(t, Q^t, T^t_{cpos}) = \left(p_1, \ldots, p_{|R|} \mid p_i = p_g(t, q_i, t_i), q_i \in Q^t, t_i \in T^t_{cpos} \right). \tag{5.33}$$

During the quantisation process the vectors P^t_g, Q^t, and T^t_{cpos} will be calculated for the unquantised onset time and unquantised duration of every note m of the performance. Then in a separate step the vector P^t_g and some additional rules (for avoiding incorrect overlappings, see Equation 5.35 and Equation 5.36) are used to select the final quantised time position out of T^t_{cpos}. For reasons of efficiency here only the two best possible grid positions (with highest p_i) are stored for each note m and passed to the selection module. In the following $o_{best}(m)$ and $o_{second}(m)$ should denote the best and second best closest possible score position in T^t_{cpos} for note m. Analogous the two duration alternatives are denoted as $d_{best}(m)$ and $d_{second}(m)$. As shown before they are calculated for each note by evaluating the grid probability $P^t_g(t, Q^t, T^t_{cpos})$. This measure gives respect to absolute distance between score and performance time position and the attraction of a score time position respectively a score duration. It does not evaluate any information about the durations or onset times that have been chosen for previous notes. In Section 5.2.4 an extension will be showed which allows the evaluation of this history information.

5.2.3 The Final Selection

After retrieving a first and second best solution for the onset time and quantised duration of all notes, the final selection will be made. Because the final selection for the onset time of note m_i depends on the final onset time selection of note m_{i-1} and – with minor influence – also on the final selection for the duration of note m_{i-1} the final selection is performed note by note instead of finalising first the onset times and then the duration of all notes. For finalising the quantised onset times of an ordered set of notes

$$M = \{ m_1, \ldots, m_{|M|} \mid onset_{perf}(m_i) < onset_{perf}(m_{i+1}) \}^4 \tag{5.34}$$

of a single voice the following constraints must be satisfied for the quantised onset time position $onset_{score}$:

1. $onset_{perf}(m_i) < onset_{perf}(m_{i+j}) \Longrightarrow onset_{score}(m_i) < onset_{score}(m_{i+j})$ (5.35)
2. $onset_{score}(m_i) := o_{best}(m_i) \vee onset_{score}(m_i) := o_{second}(m_i)$ (5.36)

Notes which did start at different performance time positions should be quantised to different score time positions preserving the order in time. If it is not possible to satisfy the onset time constraint (Equation 5.35) for two notes m_i, m_{i+1} with the previously selected best and second best onset time

[4] It is assumed that all notes with $onset_{perf}(m_i) = onset_{perf}(m_{i+1})$ have already been split to different voices or merged to chords.

alternatives, the notes must be merged to a chord and marked as a quantisation error or – in an interactive implementation – the user can be prompted to resolve the error. Similar to the duration-position errors (see Section 4.3.4) these *overlap errors* can be detected by the algorithm itself but they generally cannot be solved by the algorithm, because the reason for the error cannot be detected automatically. The errors, for example, might be caused by a wrong or missing entry in the set R or a wrong final selection for a previous note. Other quantisation errors caused by a too fine grid resolution (see Figure 5.2, right) cannot be detected directly.

In the following we describe the details of the final selection process.
For an ordered set of notes M (as defined above) the quantised durations must satisfy the following constraint:

$$onset_{score}(m_i) + duration_{score}(m_i) \leq \max\{o_{best}(m_{i+1}), o_{second}(m_{i+1})\}. \tag{5.37}$$

Because the final selection is performed note by note, the final onset time for m_{i+1} is still unknown when processing the final selection for onset time and duration of note m_i. If both duration alternatives for m_i are longer than the maximum IOI between $onset_{score}(m_i)$ and $\max\{o_{best}(m_{i+1}), o_{second}(m_{i+1})\}$ the duration of m_i must be cut down to the remaining IOI:

$$\begin{aligned} \min\{d_{best}(m_i), d_{second}(m_i)\} > \max\{o_{best}(m_{i+1}), o_{second}(m_{i+1})\} - onset_{score}(m_i) \\ \Longrightarrow duration_{score}(m_i) := \max\{o_{best}(m_{i+1}), o_{second}(m_{i+1})\} - onset_{score}(m_i) \end{aligned} \tag{5.38}$$

This means that the previously selected best duration alternatives for m_i will be discarded. It should be noted that $onset_{score}(m_i) < \max\{o_{best}(m_{i+1}), o_{second}(m_{i+1})\}$ is always true at this step, because of the onset time constraints in Equation 5.35 and Equation 5.36. If not, an error would have been already detected and resolved (*e.g.,* by a merge operation or user interaction) during the previously proceeded onset time finalisation step for note m_i.

In general the presented multi-grid quantisation offers a more adequate grid handling than strict hierarchical approaches (*e.g.,* [LHL82]) where it is not possible to use different grid resolutions on the same hierarchical level. To allow, for example, eighth triplets and eighth notes as grid positions in a single grid approach a small resolution of $1/24$ must be used. Using a multi-grid approach only half of the grid positions will have a small distance of $1/24$ (see Figure 5.5) which reduces the possibility of wrong quantisation drastically.

Improved Strategies for The Final Selection?

In priciple the previously described final selection task could also be modelled as a general optimisation task. The input would be an ordered set of notes and for each note a set of n best onset time and duration possibilities. The output would be the quantised onset time and duration for each note obtained by minimising a global error function F_{quant}. At first glance dynamic programming might be a candidate for solving this optimisation problem. But it seems not adequate (or even impossible) to transfer the context sensitive constraints for the onset times and durations (Equation 5.35–Equation 5.38) into a non-context sensitive cost function required for optimisation by dynamic programming. Another possible approach for the optimisation task would be a local search implementation combined with a random walk strategy, such as used for the voice separation module (see Chapter 2).

Tests showed that for the described selection strategy in most cases the quantisation errors were caused by not available context and history information of previously chosen durations and not by insufficient selection strategies. Instead of improving the quantisation quality only slightly by a more complex selection strategy we focus here on eliminating a general issue of the described basic multi-grid quantisation: the limited *view* to single notes and their onset time and duration data. To improve the quality of the multi-grid quantisation the distribution and frequency of selected durations (*i.e.,* integer multiples of a resolution r_i) in the local past can be used for the weighting of the possible quantised time positions of T^t_{cpos}. In the following sections this improvements to the multi-grid quantisation are described.

5.2.4 History-Based Multi-Grid Quantisation

As shown in [Smo94] the general issue of quantisation might be seen more as a categorisation issue than a simple round-off problem. Two notes at different positions in a performance with an equal observed absolute duration or IOI t might have different score durations d_1, d_2 (*e.g.,* $t = 200\text{ms}, d_1 = 1/8, d_2 = 1/12$). In general grid-based quantisation approaches which evaluate only the absolute duration of a performance note and ignoring any context information of a note might not be able to quantise equal length notes to different score durations. By evaluating the history of already processed notes (*e.g.,* a quarter note duration has been selected n times, an eighth note duration $n/2$ times) a more context sensitive behaviour can be created. Using this history information the intuitive adaptation to previously selected note durations – which might be typical for a performance or a section of it – of an human transcriber should be simulated by the algorithm. A basic assumption behind this approach is that an human transcriber gets more tolerant against deviations between performance and score timing if he has chosen some durations with a significantly higher frequency than others in the local past.

This assumption gets supported by latest research of Zanette ([Zan04]) who showed that the distribution of the different note durations in a single composition seem to follow Zipf's law and that listening to a part of a performance creates a (style) dependent expectation about the structure of the upcoming part of the performance.
In Figure 4.3.3 already an improved selection and weighting strategy (*i.e.,* the binclass approach) for evaluating the distance between an unquantised performance duration and a set of quantised score durations has been proposed. This strategy evaluates the frequency of selection for each duration class in the local past of a given time position. By using this binclass approach the distance $p_{bin}(dur, D)$ between an unquantised duration dur and the closest class of a set D of known duration (or IOI) classes also can be calculated here in the context of quantisation. Because the binclass approach can only be applied directly to a finite set of possible durations but not to arbitrary time positions, for the onset time quantisation the unquantised onset times must be expressed as unquantised score timing inter-onset intervals. Given a binclass list D, the overall attraction $p(t, q_i, t_i, D)$ between an unquantised duration (or IOI) $t = onset_{perf}(m_j)$ (or $t = IOI_{perf}(m_j)$) and a quantised duration (or IOI) $t_i \in T^t_{cpos}$ can then be calculated with[5]

$$p(t, q_i, t_i, D) := p_g(t, q_i, t_i) + p_{bin}(f(t_i), D) - p_g(t, q_i, t_i) \cdot p_{bin}(f(t_i), D), i \in 1, 2, \ldots, |R|^6 \tag{5.39}$$

where $q_i \in Q^t$ should be defined as shown in Equation 5.29, $p_{bin}(t, D) := 1 - d(t, g_{best}(t, D))$ (see Equation 4.49 for the definition of $g_{best}(t, D)$ and $d(t, g)$), and

$$f(t_i) := \begin{cases} t_i, & \text{for duration quantisation} \\ & \quad (t_i \text{ is already a duration}), \\ t_i - onset_{score}(m_{j-1}), & \text{for onset time quantisation.} \end{cases} \tag{5.40}$$

Because at this stage of the quantisation process there still exist two alternatives for the score onset time of the previous note, $onset_{score}(m_j)$, the IOI which maximizes $p_{bin}(f(t, D, i), D)$ is chosen as result of $f(t)$. Each resulting quantised IOI between note m_{j-1} and m_j is again stored in the binclass list D. If for a note m both alternatives satisfy the shown constraint the one with a smaller distance to a closest class in D is selected as final selection. During all steps the different binclass lists D need not necessarily be the same classes for duration and onset time quantisation. They also can be initialized with bias values for the different classes obtained from previously processed data. The local-statistical quantisation should be done in a two phase solution:

1. Store all possible solutions (*i.e.,* best and second best possibility) in two binclass lists *qOnset* and *qDuration*, and use these lists for the selection of the next possible solution.

2. When selecting the best solution store the finally selected duration and onset time in a *sOnset* and *sDuration* list and use these lists for the final selection of the values for the next note.

[5] During the duration quantisation, t will correspond to the duration of a note, during onset time quantisation, t will correspond to the observed IOI resulting from the onset time of successive notes.
[6] See Equation 5.33 for the definition of p_g.

It might always happen that an observed, inexact note duration matches perfectly the duration of a duration class which is different from the written duration of the score. This type of error cannot be detected by an algorithm with a single note view to the input data. Only if the algorithm would have information about relations (*e.g.,* grouping information) of successive notes this error could be improved. Therefore a pattern-based quantisation approach was developed which is shown in the following section.

5.3 Pattern-Based Quantisation

A quantisation approach using free definable rhythmical patterns to our knowledge has not been described in the literature before. There exist approaches using pattern information for quantisation (*e.g.,* Section 5.1.3, Section 5.1.5), but different from out approach, here the set of pattern is somehow hidden and cannot be defined or changed by the user. The type of used patterns also is rather small and restricted. Also the Hidden Markov Model (HMM) approach for combined tempo detection and quantisation as described by Takeda *et al.* in [TNS03] uses *learned* patterns for inferring score times and durations for performed notes. The authors use the assumption "...that the tempo is constant or changes slowly, the proportion of consecutive note lengths x is nearly independent from tempo τ." With an approach based on explicit patterns, preferable possible transitions, or typical standard transitions in sequences of note durations can be defined in a very intuitive way. So a pattern-based model can be seen as an implicit Hidden Markov Model.

Different from a neural network solution the here presented approach works with patterns that are stored explicitly (human readable and editable) in GUIDO format syntax. In a future implementation they could – similar to a neural network model – automatically be detected and learned by the algorithm itself using the self-similarity approach described in Section 3.3. In some sense this solution combines the advantages of both worlds: the self learning features of neural nets and the explicit stored and user editable data of a rule-based solution.

A first simple version of pattern-based quantisation has been already described and implemented in [Kil96]. In this implementation the patterns were only used for the final selection of quantised onset time and duration in the set of best alternatives, estimated with the multi-grid quantisation approach (see Section 5.2). In the context of this thesis we developed an improved version of the pattern-based quantisation approach which now can be applied directly to unquantised performance data, a set of (best) possible quantised time positions is not needed here. Also the patterns can now be specified in GUIDO syntax where each sequence of a segment is interpreted as a single quantisation pattern; the pitch information of the GUIDO file will be ignored for quantisation (see Section A.3).

The here described pattern quantisation is based on a bar length l indicated by an equivalent time signature (given by the user, available in the input data, or inferred automatically). Different from the pattern-based tempo detection (see Section 4.3), here the pattern must not overlap. This means that for a bar length l only such patterns $P \in \mathcal{P}$ will be evaluated where

$$duration(P) = n \cdot l \text{ or } l = n \cdot duration(P),\ n \in \mathbb{N}. \tag{5.41}$$

Here $\mathcal{P} = \{P_1, \ldots, P_{|\mathcal{P}|}\}$ denotes the set of defined pattern P_i, and $duration(P)$ denotes the duration of the pattern $P = \{p_1, \ldots, p_{|P|}\}$ given by the sum of the IOIs of all pattern notes $p \in P$; see Section 4.3 for a formal definition of pattern and related functions. For a matched pattern P the (shifted) onset time position of its first pattern note p_1 can then be shifted only to specific (quantised) score time positions:

$$onset_{score}(p_1) = \begin{cases} n \cdot l + onset_{score}(p_1), & \text{if } duration(P) = n \cdot l \\ n \cdot duration(P) + onset_{score}(p_1), & \text{if } l = n \cdot duration(P) \end{cases} \quad n \in \mathbb{N} \tag{5.42}$$

Only for (uncommon) pattern that start with a rest the term $onset_{score}(p_1)$ will be > 0. The score time positions of the successive pattern notes are defined as

$$onset_{score}(p_i) = onset_{score}(p_{i-1}) + IOI_{score}(p_{i-1}), 1 < i \leq |P|. \tag{5.43}$$

For a pattern $P = \{p_1, \ldots, p_{|P|} \mid onset_{score}(p_i) < onset_{score}(p_{i+1})\}$ the calculation of a distance measure between P and a sequence of unquantised notes $M' = \{m_i, \ldots m_{i+|P|-1}\} \subseteq M$ (as defined in Equation 4.2) can be done in four steps:

1. Shift the onset times of the pattern notes (prelimary) to the closest score time position that holds Equation 5.42.
2. Calculate a distance measure between the new (preliminary) onset time positions, $onset_{score}$, of the pattern notes and the unquantised onset time positions of the notes in M'.
3. Calculate a distance measure between the score durations of the pattern notes and the performance durations of the notes in M'.
4. Combine the onset time and duration distance measure to a single distance measure between P and M'.

For a given pattern $P = \{\ p_1, \ldots, p_{|P|} \mid onset(p_j) < onset(p_{j+1})\ \}$,[7] a set of performance notes

$$M' = \{\ m_i, \ldots, m_{i+|P|-1} \mid m_j < m_{j+1} \wedge voice(m_j) = voice(m_{j+1})\ \} \subseteq M, \tag{5.44}$$

and a current bar length l we define the closest possible pattern start position k:

$$k_{P,M',l} = \begin{cases} l \cdot (\arg\min_{k' \in \mathbb{N}_0} \{|onset_{score}(m_i) - l \cdot k'|\}), & \text{if } duration(P) \geq l \\ duration(P) \cdot (\arg\min_{k' \in \mathbb{N}_0} \{|onset_{score}(m_i) - duration(P) \cdot k'|\}), & \text{if } duration(P) < l \end{cases} \tag{5.45}$$

For P and M' we define the onset time distance d_a:

$$d_a(P, M') = \arccos \frac{< a \cdot a' >}{\|a\| \cdot \|a'\|}, \tag{5.46}$$

with

$$a = (\ a_1, a_2, \ldots, a_{|P|} \mid a_j = onset_{score}(m_{i+j-1})\ ) \tag{5.47}$$

$$a' = (\ a'_1, a'_2, \ldots, a'_{|P|} \mid a'_j = onset_{score}(p_j) + k_{P,M',l}\ ). \tag{5.48}$$

And the duration distance d_d:

$$d_d(P, M') = \arccos \frac{< d \cdot d' >}{\|d\| \cdot \|d'\|} \tag{5.49}$$

(The use of the arccos as distance measure is also proposed in [ZP03], see also Equation 4.42.)
With

$$d = (\ d_1, d_2, \ldots, d_{|P|} \mid d_j = duration_{score}(m_{i+j-1})\ )$$

$$d' = (\ d'_1, d'_2, \ldots, d'_{|P|} \mid d_j = duration_{score}(p_j)\ )$$

Independent from d_a, d_d a pattern gets rejected (not accepted as a match) if one of the following constraints is satisfied:

$$\exists\ j, 1 \leq j \leq |P| :\ |a_j - a'_j| > t_a \tag{5.50}$$

$$\exists\ j, 1 \leq j \leq |P| :\ \Big(\frac{duration_{score}(m_{i+j-1})}{IOI(m_{i+j-1})} > 0.9\Big) \wedge \Big(\frac{duration_{score}(m_{i+j-1})}{IOI(m_{i+j-1})} > \frac{duration_{score}(p_j)}{IOI(p_j)}\Big) \tag{5.51}$$

Equation 5.50 rejects a pattern if the distance between the score time of any pair of performance note and pattern note gets larger than a the threshold t_a. In the current implementation t_a is set to 1/12 (eighth triplet). The constraint in Equation 5.51 ensures that performance note which have been played

[7] We assume that a pattern can include notes as well as rests, where the rests are not explicitly encoded as pattern notes. Instead they are indicated by $duration(p_i) \neq IOI(p_i)$ or $onset_{score}(p_1) > 0$.

with no or only a small rest between them will not be shortened by applying a pattern that includes a rest.

To adapt the distance measures, d_a, d_d, to the overall accuracy (see Section 4.3.5) they can be weighted by

$$d'_a(P, M') = d_a(P, M') \cdot (2 - W_{Gauss}(onsetAccuracy, 0.5)) \quad (5.52)$$
$$d'_d(P, M') = d_d(P, M') \cdot (2 - W_{Gauss}(durationAccuracy, 0.5)), \quad (5.53)$$

where *onsetAccuracy* and *durationAccuracy* are calculated as shown in Section 4.3.5. In the current implementation then the onset time distance d_a and the durational distance d_d are combined to a single pattern distance d_p as

$$d_p(P, M') = 0.8 \cdot d'_a(P, M') + 0.2 \cdot d'_d(P, M'). \quad (5.54)$$

It should be noted that for quantisation a set $\mathcal{P}$ of given pattern is used in two different ways:

- For the quantisation of onset times the pattern induce probabilities for transitions inside sequences of notes; similar to Markov chains.
- For the quantisation of the note durations they are used as templates which can be applied to a series of notes with (approximately) quantised onset times. Without these patterns/templates it is not clearly decidable if for examples rests should be put between closely played successive notes or if their score durations should be prolonged to the size of the IOI of adjacent notes.

In the current implementation of our system – similar to the tempo detection – first all performance notes are processed by the pattern-based quantisation module and then for all regions where no matching pattern have been found, the previously described multi-grid quantisation including the binclass improvement (see Section 5.2.4) is used. In worst case (no matching pattern), or if no pattern database is available, this might be the complete performance. During this phase the score durations and IOI ratios obtained by the pattern matching phase are used to create an initial distribution for the used duration and IOI ratio binclass lists. It should be noted that the pattern matching processes for tempo detection and quantisation in general work quite similar, but differ in some essential details:

- The set of patterns used for tempo detection usually includes rather simple patterns. By filtering inverse durational accents from the clicktrack complex rhythmic regions can be avoided.
  Hence the pattern set used for quantisation includes a large number of rhythmically complex patterns which could hardly be solved with non-pattern-based techniques.
- For tempo detection the similarity measure between pattern and performance notes must give respect to possible tempo fluctuations.
  For the pattern-based quantisation distances between pattern note and performance note onset times are not treated as potential tempo fluctuations. Instead they decrease the similarity measure between pattern and performance.
- The durations of pattern notes are evaluated only indirectly by the resulting IOI during tempo detection.
  For the pattern-based duration quantisation the durations of pattern notes and rests between pattern notes are evaluated explicitly.
- Because the tempo detection module tries to use overlapping pattern they have no length restriction, each pattern can have a different length.
  Because pattern-based duration must respect a given bar length l only those pattern of a given set $\mathcal{P}$ will be used that fit the length constraint defined in Equation 5.41.

In our implementation the sets of pattern for tempo detection and quantisation are both specified as files in GUIDO syntax, where each sequence of the GUIDO segment represents a single pattern. It is possible to use the same set of pattern for quantisation and tempo detection, but as shown above usually two sets with different features (rhythmic complexity, length) should be used (see also Section A.3).

5.3.1 An Hybrid Approach

As described above in best case the pattern-based quantisation can be used to quantise in every voice, each measure or groups of successive measures with a single or a group of rhythm patterns. In the general case we must assume that there exist groups of notes for which no pattern can be found, because they have been played very inaccurately or because the pattern is not contained in the database. In worst case (*e.g.,* empty database, database contains no pattern for a specific time signature), there might be no matching pattern available for the complete piece. Our current implementation recognises these gaps (*i.e.,* notes for which no matching pattern can be found) between the start of a pattern match and the end of the previous match. The corresponding notes become then quantised by the history-based multi-grid approach described in Section 5.2. The information about note durations and IOI ratios obtained from the pattern-based quantisation also is used to update the local history information of the binclass lists used for this non-pattern-quantisation. If, for example, patterns have been used to quantise the notes $m_i, \ldots, m_{i+k}$ then the binclass lists will include the duration and IOI ratio information of these notes (bias for different classes) for the non-pattern-based quantisation of notes $m_{i+k+1}, \ldots, m_{i+k+j}$.

5.4 Results

The quantisation of Bach *Minuet in G*, mm. 1-16 resulted in an almost completely correct score. For the evaluation of the quantisation module we used the same file as for the evaluation of the tempo detection module (Section 4.4). A quantisation error in measure 11 was caused by an unfiltered ornamental note which needed to be represented as a melody note in the score (see Figure 5.6). Also the last two notes

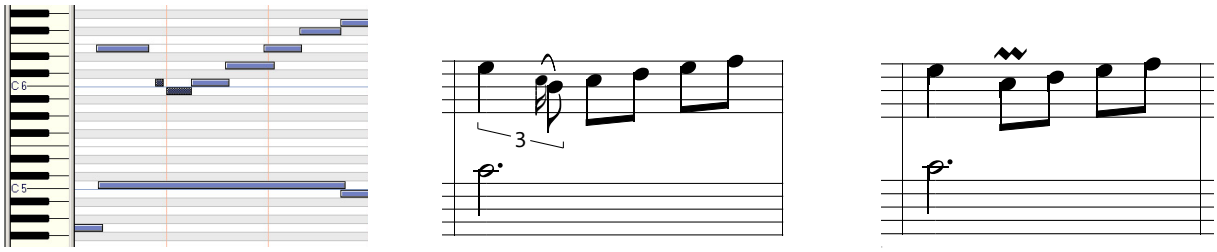


Figure 5.6: Measure 11 of the *Minuet in G* example, performance data (left), inferred score data (centre), and correct score data (right). The note with pitch b' could not be detected as an ornamental note, because it has nearly the same absolute IOI than the following eighth notes. Instead of a mordent (consisting of the first c" and the b') only the very short c" has been inferred as a grace note.

of the left hand voice were inferred incorrectly because of a strong final ritardando of the performance. Figure 5.7 shows the typical situation where the note durations of the performance (dark notes in piano roll, measure 2 and 4 in scores) are so different from the original score, that it seems not possible to infer the original note durations without an approach that allows to model preferences for rhythmical patterns. Because for the Minuet the merged clicknote track (used for tempo detection) included already all the note onset times (no clicks have been filtered) the pattern-based quantisation affected only the note durations. For the complete file consisting of 100 notes we observed a total number of three duration errors. The pattern database used for the quantisation of this file can be found in Figure A.5.

In Section 4.4 we already discussed the output of the hybrid tempo detection module for a performance of Beethoven's *Sonata in G* Op. 49, 2. As shown in Figure 5.8 also the quantisation module showed very good results. In measure two the first g' has been transcribed as a quaver instead a crotchet, because its performed duration was very close to a mechanical duration of a quaver. The errors in the upper voice in measure eight (semi-quaver c") and measure 12 (f♯' and g') have been caused by not detected ornaments. Similar to the example shown in Figure 5.6 the performed durations of these notes were to long to be detected as short ornamental notes. The dotted durations of the chords of the left hand voice in measure 10 and 14 also can hardly be avoided. Without knowing the original score there is no evidence why the transcription of these measure should be done in another way. The single events at this places – missing any local rhythmic context – also can not improved by the use of patterns. At the last quarter

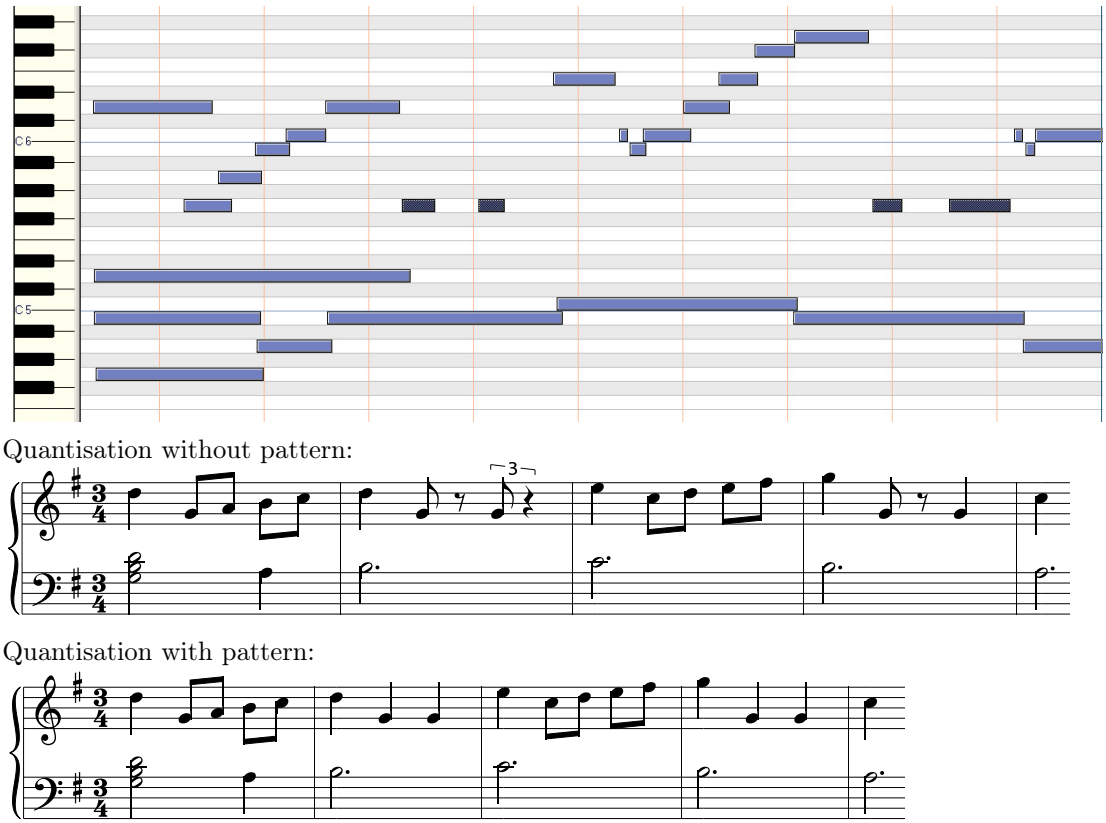


Figure 5.7: Bach *Minuet in G* piano roll representation of performance data (top), quantisation without the use of patterns (center), quantisation using a pattern database (bottom). For the c" in measure 3 and 5 ornaments have been inferred which are not shown in this scores. Please see Section 6.4 for a discussion of the inferred ornaments.

of measure 16 the rhythm switches from triplets to eighths as result of an error of the tempo detection module. As described in Section 4.4 the tempo detection switched back to the correct ternary rhythm after half of the next measure. The pattern database used for the quantisation of this file can be found in Figure A.4.

Figure 5.11 shows the transcription of the melody voice of the Jazz standard *Take Five* by Dave Brubeck. As input we used the melody voice of a performance MIDI file which had been recorded along a sequencer metronome click. The pattern consisting of a dotted half note followed by four sixteenth notes (measure 4, 5, 8, 9, 20, 21, 24, and 45) triggered pattern matches. Because the corresponding notes have been played more accurately, the typical pattern distance was decreased at this positions; other regions which had been played inaccurately then could not trigger pattern matches. Because the overall accuracy was rather low we added only eighths, quarter, and half notes to the onset time grid for quantisation. The sixteenth note groups could therefore only be realised by pattern matches. In the B part (measure 10-19) the transcription includes some measures which are different from the original score. By analysing the performance data (audio and piano roll) it turned out that already the performance was very different from the score (see Figure 5.9), so the transcription is actually correct.

Figure 5.12 shows the same MIDI data of *Take Five* imported by the Sibelius music notation software. Two allow the correct quantisation of the sixteenth notes groups, the smallest possible note duration (Sibelius open MIDI settings) has been set to the a duration of a sixteenth note. This caused that some of the onset times of the laid back played notes (*e.g.,* half notes in measures three and four, notes on last quarter of measures 3 and 7) have been quantised to wrong score positions. The options for importing MIDI files with Sibelius include some settings for handling tuplets. For triplets, and some other tuplet constructs (5-, 6-, 7-, 9-, and 10-tuplets) the user can select between four strategies: none, simple,

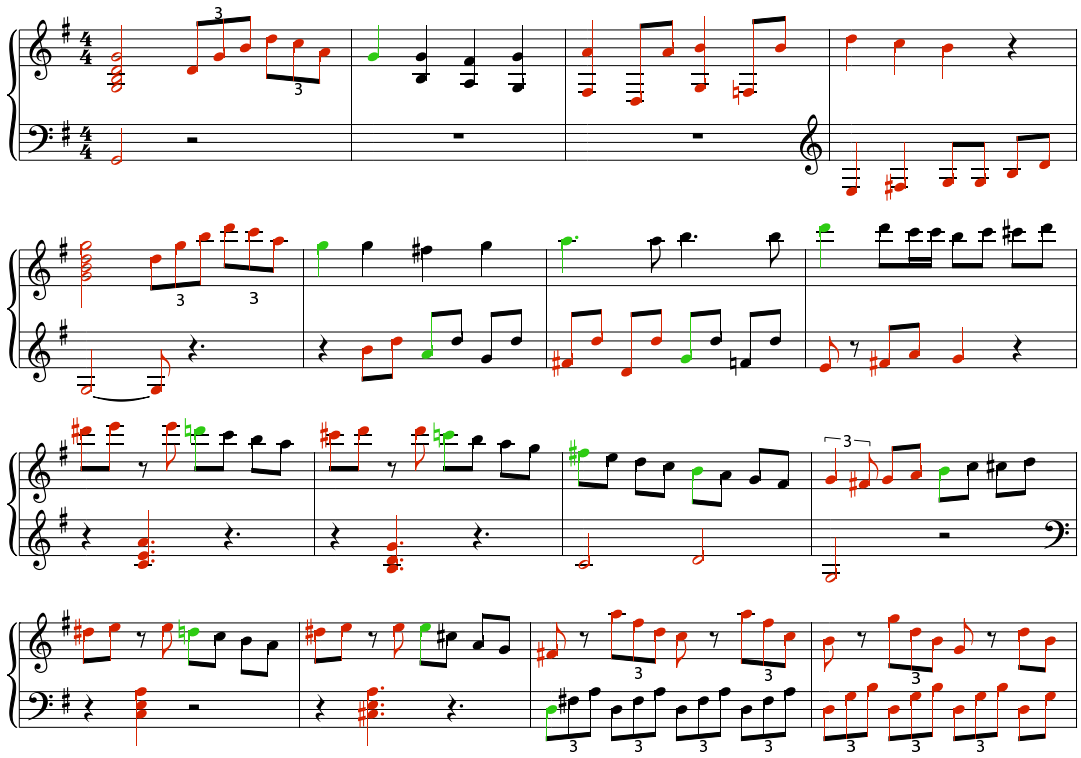


Figure 5.8: Beethoven *Sonata Nr. 20*, Op. 49, 2, mm. 1-16. The inferred ornaments have been omitted in the score for reasons of visibility. The clef changes have been added manually. Here green and black note heads indicate notes quantised by a pattern starting at the green note head position; read note heads indicate notes that could not be quantised by a pattern.

moderate, complex. Where the tuplets correspond induce a smallest possible note duration (*i.e.*, defining a grid) that can be chosen by the user. This means that if sixteenth notes and eight triplets should be allowed, also sixteenth triplets are allowed. Depending on the selected strategy the system prefers to use a binary rhythm notation or tuplet constructs for inaccurate played notes. For the score shown in Figure 5.12 we allowed simple triplet constructs, and set the strategy for all other tuplets to 'none'. An MIDI import with triplets set to 'none' resulted in notes merged incorrectly to a chord in measure 9 and 25 (see Figure 5.10). By using a moderate strategy for triplets it also was possible to create a score with a ternary transcription ([1/6 1/12]) for pairs of dotted eight and sixteenth notes in Figure 5.12.

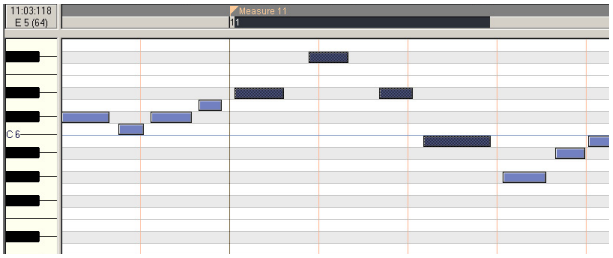


Figure 5.9: *Take Five* performance data: piano roll representation of measure 11.

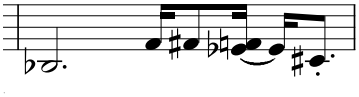


Figure 5.10: *Take Five* measure 9: result of quantisation with Sibelius, no triplets allowed.

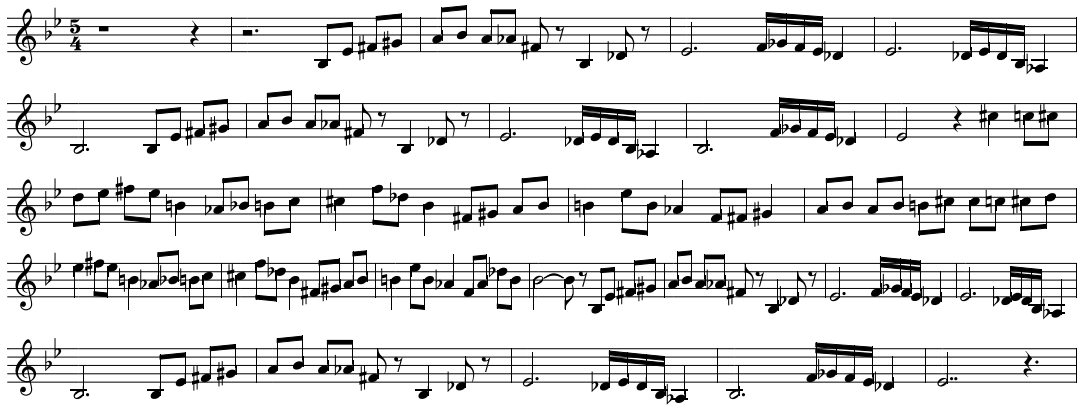


Figure 5.11: Brubeck *Take Five* melody voice, quantised with `midi2gmn`.

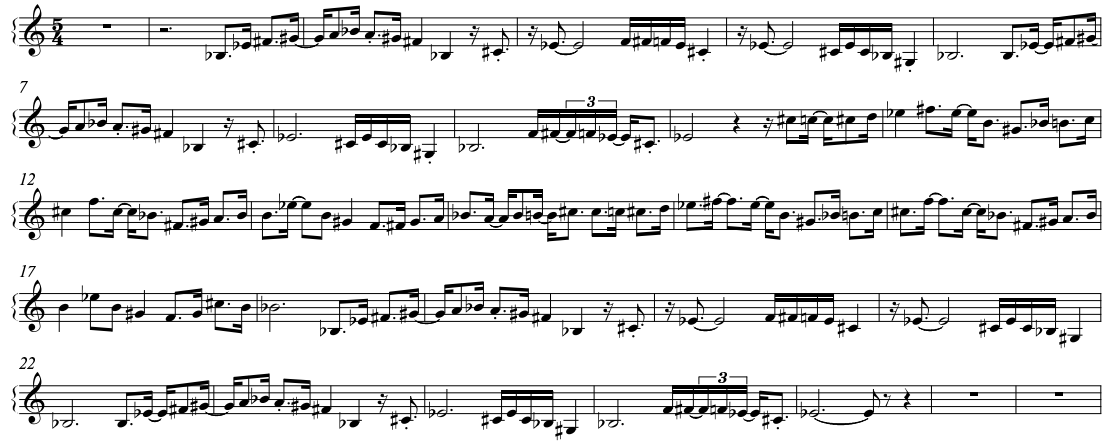


Figure 5.12: Brubeck *Take Five* quantised with Sibelius.

By modifying the list of allowed onset time positions R (adding the duration of an eighth triplet) we obtained an equivalent result also with our `midi2gmn` implementation. Different from our approach it was not possible to transcribe with Sibelius these groups as pairs of eighth notes, which is the standard way for swing melodies. Because the actual way to play rhythms with swing feeling is somewhere between a strict ternary rhythm ([1/6 1/12]) and a strict binary rhythm ([1/8. 1/16]) and because a rhythm pattern consisting of only eight notes might ([1/8 1/8]) be easier to read, scores for swing pieces commonly use this simple rhythm. It depends on the performer to add the correct amount of 'swing feeling' by adding a short delay to the onset times of all notes at 'even' score positions ($t = 2n/8$).

As described in Section 4.3.5 our approach tries to infer the type and accuracy of the given input data before starting the transcription. If the input data has been inferred as a mechanical performance, `midi2gmn` might prompt the user during the quantisation, if very small note values (*e.g.,* 1/64, 1/48) should be added to the duration grid for a most accurate transcription of the input data. For mechanical performances also the pattern-based quantisation will not be executed. In this case it is assumed that a mechanical performance includes both, correct onset times and correct durations which should not be changed by pattern matching. If a very low accuracy has been calculated, the quantisation module might ask the user if very small note durations (*e.g.,* 1/64, 1/48) should be removed from the quantisation grid, to reduce the chance of quantisation errors (see also Section 4.3.5).

5.5 Possible Extensions

During the development of the pattern-based quantisation we also tested an implementation were pattern that have been used very often in the past, were preferred to be used again. Herefore some utility functions which give a normalised measure about the frequency of the use of a pattern can be defined. For a pattern $P \in \mathcal{P} = \{P_1 \dots P_{|P|}\}$ and current input data M (a performance) we define:

$$cUsed_l(P) = \#\text{ of times } P \text{ has been used } \textbf{during} \text{ processing } M \tag{5.55}$$

$$cUsed_g(P) = \#\text{ of times } P \text{ has been used } \textbf{before} \text{ processing } M \tag{5.56}$$

$$cUsed_t(P) = cUsed_g(P) + cUsed_l(P) \tag{5.57}$$

A pattern is labeled as *used*, if it has been selected as the final match during quantisation. Using this functions we can define an *a priori* measure for a pattern $P_i \in \mathcal{P}$:

$$prior_l(P_i) = \frac{cUsed_l(P_i) + 1}{\sum_{j=1}^{n}(cUsed_l(P_j) + 1)} \tag{5.58}$$

$$prior_g(P_i) = \frac{cUsed_g(P_i) + 1}{\sum_{j=1}^{n}(cUsed_g(P_j) + 1)} \tag{5.59}$$

$$prior_t(P_i) = \frac{cUsed_t(P_i) + 1}{\sum_{j=1}^{n}(cUsed_t(P_j) + 1)} \tag{5.60}$$

The +1 terms are used to ensure that even a pattern that never has been used gets a chance to be used. It is easy to see that for $\forall\ P \in \mathcal{P}$: $prior_x(P) > 0$ and also $\sum_{j=1}^{|P|} prior_x(P_j) = 1$. Using the a priori functions the similarity measure between a series of performance notes M' and a pattern P could be biased to give respect that pattern matched very often in the local past have a higher chance to be used again:

$$d_{bias}(P, M') = \alpha \cdot prior_x(P) + (1 - \alpha) \cdot d_P(P, M'), 0 \leq \alpha \leq 1, \tag{5.61}$$

where d_p should be defined as shown in Equation 5.54. Our tests showed that the implementation of this feature does not increase the output quality. Also an own perception experiment (see Section A.14) showed that it even might not be the case that human listeners get tolerant against errors after listening to a repeated pattern. It might even be possible that listeners get intolerant against these types of errors. After listening to a number of quite similar repetitions of a single rhythmic pattern the listener might become even more sensitive to minor deviations and therefore perceive the erroneous pattern as a different one. We therefore did not include this feature any longer in the current implementation of our system.

For the current implementation of our system the patterns must be specified manually by the user in a GUIDO file. By analysing the finally quantised data it should be possible to identify significant patterns (multiple occurrences, high rhythmic complexity, uncommon durations) in the regions which could not be quantised with the given set of quantisation pattern $\mathcal{P}$ and add them automatically to the pattern database. Because the quantisation patterns must satisfy the bar length constraint (Equation 5.41) it should be a straightforward task to identify patterns with a pattern length equal to the current bar length as default or ask the user to specify the correct pattern length. An actual implementation of this strategy might include an interactive component, where the user can control which pattern should be added to the database or correct them before.

Another rather straightforward task would be the systematic evaluation of typical deviations (onset time, duration) between performance data and pattern data. For each pattern note a typical offset and its variance between pattern data and performance data could be stored in the pattern database and evaluated for the distance measure. We assume that there exist typical deviations for special rhythmic constructs (*e.g.,* the second note of a triplet group) but also typical deviations depending on the performer. We also assume that the typical deviation of a certain note of a rhythmic pattern is influenced by the melodic contour which is not evaluated by our model. If, for example, the third note of a group of four eighth notes represents a large upward interval, it can be expected that the typical deviation is different than for a small interval or an interval in the opposite direction. Because the evaluation of this typical deviations would require additional fundamental research we did not implement this feature in our

system. Nevertheless, by evaluating these additional information (deviation, melodic context) it should be possible to create a quantisation/tempo detection system which can be trained to specific performers, similar to OCR systems that can be trained to different types of hand writings. A system like this also might be used in a different way, where it somehow *teaches* or tells a student about his typical errors in the rhythmic timing of notes or patterns.

6 Secondary Score Elements

On the score level we can distinguish between at least two categories of graphical elements: basic elements indicating time and pitch information and secondary elements, such as for example, key- and time signature, intensity indications, articulation markings, or ornaments. Usually a musical score also includes additional information, such as lyrics, cue markings, or instrument specific markings (*e.g.,* sustain pedal (piano), mute indications (brass instruments), bow indications (string instruments)) which are not in focus of this thesis. Where a composition would still have a musical content if the secondary information would be removed, the musical information would be lost without the basic information for the events. For increasing the readability of the transcribed score and for improving the quality of the transcription itself (*e.g.,* a correct ornament detection will improve the output of the quantisation module), these information should be inferred automatically.
In the following sections we show possible strategies for inferring the time signature (Section 6.1), the key signature (Section 6.2), correct pitch spelling (Section 6.3), ornaments (Section 6.4) as well as some strategies for inferring a dynamic profile, slurring and staccato markings (Section 6.5, Section 6.6, and Section 6.7).

6.1 The Time Signature

Depending on personal view or musicological context the meaning of meter, rhythm, or accent is interpreted different. Where in some articles meter detection is used equivalent to inferring a time signature (*e.g.,* [MRRC82], [Row93]) in other works meter describes an interpreted rhythmical structure (*e.g.,* [Mor99]). A discussion of the general relations and interpretations of meter and rhythm can be found in [Mor99] and [LK94].

> "Meter is the periodic alternation of strong and weak accents. A metrical pattern usually contains nested hierarchical levels in which at least two levels of pulsation are perceived at once, and one level is an integer multiple of the other level." [LJ83] cited by [PK90]

Only with a given time signature the concept of bars (measures) can be introduced to musical scores. All measures (except the first and the last) have a length equal to the specified time signature, which might change during a composition. As stated by Palmer and Krumhansl ([PK90]), each beat position inside a measure defines an equivalence class (*e.g.,* the first beat of a bar, the second beat of a bear).

> "The most obvious role of meter is to allow a way of measuring time, so a performer or listener can reproduce or recognize the same set of temporal relations from one performance to another as well as within different sections of the same performance." [PK90]

Beside the basic bar length, the time signature in a musical score also specifies the different hierarchical beat levels and their relations. For example, a $\frac{4}{4}$time signature induces a whole note level (bar length), and its binary subdivisions (half, quarter, eighth, sixteenth). It should be noted that, depending on the style of music, at least one lower level might be divided ternary (*e.g.,* Swing Jazz: a quarter beat is divided into eighth triplets).
In the context of this thesis our main focus is on inferring a time signature that fits to the rhythmical structure of the observed performance data.[1] In the remainder of this section we first show some existing

[1] In the context of this thesis we use meter signature and time signature with equal meaning.

approaches for time signature detection and then we show some details about the adaptations we made to the approach of Brown for using this algorithm on quantised and unquantised, polyphonic symbolic input data. At the end of the section we show some results and discuss the limitations of the approach.

6.1.1 Existing Approaches

In literature exist some approaches focussing on the detection of meter signature. In the following we will give a short description of two hierarchy-based approaches and an autocorrelation approach.

Hierarchical Approaches

Chafe *et al.* describe in [MRRC82] (also discussed in [Row93] pp. 150) an approach for meter detection based on inferring an hierarchy of strong and weak beats in a performance. The approach works in several passes. During the first pass melodic and rhythmic accents (*e.g.,* local minima and maxima in pitch contour or duration) are identified as so-called *anchor notes*. Here all adjacent pairs of anchor notes with a similar distance are marked as *simple bridges*. In a second step the algorithm tries to propagate the distance (in time) between simple bridges $|a_{i-1}, a_i| \approx |a_i, a_{i+1}|$ to the left and right of this points so that the next anchor points a_{i+2}, a_{i-2} coincide with integer multiples of the simple bridges. If it is assumed that the performance includes a approximately stable tempo the distances between successive anchor points can be clustered as an hierarchy and expressed by (approximate) multiples of a smallest bridge length. Chafe *et al.* then assume that each meter signature has a typical hierarchy structure (*e.g.,* $\frac{4}{4}$ = 1:2:4) which can be used for inferring the meter signature from the inferred hierarchical structure.

A similar hierarchical structure for meter signatures has been proposed by Longuet-Higgins and Lee in [LHL84] (also discussed in [Row93], pp.151). They showed how listeners might perceive meter as a hierarchical strcutre of sub divisions. For example, a $\frac{4}{4}$ time signature can be represented as a binary tree where the beats of each level typically are subdivided into two beats of the next smaller level. A $\frac{6}{8}$ meter would be divided as [2 3], a $\frac{3}{4}$ meter as [3 2].

An Autocorrelation Approach

Brown proposed in [Bro93] an algorithm for the detection of meter signatures implemented as an autocorrelation model. The algorithm works with monophonic audio data as input, given as an ordered set of time slices $x[0], \ldots, x[N-1]$, each representing the average amplitude (audio) over the size of a single slice. For a set of potential bar lengths $L = \{l_1, \ldots, l_{|L|}\}$ then the most probable bar length is calculated using a short-time autocorrelation approach. Depending on the time units in which the elements of L are given the estimated best bar length will be in real-time units, or score time units. A bar length given in score time units can directly be used as a time signaure. For a bar length given in real-time units, the corresponding time signature must be calculated with Equation 1.1, where a local tempo s must be given.

Brown's approach is based on some assumptions about the distribution of note onset times to positions in measures/bars proposed by Palmer and Krumhansl ([PK90]). They showed by analysing the occurrences of notes at different metrical positions, that the most notes occur at the beginning of a measure – the downbeat position.

Using a given set of time slices $x[0], x[1], \ldots, x[N-1]$ with $d :=$ the duration of a time slice $x[i]$, Brown defines a weight function $A(m)$ for a bar length $l = d \cdot m$ as

$$A(m) = \sum_{n=0}^{N-1} x[n] \cdot x[n+m], \tag{6.1}$$

where $x[n] > 0$ if the slice $x[n]$ include any onset time, and the parameter N specifies the integration time of the autocorrelation.
Brown showed that results obtained with a short integration time (*e.g., N* = approximately 3s) are not significantly worse than using longer integration times (*e.g.,* half or complete duration of the performance).

For live performance data – most evaluation of Brown was done with quantised data – the output quality decreases with longer integration times, which can be explained by tempo fluctuations of the performer resulting in changes of the absolute bar length.

The bar length with a maximum value $A[m]$ can then be assumed as *original* bar length of the input data:

$$\text{bar length} = d \cdot \arg\max_{m \in \mathbb{N}} A[m], \tag{6.2}$$

where d might be given in real-time units (*e.g.,* seconds) or score time units.

Because only the bar length and not the time signature is inferred directly, ambiguous time signatures cannot be differenced with this approach. For example, it cannot be decided if a performance have been written with a $\frac{2}{2}$ or $\frac{4}{4}$ time signature, Also human listeners might not be able to decide between ambiguous time signatures without additional information, such as style, harmonic changes, or other musicology context. If applied to real-time data it is easy to see that Brown's approach can be used for all data with a nearly, or at least local constant, (performance) bar length. If the absolute bar length changes because of tempo fluctuations – the time signature stays constant – the autocorrelation model would fail.

Brown also proposes a narrowed autocorrelation function based on terms, such as $f(t) \cdot f(t + 2r), f(t) \cdot f(t + 3r)$, *e*tc. which should improve the results with audio data input. For symbolic input data we could not obtain significant better output than with the standard autocorrelation function as defined in Equation 6.1.

6.1.2 Autocorrelation on Symbolic Data

For our implementation we adapted Brown's approach – which was originally designed for streamed audio data – for the usage with symbolic input data. Our adaptation is similar to the work of Meudic ([Meu02]) who adapted Brown's autocorrelation approach for detecting and extracting metric groupings (*i.e.,* repeated groupings of equal length) from quantised MIDI data. Instead of splitting a continuous wave signal (where no explicit information about note on- and offsets is available) into time slices, we can use sequences of notes as input on the symbolic level. Because on this level for each note information about pitch, intensity, duration, and for chords also the number of notes is available, these information can be used to calculate a note weight which can be evaluated during autocorrelation. Meudic proposes to use a combination of five weights: absolute intensity, interval, difference between IOI and duration, duration, and number of notes (for chords).

Given an ordered set of quantised notes $M = \{m_1, m_2, \ldots, m_{|M|}\}$ with $onset_{score}(m_i) < onset_{score}(m_{i+1})$ and $voice(m_i) = voice(m_{i+1})$, and a bar length l given in score time units, we define a weight function $A(l)$ as

$$A(l) = \sum_{i=1}^{N} w(m_i, l), N \leq |M|, m_i \in M \tag{6.3}$$

where for quantised data

$$w(m_i, l) = \begin{cases} w'(m_i) \cdot w'(m_{i+j}), & \text{if } \exists\ m_{i+j} : onset_{score}(m_{i+j}) - onset_{score}(m_i) = l, \\ 0, & \text{otherwise.} \end{cases} \tag{6.4}$$

The weight function $w'(m)$ can be an arbitrary weight/significance function depending on the duration, IOI, IOI ratio, intensity, and number of chord notes of note $m \in M$.
Analogous to Equation 6.2 now the bar length l with a maximum value for $A(l)$ should be inferred as correct bar length:

$$\text{bar length} = \arg\max_{l \in L} \{A(l)\}. \tag{6.5}$$

Different from [Meu02] we allow a certain amount of inaccuracy for unquantised, symbolic input data and therefore refine the definition of w:

$$w(m_i, l) = \begin{cases} w'(m_i) \cdot w'(m_{i+j}), & \text{if } \exists\ m_{i+j} : l - \epsilon < onset_{score}(m_{i+j}) - onset_{score}(m_i) < l + \epsilon \\ 0, & \textit{otherwise.} \end{cases} \tag{6.6}$$

The size of ϵ depends on the expected amount of inaccuracies in the data. The window size d of the time slices $x[i]$ used by Brown has the same effect than parameter ϵ.

For applying the autocorrelation method now also to polyphonic data consisting of multiple voices there exist three different possibilities:

1. Merging all notes to a generic voice/track (similar as shown in Section 4.3.1) and applying the autocorrelation to this generic voice.
2. Applying the autocorrelation to each voice separately and selecting then a best bar length out of the set of retrieved bar lengths for each voice.
3. Applying the autocorrelation to each voice in parallel, which can be done very easy if the note data of all voices is stored in a single linked list. This is similar to method Item 1, but no merging operation is required.

In the following we assume that the input data consists of polyphonic, unquantised performance data, separated into several voices.
Given an ordered set of notes

$$M = \{m_1, \ldots, m_{|M|} \mid onset_{score}(m_i) \leq onset_{score}(m_{i+1}) \\ \wedge\, onset_{score}(m_i) = onset_{score}(m_{i+1}) \Rightarrow voice(m_i) \neq voice(m_{i+1}), \quad (6.7)$$

and a score time position t, a weight function w_{poly} for polyphonic data can be defined as[2]

$$w_{poly}(m_i, l) = \begin{cases} w'(m_i) \cdot w'_{sel}(m_i, l, \epsilon), & \text{if } \exists\ m_j : l - \epsilon < onset_{score}(m_j) - onset_{score}(m_i) < l + \epsilon, \\ 0, & otherwise. \end{cases} \quad (6.8)$$

Here the weight function w'_{sel} should be defined as

$$w'_{sel}(m_i, t, \epsilon) \;=\; \max_{k \in \mathbb{N}} \{w'(m_{i+k})\},\ \text{with } t - \epsilon \;<\; onset_{score}(m_{i+k}) - onset_{score}(m_i) \;<\; t + \epsilon. \quad (6.9)$$

In addition to the performance data, the autocorrelation approach needs a finite set L of potential bar lengths as input. If the performance data includes score level timing information the bar lengths can be given in score time units equal to their time signature.
For the elements of the bar length list again two approaches are possible:

1. L is a list of all integer multiples of a duration d in a specified range:

$$L = \{\ l \mid l = i \cdot d, i \in \mathbb{N} \wedge a < i < b\ \}, \quad (6.10)$$

   where a, b might be arbitrary constants. For example, $L = \{\ \frac{3}{16}, \frac{4}{16}, \ldots, \frac{32}{16}\ \}$.

2. L includes a list of the most common time signatures, which might also be biased by the frequency of their occurrence in standard music literature. For example, $L = \{\ \frac{4}{4}, \frac{3}{4}, \frac{2}{4}, \frac{4}{8}, \frac{6}{8}, \frac{3}{8}, \frac{5}{4}, \frac{7}{8}\ \}$.

If no valid solution can be found the system should ask the user. If the data has no rhythm, no maximum for a special time signature can be derived (*e.g.,* $A(\frac{4}{4}) = A(\frac{3}{4}) = A(\frac{5}{4})$). In this case also the user should be prompted for a valid time signature, or a default time signature (*e.g.,* $\frac{4}{4}$) should be used.

The derived time signature might be normalised ($\frac{6}{8} \leftrightarrow \frac{3}{4}$) by using heuristic preference rules (*e.g.,* denominator of time signature depends on denominator of most common used note duration). We assume that the correct, normalised time signature also depends on harmonic progressions or style dependent features which cannot be evaluated automatically without musicological background information.

[2]Here a note $m \in M$ can be equivalent to a single note or a chord.

6.1.3 Inferring Time Signature Changes

As shown by Brown a short integration time of approximately three seconds is enough for inferring a bar length and a time signature of a performance. Therefore it should be possible to start the autocorrelation at different positions of the performance data and infer a time signature change if different bar lengths are inferred for different start positions of the algorithm. For this approach several strategies need to be decided in advance:

1. Which and how many start positions should be used for the autocorrelation? Too many starts would increase the run-time, too less would result in skipped time signature changes.
2. When should a signature change should be triggered? If a bar length l' has only a slightly better weight $A(l')$ than an already established bar length l, it should not necessarily be selected as a new time signature.
3. If a bar length l has been inferred for a start position m_i and a different bar length l' for a start position m_{i+k} it needs to be calculated at which position $j, i < j \leq i + k$ the time signature change should be placed in the score.

A possible approach for the detection of time signature changes might be implemented according the following outline:

1. Search in the input data for potential positions $s_1, \ldots, s_n$ for a time signature change, for example, break points or rests in all voices, or a change of a most common note duration in one or several voices. To detect these positions the floating average approach described in Section 3.4.1 could be used.
2. Proceed autocorrelation for all inferred break points.
3. Infer a time signature change at position s_{i+1}, if the weight/rank of a bar length l inferred at position s_i is significantly lower than at position s_{i+1}.
4. If a time signature change was inferred for a position s_{i+1}, search for a best start position for the new time signature in all notes $s_i < onset(m_j), \ldots, onset(m_{j+k}) \leq s_{i+1}$. In worst case the autocorrelation would need to be restarted for each note in that range.

Because the time signature change detection was not in the main focus of this thesis it has not been implemented yet.

6.1.4 The Anacrusis (Upbeat)

After tempo detection and inferring the time signature, it is still unknown if the first performance note starts at score position zero (the score would start with a complete first measure) or if the score starts with an incomplete first measure – named anacrusis or upbeat. A wrong placement (time position) of the first note would shift the onset times of all successive notes to wrong score positions, which could result in unreadable scores (see Figure 6.1) and scores which do not reflect the structure of music as it is perceived by a listener.

Assuming that the correct time signature for a sequence of notes is given and assuming that – because the *one* of each bar has the highest metrical strength ([LJ83, Mor99]) – also a major number of notes that occur at the beginning of a bar (in the original score) will have a high rhythmic and/or melodic strength (accent), we try to shift the complete series of (quantised) notes in a way that the number of accented notes at beat one positions gets maximised. To achieve this maximum at least two strategies can be used:

1. Shifting of the score times of all onset times (their distances stay fixed) to a global optimum, by using a cost function $f(a, t)$ that evaluates the relation between a melodic/rhythmic strength a of a note and a time position t within a bar. In a finite set of possible time positions for the onset

time of the first note of a performance (or a series of notes), we can select that time position which maximises the sum of $f(a,t)$ for all notes of the performance. Because of the limited hierarchical structure of possible beat positions within a bar (*e.g.,* $\frac{4}{4}$: $32 \cdot 1/32$) the set of possible time positions for the first note is finite and also rather small. This approach would have difficulties to find a correct solution, of the first note of the performance is a syncopated note, *i.e.,* its correct score position is not on the beat.

2. Searching directly for a note at the beginning of the performance which should be placed at the first beat of a bar. For a given, fixed time signature (bar length) l and a note m_i we can calculate a single autocorrelation $A(l,i)$:

$$A(l,i) = \sum_{j=1}^{N} w(m_i, onset_{score}(m_i) + j \cdot l),\ N \leq |M|, \tag{6.11}$$

where $w(m,l)$ should be defined as shown in Equation 6.4. Now we search in the first k notes of the performance for the index b of that note which maximises the autocorrelation function $A(l,i)$:

$$b = \arg \max_{i \in \{1,2,\ldots,k\}} \{A(l,i)\},\ k < |M| - N, \tag{6.12}$$

where N denotes the integration time used in Equation 6.11. Because of the high value of the autocorrelation function $A(l,b)$ for this note m_b, it can be estimated that it is correct to shift it to the one of a new bar, all other notes can be shifted by the resulting offset. To stabilise a perceived tempo and a rhythmic structure the beginning of a piece must avoid too much syncopation,therefore we can assume that there exist at least some notes in the first k notes, located originally at the first beat of a bar.

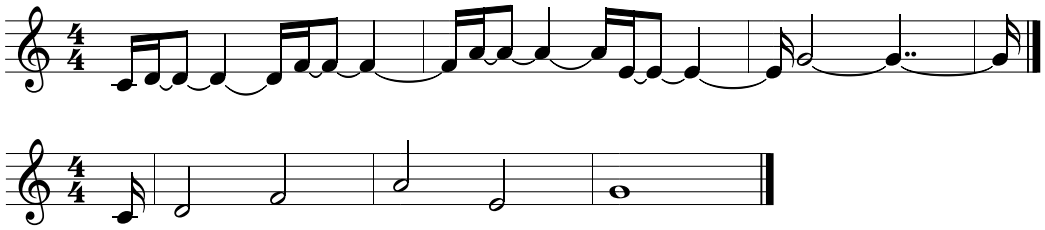


Figure 6.1: Shifting of score times through wrong (top) and correct (bottom) anacrusis.

In the current implementation, the meter detection including the upbeat shifting is performed directly before starting the quantisation module. The unquantised onset time positions must be approximately close to their correct beat positions (inside a measure) to ensure that the quantisation patterns are applied to the intended data.

6.1.5 Results

Because meter detection is not be the main focus of this thesis we performed only a briefly evaluation of our meter detection approach. We run the current implementation with several files of different styles as input and evaluated the transcribed files for correct meter signature. The results are shown in Table 6.1.

Because of the overall nearly complete sixteenth note grid in the Bach examples we assume that here the meter signature perceived by humans is created rather by harmonical progressions than on the rhythmical structures itself. Therefore we do not expect that meter detection approaches which focus only on rhythmical structure will usually have here an increased error rate. Nevertheless, the current implementation inferred the 'unusual' $\frac{9}{8}$ for Inventio 10 correctly. Similar to Brown we did not count an error for cases where the algorithm inferred integer multiples or divisions of the correct time signature (*e.g.,* $\frac{3}{4}$ instead of $\frac{12}{8}$ or vice versa).

collections	# files	# sign. errors
Bach *Inventio 1–15*	15	4
Bach *Sinfonia 1–15*	15	2
Bach *Well-Tempered Clavier Book II* *Prelude and Fugue 1–12*	24	0

single files	score	inferred
B. Bartók *Mikrokosmos 101*	$\frac{2}{4}$	$\frac{4}{4}$
B. Bartók *Mikrokosmos 144*	$\frac{4}{4}$	$\frac{2}{4}$
D. Brubeck *Take Five*	$\frac{5}{4}$	$\frac{5}{4}$
Debussy *Suite Bergamasque:*		
Prelude	$\frac{4}{4}$	$\frac{4}{4}$
Minuet	$\frac{3}{4}$	$\frac{6}{8}$
Clair de Lune	$\frac{9}{8}$	$\frac{9}{8}$
Passepied	$\frac{4}{4}$	$\frac{4}{4}$
Mozart *Clarinet Quintet* KV 581:		
set 1	$\frac{4}{4}$	$\frac{4}{4}$
set 2	$\frac{3}{4}$	$\frac{4}{4}$
set 3	$\frac{3}{4}$	$\frac{6}{8}$
set 4	$\frac{2}{2}$	$\frac{2}{4}$
Beethoven *Sonata Nr. 20, Op. 49, 2*	$\frac{4}{4}$	$\frac{4}{4}$
Bach *Minuet in G*	$\frac{3}{4}$	$\frac{3}{4}$
Dvořak *Humoresque*	$\frac{4}{4}$	$\frac{4}{4}$
Chopin *Op. 6, Mazurka 1*	$\frac{3}{4}$	$\frac{4}{4}$
Chopin *Op. 67, Mazurka 2*	$\frac{3}{4}$	$\frac{4}{4}$
sum	70	9

Table 6.1: Evaluation of the `midi2gmn` meter detection module.

6.2 The Key Signature

Beside accidentals (flats and sharps) for single notes, musical scores also include a key signature information, which indicates the default accidentals for a complete piece or larger parts of a piece. In Western tonal music there exist seven key signatures using sharp symbols , seven key signatures using flat symbols, and a natural key signature. Each of these 15 key signatures can denote a major or minor key of the circle of fifths.

A key signature might be interpreted different depending on the view to the score: with a harmonic view the key signature should be equal to the tonal centre of the current harmonic progression; with a melodic view the key signature should minimise the number of required additional single note accidentals. We assume that for tonal (classical) music both cases usually result in the same key signature, with an harmonic view to key signature there might also exist a unique correct key signature, at least for individual segments of a piece. For contemporary music which sometimes avoid any traditional harmonical progressions or any notion of traditional harmony we would assume, that there might exist no unique correct key signature. But the melodic view for minimising the number of additional accidentals could also be applied to music of this style.

If we assume that (for tonal music) the harmonical context – or the typical pitch classes used in the melody – may change during a piece (*i.e.,* modulation) there arises the question: when should these changes explicitly be indicated by a change of the key signature? Composers and arrangers usually do not indicate modulations by key signature changes, if these modulations are rather short and return to the original key after a few measures. Key signature changes are usually indicated only at positions where

a new part or set of the piece starts (*e.g.,* Trio, Verse, Chorus) or changes of other features also take place (*e.g.,* tempo, meter). Most known models for key detection (including our approach) are focussing only on the detection of a single key signature and omit the issue of signature changes. If the input data could be segmented in advance (based on other features, such as detected repetitions), then these key detection models could be applied the each segment separately.

6.2.1 Existing Approaches

In literature different approaches and models for inferring a key signature for symbolic performance data have been described. In the following we give a short overview about these models and approaches.

A Connectionist Approach

Scarborough, Miller and Jones presented in [SMJ89] a connectionist model for tonal analysis of Western tonal music. Their approach consists of a key finding and a harmonical analysis part. The key detection module is pitch-class-based, it evaluates the distribution of observed pitch classes in a piece. The basic assumption is that for each key signature the pitch classes of its corresponding scale can by characterised by a generic weight. For example, the prime, third and fifth of the scale have a high weight because they form the tonic root chord. For inferring a key signature their model uses a three layer inter connected net. Layer one consists of twelve pitch class nodes where each node represents an octave independent pitch class. Layer two represents chord nodes where each chord node is connected by weighted connections to the pitch pitch class nodes corresponding to its three chord notes. Finally layer three represents key signature nodes where each node is connected (weighted) to the chord notes if its root, subdominant and dominant chord.
The pitch class nodes become activated by the notes of the performance which are evaluated in ascending time order. The activation caused by a single note depends on the duration of that note: longer notes result in an higher activation level; the level of activation decays automatically in time. The activation level of a chord node depends on the activation level of the three connected pitch class nodes and the strength/weight of the corresponding connection. Analogous the activation level of a key signature node depends on the activation level of the three connected chord nodes and the strength/weight of the corresponding connection.

Because the notes respectively their pitch class is evaluated only by their onset time and their duration this model needs no information about the voice or the chord to which a note belongs, and can therefore be applied to monophonic data as well as to arbitrary polyphonic data.
Instead estimating the connection strength between nodes of the different layers by machine learning or training of the net, the authors use intuitively, manually adjusted weights. Because the mapping between the occurrence of pitch classes and key signature is given via the chord layer it is not clear if this approach can also be used for inferring a accidental minimising key signature for atonal music. Unfortunately the authors give in [SMJ89] no detailed evaluation of their results.

Cypher

Similar to the approach of Scarborough *et al.* also the key induction module of Rowe's Cypher system [Row93] (see also [Row01]) applies weights to possible key signatures that depend on the pitch class of observed chords. The weights depend on the musicological relation between an observed chord and 24 possible major and minor key signatures. Different from [SMJ89] the weights might also be negative if an observed chord would not fit to a certain key signature. [Row01]: "A C major chord for example, will reinforce theories for which the chord is functionally important (C major, F major, *e*tc.) and penalize theories to which the chord is alien (C minor, B major, *e*tc.)."

Melisma Analyser

David Temperley proposed in [Tem01] an implementation of a revised version of a key finding algorithm by Schmuckler and Krumhansl. Similar to the approaches described above also this algorithm is based

on the assumption that there exist for each key typical distributions in the number occurrence of the twelve pitch classes. For inferring a key signature for a sequence of notes s the correlation between the measured distribution of used pitch classes in s and the known typical distribution of pitch classes for each possible key signature is calculated.

For inferring key changes the piece must be segmented (*e.g.,* in single measures or phrases), then for each segment a weighted list of possible keys can be calculated. By using two rules ([Tem01], p. 188):

- **KPR 1**: For each segment, prefer a key which is compatible with the pitches in the segment, [...]
- **KPR 2**: Prefer to minimise the number of key changes from one segment to the next.

and a cost functions for key signature transitions between successive segments. A global, optimised set of key signatures for the complete set of segments is retrieved by using a dynamic programming approach.

Because the model is based on the evaluation of pitch classes, (ignoring pitch relations between successive notes within a voice) it does not require a voice separation. By estimating that there exists different typical pitch class distributions for minor and major keys it can also detect the key and its gender (minor/major).

In Section 6.2.3 the output of Melisma's key finding module is compared to the results obtained with our approach.

The Spiral Array

A complete approach for detecting the key signature and also key changes (key boundaries) within performances is proposed by Chew in [Che02]. By using the here described *Spiral Array* approach, the pitch classes become organised in along a three dimensional spiral in the order of their occurrence in the circle of fifths. The distance between two successive pitch classes is equivalent to a quarter turn of the spiral. Two vertical aligned neighbours – after four quarter turns of the spiral – have a pitch distance of a major third.
Similar to the model of Scarborough *et al.* major and minor chords (including three notes) are represented by weighted combinations of their chord notes. Depending on the weights a chord becomes represented by a point on the triangle plane created by the spatial points (on the spiral) of its three chord notes. Also similar to the approach of Scarborough *et al.* a key is represented by a spatial point in the triangle plane created by the spatial representation of its tonic, subdominant, and dominant chord.

For inferring a key for a given series of notes s, first all spatial positions of all pitch classes in s will be calculated. Then the *centre of effect* c centre of effect of these pitch class positions, *i.e.,* representing a weighted average over the calculated spatial points, can be calculated. The weight of each pitch class position to c depends on the duration (not frequency) of its usage in s. As key signature that key signature in the set of all possible key signatures – created by all possible chords in s – is chosen, which is closest to the centre of effects c. The distance between c and the spatial point representing the key can be interpreted as the likelihood of the key for the series s.

Chew also proposes a Boundary Search Algorithm (BSA) which can detect key boundaries (*i.e.,* key signature changes). Similar to the Melisma approach, this approach does not decide if a detected boundary (*e.g.,* caused by a modulation) must or should be indicated by an explicit key signature change in a corresponding score. BSA searches for a set of optimal boundaries for which the global sum of distance between key and centre of effects becomes minimised. By limiting the total number of boundaries to a constant m and adding further – style dependent – constraints, such as "Adjacent key areas should be distinct from each other, ..." and "If the passage to be analysed is a complete piece, the first and last key areas should be constrained to be the same, ..." the search space of $O(n^m)$ (where n denotes the number of notes in s and $m \ll n$) can be reduced.

Similar to the approach of Scarborough *et al.* this approach by Chew evaluates only pitch classes, ignoring the order of their occurrence. Different from Scarborough's approach the weight of a single pitch class depends on the sum of its duration and not on the number of occurrence. Chew presents two small examples which show good results for the BSA algorithm (compared) to manually inferred key boundaries. For the two examples the number of boundaries was limited initially to the correct number of two boundaries. Chew proposes that it should be possible to use BSA also in a real-time system.

Where the approaches described above evaluate the pitch classes of all notes of a voice or a complete performance, Chafe *et al.* proposed in [MRRC82] a different model that evaluates only the pitch classes of significant rhythmic or melodic accents (see Section 6.1.1). For pitch-class-based algorithms this seems to be an interesting concept which should be evaluated in combination with the different approaches for key detection shown above. For approaches that evaluate the distribution of observed pitch transitions (as shown in the following subsection), instead the distribution of absolute pitch classes, a filtering of significant notes might destroy the distribution of pitch class transitions between successive notes and should therefore be avoided.

6.2.2 Transition-Based Key Detection

In the context of this thesis we developed a more simple approach for key detection which can work without any harmonic analysis (approaches for harmonic analysis of musical data are described, for example, in [Win68]). Our approach infers a key signature for a sequence of notes M by evaluating the transitions (intervals) between successive note $m_i, m_{i+1} \in M$. The basic assumption here is that because each key signature can be characterised by the position of the two semitone steps inside their diatonic scale (see Table 6.3), it should be possible to infer a key signature by analysing the number and position of observed semitone steps between successive notes of a performance. If, for example, for a performance M the semitone transitions f$^\sharp\leftrightarrow$ g and c$^\sharp\leftrightarrow$ d, have the highest frequency of occurrence in M, then D major (or B minor) can be estimated as the correct key signature. Because here the focus is only on the correct *spelling*[3] of the key signature major keys can be used equivalent to their related minor key signature.

Our *simple* key signature detection algorithm works on quantised or quantised data which has already been separated into voices and/or grouped to chords (see Chapter 2). It includes two separate steps:

1. Count in each voice for all pairs of successive notes m_j, m_{j+1} the frequency of occurrence for each of the twelve possible semitone transitions $t_i = pc_i \leftrightarrow pc_{((i+1) \bmod 12)}, i = 0, 1, \ldots, 11$ (*i.e.,* c$\leftrightarrow$c$^\sharp$, c$^\sharp\leftrightarrow$d, ..., b$\leftrightarrow$c) (see Table 6.2):

$$\forall\ j \in 1, 2, \ldots, |M| - 1 : |pitch(m_j) - pitch(m_{j+1})| = 1 \Longrightarrow \text{ increase counter } o_i, \tag{6.13}$$

   with $i = \min\{pitch(m_j), pitch(m_{j+1})\} \bmod 12$. In situations where m_j or m_{j+1} represent a chord consisting of several chord notes we evaluate all possible transitions from each chord note of m_j and every chord note in m_{j+1}. So the approach can also work with polyphonic music given as a set of successive chords not split into single voices.

2. Find a transition pair t_i, t_j with $j = (i+7) \bmod 12$ and a maximum sum for the number of occurrence of o_i and o_j. Because each key signature is characterised by the position of the semitone steps in its diatonic key scale (major key: semitone step between note $3 \rightarrow 4$ and $7 \rightarrow 8$; the distance between note 3 and note 7 of a diatonic major scale is always 7 semitones) each transition pair o_i, o_j with $j = (i + 7) \bmod 12$ correspondents to a specific key signature or its enharmonic equivalent (see Table 6.3).

As shown in Section 6.2.3 even this simple approach shows surprisingly good results for retrieving a single key signature for a given sequence of notes. The described outline cannot be used directly for detection of key signature changes. By analysing the distribution of observed semitone transitions a probability for a key signature change could be derived (*e.g.,* if a significant number of occurrences for more than two semitone transitions has been observed), but this analysis gives no indication about the correct position of a key signature change .

Because the main focus of this thesis was not on key signature we did not evaluate in detail the aspects about the quality and requirements for the input data for using this approach. But we assume that if this approach would be applied to atonal music the inferred key signature would minimise – if possible at all by using standard key signatures – the number of additional accidentals.
We also assume that it should be possible to apply the algorithm only to segments of a performance for inferring the key of these segments. The segments and their boundaries could be inferred in advance by

[3] Correct number of accidentals.

id	transition		
0	c	↔	$d^{\flat}$
0	$b^{\sharp}$	↔	$c^{\sharp}$
1	$c^{\sharp}$	↔	d
2	d	↔	$e^{\flat}$
3	$d^{\sharp}$	↔	e
4	e	↔	f
5	$e^{\sharp}$	↔	$f^{\sharp}$
5	f	↔	$g^{\flat}$
6	$f^{\sharp}$	↔	g
7	g	↔	$a^{\flat}$
8	$g^{\sharp}$	↔	a
9	a	↔	$b^{\flat}$
10	$a^{\sharp}$	↔	b
10	$b^{\flat}$	↔	$c^{\flat}$
11	b	↔	c

Table 6.2: Possible semitone transitions.

major key	minor key	transition 1	transition 2
C	a	e ↔ f	b ↔ c
G	e	b ↔ c	$f^{\sharp}$ ↔ g
D	b	$f^{\sharp}$ ↔ g	$c^{\sharp}$ ↔ d
A	$f^{\sharp}$	$c^{\sharp}$ ↔ d	$g^{\sharp}$ ↔ a
E	$c^{\sharp}$	$g^{\sharp}$ ↔ a	$d^{\sharp}$ ↔ e
B, $C^{\flat}$	$g^{\sharp}$, $a^{\flat}$	$d^{\sharp}$ ↔ e, $e^{\flat}$ ↔ $f^{\flat}$	$a^{\sharp}$ ↔ b, $b^{\flat}$ ↔ $c^{\flat}$
$F^{\sharp}$, $G^{\flat}$	$d^{\sharp}$, $e^{\flat}$	$a^{\sharp}$ ↔ b, $b^{\flat}$ ↔ $c^{\flat}$	$e^{\sharp}$ ↔ $f^{\sharp}$, f ↔ $g^{\flat}$
$C^{\sharp}$, $D^{\flat}$	$a^{\sharp}$, $b^{\flat}$	$e^{\sharp}$ ↔ $f^{\sharp}$, f ↔ $g^{\flat}$	$b^{\sharp}$ ↔ $c^{\sharp}$, c ↔ $d^{\flat}$
$A^{\flat}$	f	c ↔ $d^{\flat}$	g ↔ $a^{\flat}$
$E^{\flat}$	c	g ↔ $a^{\flat}$	d ↔ $e^{\flat}$
$B^{\flat}$	g	d ↔ $e^{\flat}$	a ↔ $b^{\flat}$
F	d	a ↔ $b^{\flat}$	e ↔ f

Table 6.3: Standard key signatures and corresponding pairs of characteristic semitone transitions.

analysing changes of other features (*e.g.,* rhythm, tempo, voice structure) of the sequence of notes (see Section 3.4). If it could be shown that, for example, in a major key the 7 to 8 semitone transition occurs significantly more often than the 3 to 4 transition, then it should also be possible to infer the key gender (*i.e.,* minor/major) by analysing the ratio frequency of occurrence (in the input data) between the two semitone steps of the inferred key signature.

6.2.3 Results

The key detection module of our current implementation of `midi2gmn` has been evaluated with a selection of performance MIDI files and quantised MIDI files. The files have been processed with standard settings for the voice separation module of `midi2gmn` and the inferred key signature (number of accidentals) has been compared to the key signature of the original score. The result of the evaluation and also the result obtained with the Melisma key finding module is shown in Table 6.4. Here cases where Melisma inferred a wrong key but a correct key signature (*e.g.,* A minor instead of C major) have been evaluated as correct. It should be noted that Melisma's key detection module inferred a lot of harmonical correct changes of the key, but especially for the Bach examples, these changes are not indicated explicitly in the original scores. For a decision about exporting these inferred changes of the tonal centre explicitly to a score additional heuristics or additional user input would be required.

For the *Sinfonia 7* the correct key signature is E Minor (single ♯), `midi2gmn` infers here a key signature with two ♯ (*i.e.,* D Major or B Minor), because the melody includes a major number of $c^{\sharp}$ notes. The key detection for the melody voice of *Take Five* showed the largest deviation between original key signature and inferred key signature (for `midi2gmn` and even more for the Melisma system). Because of a large number of alterations the key detection module calculated equal scores for the key signatures $D^{\flat}$, $G^{\flat}$, and $B^{\flat}$. Because our current implementation prefers in this case the key signature with the lowest number of accidentals, $B^{\flat}$ was selected automatically (see Figure 5.11). In the interactive mode the user would have been asked for selecting his preferred key signature.

In general the evaluation shows that our somehow simple approach of analysing the distribution of pitch transitions gives similar results than more complex approaches, such as for example, used in the Melisma system.

		midi2gmn		Melisma	
collections	# files	# errors	max error	# errors	max error
Bach *Inventio 1–15*	15	2	1	2	1
Bach *Sinfonia 1–15*	15	1	1	4	1
Bach *Well-Tempered Clavier Book II:*					
Prelude and Fugue 1–12	24	0	0	1	1
Mozart *Clarinet Quintet* KV 581, set 1–4	4	2	1		
single files	score		inferred		inferred
Debussy *Suite Bergamasque*					
Prelude	F/Dm	0	F/Dm	0	F
Minuet	C/Am	1	G/Em	-	-
Clair de Lune	D♭	0	D♭	0	D♭
Passepied	A/F♯m	0	A/F♯	1	Bm
Brubeck *Take Five* (melody)	E♭m	1	B♭/Gm	1	G♭
Beethoven *Sonata Nr. 20, Op. 49, 2*	G	0	G	0	G
Bach *Minuet in G*	G	0	G	0	G
Dvořak *Humoresque*	G	0	G	0	G
Chopin *Op. 6, Mazurka 1*	F♯m	0	A/F♯m	1	Bm
Chopin *Op. 67, Mazurka 2*	Gm	0	B♭/Gm	0	B♭
sum	68	7		10	

Table 6.4: Evaluation of the key detection of `midi2gmn` and Melisma. The entries for *Suite Bergamasque, Minuet* are missing for the Melisma system because it could not process the MIDI file.

6.3 Pitch Spelling

Another general issue related to the key signature is correct pitch spelling of notes. If the input data contains pitch information encoded only in semitone steps – omitting explicit accidental information – then there might always ambiguous situations occur where different pitch spellings for a single note are possible. If, for example, the pitch class c' in the first octave is encoded with pitch = 60 and the pitch class d' with pitch = 62 than, without any additional context information, it is not decidable if pitch = 61 should denote c♯' or d♭'. Such different labelled notes which represent the same pitch class are called *enharmonic equivalent*. As shown in Figure 6.2 there also exist enharmonic equivalent pitch spellings for natural notes, such as g'. The correct pitch spelling for a note depends on the melodic and harmonic

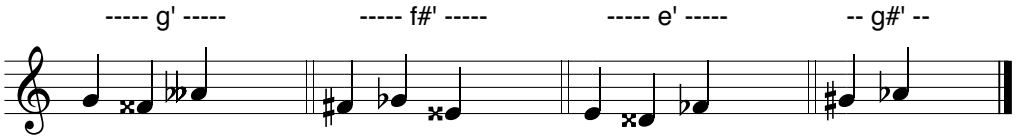


Figure 6.2: Example for different correct pitch spellings for four pitch classes.

context and on the current key signature. The issue of correct pitch spelling can be separated in two categories: correct pitch spelling of (monophonic) melody lines and correct pitch spelling of chords.

6.3.1 Existing Approaches

There exist several approaches for correct pitch spelling in the literature. In the following we give a short overview about some of these approaches and then show the details on the approach developed and implemented in the context of this thesis.

Meredith describes and compares in [Mer03] the pitch spelling approaches of Temperley ([Tem01], [TS]), Cambouropoulos ([Cam01b]), Longuet-Higgins ([LH76]), and his own implementation. All these ap-

proaches create pitch spelling for monophonic lines and do not evaluate the pitch spelling of chords as a separate issue.
Temperley proposes a preference-rule-based approach combined with dynamic programming for finding the overall optimal solution. The model (as implemented in the Melisma system) consists of three rules: *Pitch Variance Rule* for labelling nearby events, so that the resulting pitch classes are close together in the 'line of fifths';[4] *Voice Leading Rule* for correct pitch spelling of chromatic scales; *Harmonic Feedback Rule* for consistent pitch spelling of harmonic context. His model requires that a piece must be segmented into short segments before starting the pitch spelling module.
Similar to the approaches of Longuet-Higgins, Cambouropoulos, and Temperley, Meredith's approach (called ps13) also evaluates only the melodic pitch spelling of monophonic lines. Here an optimisation approach is used for creating correct pitch spellings of melodic lines. In a second step of his model then neighbourhood conflicts (*e.g.,* chromatic scales) are corrected separately.

Meredith tested all four approaches (he implemented Longuet-Higgins and Cambouropoulos approaches himself) with the complete set of MIDI files of Bach *Well Tempered Clavier Book I*. His evaluation showed that the results for the implementation of the approach proposed in ([Cam01b]) are significantly worse than for the other approaches.

Another recent work in the area of key detection and pitch spelling has been proposed by Chew in [CC03]. Here the key signature information inferred by the Spiral Array approach ([Che02], see also Section 6.2.1 for a short introduction) is used for inferring correct pitch spelling. Different from the other approaches described above, this model also evaluates the harmonic context and should therefore be able to provide correct pitch spelling inside of chords.

6.3.2 A Rule-Based Approach

Because pitch spelling is not in the main focus of the HEISENBERG project, respectively this thesis, in the current implementation only a simple rule-based approach is used, which provides reasonable results in very most cases. This model works in two steps:

1. All regions with ascending or descending chromatic scales are recognised. For the ascending scales then all pitch information is normalised to a pitch spelling using naturals and sharps where, depending on the current key signature, also double sharps might occur; the descending scales are normalised to pitch spellings using naturals, flats, and double flats. This heuristic minimises the number of additional accidentals in chromatic scales (see Figure 6.3) which is intuitively done by human transcribers and score writers. This step should correct the same errors that are corrected

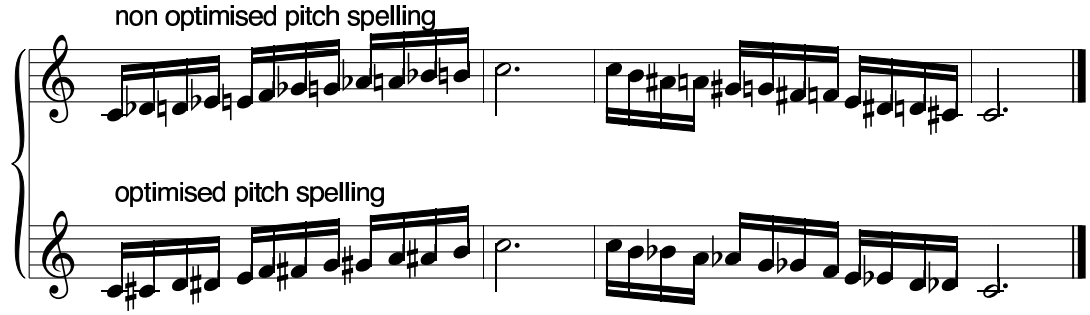


Figure 6.3: Pitch spelling of chromatic scales.

in part II of the approach described in [Mer03]. As shown by Temperley ([Tem01], p. 129, Figure 5.14) there exist exceptions where composers (*e.g.,* Beethoven) do not follow this rules and use, for example, a '♭' for the minor 7th step of a scale even in ascending chromatic scales. These exceptions are currently not respected by our implementation.

[4] The line of fifths is similar to the circle of fifths except that it extends infinitely in either direction ([Row01]).

2. For all other regions where no chromatic context can be detected, the pitch spelling is created from a look-up table for each of the 15 standard key signatures used in Western tonal music.[5] The data of the default look-up tables is similar to the *Pitch Variance Rule* used in the Melisma system (see [Tem01], p. 125). It should be noted that the pitch spelling is independent from major or minor key signature. Only the number of accidentals of the key signature must be evaluated. See Section A.3 for more details about specifying look-up tables for the current implementation of `midi2gmn`.

   Where this table look-up approach produces usable results for melody lines, it might fail for correct pitch spelling of chords because the harmonic context is not evaluated. In the current implementation chords are normalised to use either flats (♭) or sharp (♯) accidentals for all notes of the chord, but no mixture of flats and sharps. This rule will not always result in the (musicological) correct spelling for chords For example, a C minor seventh chord with augmented fifth ($Cm^{7/\sharp 5}$) should be spelled correctly as c, $e^{\flat}$, $g^{\sharp}$, $b^{\flat}$. The spelling preferred by our approach can ensure at least somehow readable scores: the here inferred incorrect version of $Cm^{7/\sharp 5}$: c, $e^{\flat}$, $a^{\flat}$, $b^{\flat}$ is equivalent to $A^{\flat 9}$.
   We assume that only pitch spelling approaches which evaluate harmonic context (*e.g.,* [CC03]) and also infer harmonic chord relations, can create correct pitch spellings for chord notes. Approaches that evaluate only horizontal note-to-note relations might fail in this cases.

In a future implementation of `midi2gmn` the simple rule-based approach could easily be replaced by one of the discussed advanced pitch spelling algorithms. A possible implementation could also be done as a standalone GUIDO-to-GUIDO tool, using arbitrary GUIDO files as input and converting them to equivalent files including correct spelled pitch information.

6.4 Ornaments

Beside the *real* performance notes (in following called *melody* notes) – these are explicitly written in a score with a default notehead size and their duration increases the overall score time – a musical piece can also include ornamental notes. These notes are denoted by special symbols – attached to melody notes or chords – or noteheads with a smaller size. Examples for standard types of ornaments are grace notes, turns, trills, mordents, or glissando. It depends on the performers expression and the style of music how the different ornaments should be performed in detail (*i.e.,* speed, intensity, number, and order of notes). An algorithm for inferring score information from performance data should be able to detect sequences of ornamental notes and replace them by the correct ornament score symbol attached to a root note of the ornament.

A detection and filtering of ornamental notes before the quantisation and tempo detection (but after the voice separation) increases the quality of these following steps (see Chapter 4, Chapter 5). Therefore filtering the ornamental notes can be seen as equivalent to filtering *rhythmical noise* from the input data. The task of inferring ornaments can be separated in different steps:

1. Detect a sequence of ornamental notes.

2. Infer an ornament type (ornament symbol) for the detected sequence.

3. Select a root note for the ornament, depending on the inferred ornament type, and adapt the duration or the onset time of the root note.

The most significant, common feature of all mentioned ornaments is the very short duration of the single ornamental notes. Therefore step 1. can be solved by searching the input data for notes with a duration below a certain threshold. This threshold can be estimated by an analysis of all note durations of the performance data. Beside the absolute duration exist several other criteria which identify an ornamental note:

[5] Possible key signatures based on the circle of fifths: $C^{\flat}$, $G^{\flat}$, $D^{\flat}$, $A^{\flat}$, $E^{\flat}$, $B^{\flat}$, F, **C**, G, D, A, E, H, $F^{\sharp}$, $C^{\sharp}$.

1. The absolute duration of ornamental notes is typically short (see also [WAD+00, WAD+01]). Ornaments can be roughly divided into two groups: acciaccatura with a performance duration independent from the local tempo and appogiatura performance duration depends on local tempo ([DH91], see also [Cam00b]). In the context of this thesis we will focus on the acciaccatura type. Ornamental notes with a long absolute duration might also occur, but these notes can be written without any side effect to quantisation explicitly as melody notes in a score. From a musicologists view they must be treated as ornaments (*e.g.,* not belonging to a main theme), but without knowing style of music and original theme there are no indications for inferring (and transcribing) any long notes as ornaments.
2. In a given performance the number of ornamental notes is lower than the number of melody notes (*i.e.,* non-ornamental notes). Even in pieces of the barock era where the use of ornaments was very common, for a single performance the total number of melody notes is significantly higher than the number of ornamental notes.
3. For all ornamental notes belonging to a single ornament (*e.g.,* trill, turn), the absolute duration will be similar.

These criteria can be described in mathematical terms:

1. A note m can only become an ornamental note if its performance duration $duration_{perf}(m)$ is not significantly larger than a certain threshold t_{orn}. This can be expressed by a Gaussian window function $p_{absorn}(m) := W_{Gauss}(duration_{perf}(m), t_{orn})$. Where parameter t_{orn} is used as parameter σ for W_{Gauss}.
2. Because the number of ornamental notes is usually small compared to the total number of notes of a piece, the duration of an ornamental note m must be lower than the mean of the durations of all notes:

$$duration_{perf}(m) < mean(\text{all durations}) \\ \Longrightarrow p_{meanorn}(m) := W_{Gauss}(mean(\text{all durations}) - duration_{perf}(m), \sigma) \quad (6.14)$$

3. if note m_i was identified as an ornamental note, with a high probability all successive notes $m_{i+1}, \ldots, m_{i+l}$ with a similar duration belong to the same ornament. Also here a Gaussian window function can be used:

$$\text{note } m_i \text{ has been inferred as an ornamental note} \\ \Longrightarrow p_{succorn}(m_{i+1}) := W_{Gauss}(duration_{perf}(m_i) - duration_{perf}(m_{i+1}), \sigma) \quad (6.15)$$

By evaluating $p_{absorn}, p_{meanorn}$, and $p_{succorn}$, ornamental notes can be filtered from the melody notes of the input data (see Table 6.5 and Table 6.6). In the following $ornament(m) = true$ should denote that note m was inferred as an ornamental note by using the functions defined above.

type	mean duration	std. deviation
glissando	38.918ms	18.195
grace	51,323ms	13.632
trill	88.32ms	22.37
turn	85.71ms	22.15

Table 6.5: Measured performance data.

	ms
mean	78.190
median	77
std. deviation	17.094
min	47

Table 6.6: Typical values for notes played as fast as possible. (skilled player with one hand.)

As shown in [TAD+02] – where the opposite task of correct performance of scored grace notes is discussed – there exist different types of grace notes. Especially the performance duration of shortly played graces

notes played are independent from the performed tempo.
In [Mac02] a method for separating grace notes from melody notes is shown. Here an approach based on Bayesian statistics is used for clustering typical note durations and inferring the cluster with smallest duration as grace notes. The proposed type of clustering note duration works only well with special type of input data (*e.g.,* few duration classes, no tempo drift). It is not clear how many clusters should be used and without any estimation about the performance tempo it is not clear how the initial mean value for a cluster (*e.g.,* an eighth notes cluster) should be selected. Also the algorithm is designed with o focus on single grace notes, so it would require an adaptation for the detection of general ornamental notes.

If a voice separation has been performed before, now the detected sequences of ornamental notes can be grouped to ornaments. If a note m_{i-1} was not identified as an ornamental note and the successive note m_i was identified as an ornamental note – satisfying the constraints shown above – then m_i can be marked as the start note o_{start} of the ornament (*i.e.,* a sequence of ornamental notes):

$$ornament(m_{i-1}) = false \wedge ornament(m_i) = true \Longrightarrow o_{start}(m_i) := true \tag{6.16}$$

Each inferred ornament of a score belongs to a dedicated *root* note to which a corresponding score symbol is attached. With the exception of a special version of a glissando, this root note is always played after the ornamental notes itself. If a note m_i has been identified as a first ornamental note ($o_{start}(m_i = true)$), the root note o_{root} can be characterised as

$$\forall\, j = 0, 1, \ldots, k : ornament(m_{i+j}) = true \wedge ornament(m_{i+k+1}) = false \\ \Longrightarrow o_{root}(m_{i+k+1}) := true. \tag{6.17}$$

For an ordered set of notes $M = \{m_1, \ldots, m_{|M|}\}$, belonging to a single voice, the set $M_{orn} \subseteq M$ of ornaments can be defined as

$$M_{orn} := \{\ m \mid m \in M \wedge o_{root}(m) = true\ \}. \tag{6.18}$$

All sequences of ornamental notes $m_i, \ldots, m_{i+k}$ with $ornament(m_{i+j}) = true$ for $j = 0, 1, \ldots, k$, and $o_{start}(m_i) = true$, and $o_{root}(m_{i+k+1}) = true$ can be removed from M and attached as a set $M_{perforn}$ of performed ornamental notes to the ornament root m_{i+k+1}. The score information (pitch and duration) for the removed series $m_i, \ldots, m_{i+k}$ will not be written explicitly in the transcribed score. Instead m_{i+k+1} will be notated with an attached ornament symbol (*e.g.,* grace, trill, turn) in the score.

For each ornament root $m \in M_{orn}$ respectivly the attached sequence of performed ornamental notes $M_{perforn}(m)$ now the ornament type needs to be inferred. For classifying the ornaments different approaches could be used (*e.g.,* rule-based). For our system we decided to use a k-NN classifier for classifying the feature vectors of the different ornament types.

Each ornament type has different values for a set of typical features (*e.g.,* number of notes, number of different pitches, ambitus, direction of successive pitches, see Equation 6.4) by defining the feature vectors for prototypes of these ornaments a k-NN search on detected ornaments can be done and the correct ornament type can be inferred.

For each ornament type a prototype feature vector and a vector with the weight of the different features can be defined. For example, for a *turn* the number of notes and ambitus is more significant than for a grace note. The k-NN classifier can be implemented in several ways using different measures for the distance d between two n-dimensional vectors. A possible distance measure is the Euclidean distance between two n-dimensional vectors given as

$$d = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}, \tag{6.19}$$

where $X = (x_1, \ldots, x_N)$ is the feature set of a prototype class and Y the feature set of a test class. Because each features might have a different significance for the different prototype classes a weight ω_i can be applied to each feature:

$$d = \sqrt{\sum_{i=1}^{N} \omega_i (x_i - y_i)^2} \tag{6.20}$$

ornament type	Feature id				
	1	2	3	4	5
glissando up	up	large	small	large	many
glissando down	down	large	small	large	many
turn	fuzzy	equal	small	small	few
grace	straight	n.n.	n.n.	n.n.	few
mordent	fuzzy	equal	very small	very small	three
trill	fuzzy	equal	small	small	many

Features

1 pitch direction of ornamental notes;
fuzzy should denote up, down in unknown order
followed by a return to the original pitch

2 interval size between first and last ornamental note
3 typical interval between successive ornamental notes
4 ambitus of all ornamental notes
5 number of ornamental notes (without root note)

Table 6.7: Features of ornament types

The weight vector $\Omega = (\omega_1, \ldots, \omega_N)$ might be used for all prototypes or each prototype vector X_a might have a specific weight vector Ω_a. Beside the feature set for each prototypes class also the features weights Ω must be estimated. For larger number of features and prototypes this should be done, for example, with an genetic algorithm. Because of the small set of prototypes, and the small set of features for the ornament detection it was possible to find a set of weights by manual adjustment.

After inferring the ornament type only step 3 – the correct root note and its duration and onset time – is still missing. This can be solved by a simple rule system shown in Table 6.8. Especially grace notes and mordents can be played in at least two different ways: keep the onset time of the root note and play the ornamental notes before the root note (see Figure 6.4(1)); or start with the ornamental notes at the written onset time of the root note and shift the onset time of the root note (see Figure 6.4(2)).

ornament	root note	duration	onset time
glissando	o_{n+1}	$\text{dur}(o_{n+1})$	$\text{onset}(o_{n+1})$
turn	o_{n+1}	$\text{offset}(o_{n+1}) - \text{onset}(o_1)$	$\text{onset}(o_1)$
grace 1	o_{n+1}	$\text{dur}(o_{n+1})$	$\text{onset}(o_{n+1})$
grace 2	o_{n+1}	$\text{offset}(o_{n+1}) - \text{onset}(o_1)$	$\text{onset}(o_1)$
mordent	o_{n+1}	$\text{offset}(o_{n+1}) - \text{onset}(o_1)$	$\text{onset}(o_1)$
trill	o_{n+1}	$\text{offset}(o_{n+1}) - \text{onset}(o_1)$	$\text{onset}(o_1)$
Inferred ornamental notes (with short durations): $o_1 \ldots o_n$			
First non-ornamental note (longer duration) after ornament: o_{n+1}			

Table 6.8: Rules for onset time and duration of ornament root notes.

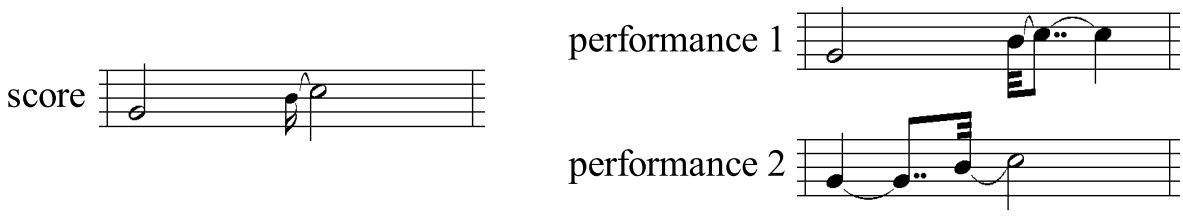


Figure 6.4: Difference between score description and performance of grace notes.

	detected	not detected	false positive	wrong type
Bach *Minuet in G*	5	1	0	0
Beethoven *Sonata Nr. 20* mm. 1–28	6	0	1	0
Bach *Inventio I*	8	0	0	0
Brubeck *Take Five* (melody voice)	0	0	0	0

Table 6.9: Evaluation of the ornament detection module.

The decision how a grace note should be performed depends on style of music and the composer. It can be tried to infer the intended type by investigating the relation between grace note and root note intensity. A higher intensity should usually indicate the *written* onset time of the root note because a more stressed note would be inferred as the *beat*. But even if inferred incorrectly the error (shift of onset time) should be small enough that it can be corrected later by the quantisation module.

Results

For the evaluation of an ornament detection approach we can distinguish between three types of errors: not detected ornamental notes; melody notes incorrectly inferred as ornamental notes (false positive errors); and incorrect inferred ornament type (*e.g.,* grace instead of trill). Similar to the evaluation of the tempo detection and quantisation module, here again the inferred ornament types must be compared to the ornament types inferred by an human transcriber instead a simple comparison to the original score. Depending on the style of music and the amount of musical expression the performer might have added ornaments that are not included in the original score or the performer might have chosen an ambiguous way of playing the ornament.

The evaluated performance of Bach's *Minuet in G*, for example, includes a number of ornaments where the original score included no ornaments at all. Table 6.9 shows the results for some selected files. As already shown in Figure 5.6 the performance also includes an ornamental note with a duration very close to the duration of melody notes which therefore could not be detected as an ornamental note. As shown by the evaluation of the *Sonata in G* and the *Inventio I* the identification of ornamental notes based on the relation between absolute duration of a single note and the mean of all durations works even for pieces that include many short notes.

In general the current implementation of our ornament detection approach is able to distinguish correctly between ornamental and melody notes and between different types of ornaments. For certain combinations of ornaments (*e.g.,* a trill ending with two grace notes) the correct type might be hard to infer. Because we developed the ornament detection module with a main focus on 'noise reduction' for tempo detection and quantisation (by filtering the ornamental notes) we omitted a more detailed evaluation of this module. We assume that it should be possible to improve the output quality of the ornament detection module by optimising the features weights (currently manually adjusted) by using, for example, a genetic algorithm.

6.5 Intensity Marking

From the intensity information of the performance data a dynamic profile can be inferred. A musical score includes different types of dynamic information:

- Standard information, such as for example, piano (*p*) or forte (*f*) indicating a general average intensity (or volume) for regions of a score.
- Accentuation information for single notes or short passages indicated, for example, by a sforzando (*sfz*) below the staff or by accent symbols attached to single notes, such as '^' or '<'.
- Slow changes of the intensity during a passage as indicated by *crescendo*, *diminuendo*, or *decrescendo*.

The *intelligent* transcription of intensity profiles is hardly mentioned in the literature. Similar to the interpretation of tempo indications, the interpretation of score intensity information depends even more on the intention of the performer as well as on the style of music. Especially the accentuation and the intensity changes (*e.g., crescendo*) can be performed in different ways. Analogous to the timing information of a performance the intensity information will include no completely exact values. It must be assumed, that a human player is not able to press a key always with exactly the same speed, pressure, or velocity.

As defined in Section 1.4.1 the intensity of a note m should be available as a float number in range $(0:1]$ where 1 indicates a maximum intensity. This range can be divided into the standard musical categories of intensity (dynamics) indications as shown in Table 6.10, where the limits between the categories must be assumed to be only unsharp and approximate values. Single notes or a sequence of only a few notes with an intensity slightly out of the range of an intensity category should not be indicated with additional intensity symbols in a score. Only significant changes of the intensity should be indicated by accentuation markings (for short passages or single notes) or by a dynamic marking (for larger regions). So the main task during inferring an intensity profile is to cluster successive notes by their intensity information and assign then each cluster an intensity category.
Crescendi or diminuendi can be inferred by a regression analysis of the input data. This feature in not in focus of this thesis. For an efficient clustering the floating average approach described in Section 3.4.1 could be used in an future version of our system. The transcription/inferring of dynamic information has been implemented only as a prototypical approach within `midi2gmn`.

	$(0,1]$ range	MIDI range
ppp	> 0	> 0
pp	> 0.15	> 19
p	> 0.31	> 39
mp	> 0.46	> 59
mf	> 0.62	> 79
f	> 0.78	> 99
ff	> 0.91	> 115
fff	$= 1$	$= 127$

Table 6.10: Possible mapping between dynamic information in score intensity indications (dynamic markings) and MIDI velocity information.

6.6 Slurring

One basic meaning of slurs in graphical scores is to indicate that the slurred notes should be played legato. Other meanings of slurs are markup of phrases or indicating performance related requirements depending on the used instrument (*e.g.,* for brass instruments the control of air, for string instruments control of bow). The actual meaning of a slur is not indicated explicitly in a score, it must be inferred by the performer. In the context of this thesis only the first meaning of slurs – indicating legato – is relevant.

If successive notes are played legato then there should be no gap between offset of the first note and the onset time of the following note. On polyphonic instruments, such as piano or organ there even will be small overlaps between the successive notes, because the performer presses the next key before releasing the first one. The property of 'no gap between successive notes' can therefore be used to detect slurred notes in performance data – after voice separation, before tempo detection. The detected slurred passages can be marked and a slur tag (*i.e.,* \slur) can be applied to them when creating the GUIDO output file.

The slurs detected with this method might be different from the original slurring of a given score. This can be caused by different reasons:

- The performer has made a mistake – more or less legato notes than indicated by slurs.
- The slurs in the score were no legato slurs.
- The performance (MIDI file) is the export of a notation software which did not care for performance-like note durations at all.
- There exist several 'correct' solutions for slurring.

Resolving errors caused by the first two topics can only be done with knowledge about style and musicological background which is far beyond of the scope of this thesis. If the input data contains only static

note durations (*e.g.,* 80% or 100% of the written note duration) which is often the result of creating MIDI files by export from notation software, here also no slurring can be inferred without additional information. Several 'correct' solutions of slurring can exist because two successive slurred passages might be written as a complete slurred passage or grouped in another way. An evaluation of phrase marking ([TSH02]) showed how different human listeners percept musical phrase boundaries.

6.7 Staccato

Beside slurs there exist further symbols marking durational articulations of notes in graphical scores. Different from legato slurring where note durations are slightly increased, most durational articulation marks, such as marcato or staccato, decrease the performed duration compared to the written note duration. The interpretation of specific articulation marks usually depends on the style of music, the composer's intention, and the performer's intention. The inverse process of inferring *the* written articulation mark from the performance data therefore would need information about these meta information which is out of scope of this thesis.

An exception is the staccato marking. Notes with an attached staccato symbol – a dot above or below the notehead – are supposed to be played very short compared (*e.g.,* half of the notated duration) to their score duration (*e.g.,* half of the notated duration). Beside for reasons of style, their usage reduces the complexity and increases the readability of scores as shown in Figure 6.5. Staccato markings are usually used for score durations equal or shorter than a quarter note, for longer notes they are rather uncommon.

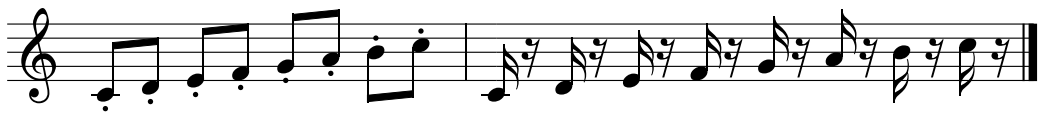


Figure 6.5: Complexity of scores with (left) and without using staccato articulation marks (right).

By evaluating the performed IOI and duration of a note m of the performance data (after voice separation), its duration after the tempo detection and quantisation, and using the features of staccato played notes shown above, the staccato notes can be detected and marked with a \stacc tag in the GUIDO output file. For a note m in a single voice – consisting of non-overlapping notes – it can be assumed that $duration_{perf}(m) \leq IOI_{perf}(m)$ and also $duration_{score}(m) \leq IOI_{score}(m)$. Using the articulation rules for the correct interpretation of staccato note m it can be defined:

$$m \text{ was performed as staccato } \implies duration_{perf}(m) \ll IOI_{perf}(m) \tag{6.21}$$

The opposite direction '$\Leftarrow$' cannot be assumed, because the right side of the equation will be true for any arbitrary note followed by a rest. By comparing the performance and (inferred) score IOI and duration of m a criterion for the opposite direction can then be defined:

$$duration_{score}(m) \leq \frac{1}{4} \wedge \frac{duration_{perf}(m)}{IOI_{perf}(m)} \ll \frac{duration_{score}(m)}{IOI_{score}(m)} \implies m \text{ was performed staccato.} \tag{6.22}$$

Here again, in general, the '$\Leftarrow$' direction cannot be assumed, because the quantisation module might have decided to quantise the duration close to the performed duration. As shown in Section 5.3, during transcription there might be no indication to not allow the combination of a quaver note followed by an eighth rest, which was actually a dotted crotchet in the original score.

Epilogue

"Musik ist der vollkommene Typus der Kunst: sie verrät nie ihr letztes Geheimnis."
Oscar Wilde

Generating a readable score from given performance data should be the main target of a computer-based transcription system. The rhythmic information of the generated score should be as close as possible to the given performance data but at the same time the inferred score should avoid as much as possible complex structures, which would reduce its readability. As shown in the results sections of the previous chapters there are several general issues in the context of the evaluation of the output of transcription systems: the limited number of publicly available adequate test files; the huge amount of manual work caused by multiple ambiguous correct transcriptions for a single performance; and the lack of standard test data sets for comparing and evaluation the results obtained with different systems.

As previously described, our approach has been implemented as an usable system named `midi2gmn` . It has been implemented as a command line tool written in ANSI C++ and can therefore be compiled and used on any standard operating system. Our implementation can directly read different types of input file formats (*e.g.,* MIDI, GUIDO files) and creates output files in Basic GUIDO syntax containing the transcribed score level information. These files can be converted into graphical scores using, for example, the online GUIDO NoteServer or the standalone GUIDO NoteViewer. In the case of MIDI input files, `midi2gmn` creates additionally to the score level transcription a one-to-one conversion of the original MIDI data in Low-Level GUIDO syntax. These unprocessed version of the input data could be used to create the test library in Low-Level GUIDO syntax as proposed in Section 1.5. It should be a rather straight forward task to convert other proprietary performance data file formats (*e.g.,* ASCII note lists) into a GUIDO dialect that can be parsed by our system (see Section 1.4.2). Beside a command line version, we also provide an online version of our system. This service (located at `http://www.noteserver.org/midi2gmn/midi2gmn.html`) allows the online conversion of MIDI files into GUIDO files which then directly can be converted into graphical scores using the GUIDO NoteViewer. Different to the command line version of `midi2gmn` where settings must be specified in an initialisation file, the online interface includes some graphical control elements (*e.g.,* edit boxes, radio buttons) for specifying user definable settings. but therefore the online version cannot provide any interactivity features.

With the current implementation of our system we could proof that computer aided transcription benefits from the pattern-based models and the interactive features proposed in this thesis. Different from the large number of existing approaches addressing different issues in the context of computer aided transcription, our system tries to estimate the overall accuracy of the input data (performance accuracy) in advance and uses this information during tempo detection and quantisation for adjusting thresholds, the size of search windows, and the resolutions of metrical grids. It also tries to detect automatically potential errors and asks the user for feedback when these cannot be corrected automatically. Also different from many other approaches our system allows the creation of different types of transcriptions of a single performance (*e.g.,* different types of voices separations; prefer binary or ternary durations during quantisation) by changing user definable, intuitive settings.

A general issue respectively disadvantage of the current implementation is the restriction to the command line interface for any interactivity between user and system during runtime. Clearly, the current interactive command line interface can only be used for proof of concept. In future versions especially the user interaction (including the display of unquantised performance data) should be further improved. Another disadvantage of the current implementation is the fact that the specified settings become always applied to the complete input performance data. An improvement of this situation would also significantly increase the usability of our system.

Future Work

Because the main focus during the implementation of the current system was rather on correctness and proof of concept than on optimised runtime and an high end user interface, each of the system's modules includes certain parts which could be improved in future versions of the system.

Voice Separation

Beside a further improvement of the runtime speed of our voice separation module it would be beneficial to integrate some interactivity to this module. Currently the user has to specify the parameters for the voice separation in advance (or use the defaults) and these become applied to the complete performance. With a GUI-based implementation it should be possible to select only regions of the performance and it also should be possible to see the result of changes of the parameters (*e.g.,* controlled via sliders) in real-time. This would allow a more intuitive adjustment of the voice separation parameters. Another future direction is the optimisation of the default settings for these parameters using machine learning or state of the art optimisation approaches, such as genetic algorithms. Because of the ambiguities in musical scores and the lack of standardised 'normal forms' the automatic assessment of the quality or 'correctness' of an inferred voice separation is a very complex and potentially impossible task.

Similarity Analysis, MIR

As shown in Section 3.3 we implemented a prototypical version of the MusicBLAST algorithm for musical input data. As expected this model can retrieve repeated, approximate patterns of a performance. It would be a major improvement if the quantised version of the most significant patterns could be automatically added to the pattern database (used for quantisation and tempo detection). This would require that two issues are solved: the retrieved patterns need to be clustered and a prototype for each cluster must be identified; the prototype should be quantised (including tempo detection) and added to the database. Similar to the general behaviour of our system here again the system should automatically distinguish between simple input data which could be processed automatically and complex input data where it is required that the user might approve or correct the quantisation before adding the pattern to the database.
As mentioned in Section 6.1 and Section 6.2 the output of a structural analysis (through self-similarity analysis) should be used to infer features, such as key- and time signature, for each inferred segment separately. Assuming that the inferred segment borders are correct in a musicological sense, this method could be used to infer changes of key- and time signature.

Quantisation and Tempo Detection

The pattern-based parts of these modules could be improved by integrating the typical deviations between performance data and score data for each note into the pattern databases. If the systems could 'learn' these typical deviations for specific styles of music or performers it would be possible to train these modules to specific styles and/or specific performers. Similar to training an OCR system to the handwriting of a person the quantisation and tempo detection results could be improved or the system could be used to train a student by detecting his typical errors.
Another improvement would be in automatical learning of the probabilities for pattern transitions. Similar to the binclass approach for note durations we assume that there also exist typical distributions for the use of patterns. If, for example, pattern A has been used very often in a performance then the chance that pattern B is also part of the original score is very low (*e.g.,* son-clave to son-clave, rumba-clave to rumba-clave). Instead of the implemented pattern matching of complete patterns it might also be possible to find *rules* how a given pattern could be extended or how new patterns could be created by combinations (*e.g.,* stratification) of other patterns included in the database. Also the inverse task of minimising the pattern database by decomposition of existing patterns into layers of simple patterns might be an interesting direction for further research. The general way of pattern matching could also be further improved by using other classification techniques, such as k-NN classifiers or support vector machines (SVM). Because of the different length of the pattern (especially for tempo detection) it might

be a non-trivial task to use these techniques (usually applied to equal length vectors) to these types of rhythmic patterns.

User Interface And Settings

The current implementation of our system reads all user definable settings from an initialisation file (see Section A.3 for a description). If needed it can also request additional user input via the command line interface. The online version of our system (`http://www.noteserver.org/midi2gmn/midi2gmn.html`) already provides an HTML-based user interface for specifying many parameters in edit fields and drop down boxes.
An improved, future version of `midi2gmn` should include a graphical, interactive user interface (GUI) where the user can specify the required settings and the system can ask for additional input. A powerful user interface also should include capabilities for displaying quantised graphical scores and an adequate piano roll display for displaying unquantised scores. It should be possible to select only specific regions of a performance and process only the selected data with certain settings for a desired type of output, where other regions might be processed separately. A speed optimised implementation of our system would allow to see the result of different user settings in real-time and intuitively adjust them to the optimal settings. The integration of our system into a powerful notation software package which provides the here required editing features (*e.g.,* NoteAbility) seems to be a promising direction for implementing such a type of user interface. Beside the more comfortable input of settings, a graphical user interface could be used to improve the handling of the interactive features of our system.
As shown in Section 4.3.4, Section 4.3, and Section 5.4 the system will ask the user for clarification in ambiguous situations. In the current command line based version these functions have been implemented prototypically: the user gets prompted for entering an onset time position and the correct duration for the corresponding note. With a GUI the system could display the piano roll notation of the detected error and the user might just select one or several notes and specify the correct score durations by a single mouse click.

With the evaluation of our system we could show that there exist compositions and performances which can be transcribed automatically but that there also exist performance files that are very hard to transcribe automatically because of a large amount of ambiguities. In general, there is some evidence that a fully automated transcription of musical performance data into musical scores might not be possible for arbitrary input data. An adequate transcription system should therefore automatically detect if the input data is too complex for a completely automatic transcription. In this case such a system might ask the user for additional information about the input data to solve ambiguous situations.For tempo detection, beat tracking, and time signature detection our tests showed some evidence that there exist compositions where an approach based only on the analysis of rhythmic features cannot create correct transcriptions. For this type of compositions – where the metrical information is induced by harmony and/or melody – future approaches should therefore try to evaluate the melodic and harmonic information in addition to the rhythmic information evaluated by the current models. Such approaches should be designed in a way that they can automatically detect if a piece or performance includes any harmonical features which can be evaluated (*e.g.,* standard chord progressions). Otherwise they would be restricted to be used only for pieces of certain styles or type (*e.g.,* polyphonic data).
In general we would propose that future computer-based transcription systems should be designed as adequate, interactive, and flexible systems as shown above. Because there seems to be – at least in some styles of compositions – a correlation between the perception of rhythm and the perception of melody and harmony these features should be evaluated even during rhythm transcription (*e.g.,* tempo detection, quantisation). For complete transcription systems the usability depends highly on the user interface but also on the used output file format. For the implementation of such systems file formats should be preferred that are capable to represent natively all inferred score level information and that can be rendered into graphical scores or converted into other formats. Because of the high amount of ambiguity in the relation between performance and corresponding score level information the automatic evaluation of such systems might remain a key issue in the future. We assume that in the future the functionality and usability of automatic transcription system can be further increased. But there might always exist types of performances which cannot be transcribed automatically – even a trained, educated human listener cannot always be completely sure that he decoded the composer's intention correctly.

A Appendix

A.1 Dynamic Programming for String Matching

"String matching by dynamic programming (DP) uses the *edit distance* concept, where the edit distance are the costs for changing a sequence $a = a_1, \ldots, a_{|a|}$ into a sequence $b = b_1, \ldots, b_{|b|}$" ([SMW98]). A sequence consists of symbols (*e.g.,* characters, DNA symbols, musical notes, pattern) and an *edit operator* gives the costs of changing on symbol into another. DP for string matching goes back to Sankoff and Kruskal 1981. For creating the DP matrix (or DP table) with size $(|a| + 1) \times (|b| + 1)$ the following recurrence equation is used:

$$d_{i,j} = \min \begin{cases} d_{i-1,j} + w(a_i, \emptyset) \\ d_{i-1,j-1} + w(a_i, b_i) \\ d_{i,j-1} + w(\emptyset, b_j) \end{cases} \quad 1 \leq i \leq |a| \text{ and } 1 \leq j \leq |b|, \tag{A.1}$$

with $w(a_i, b_i)$ is the cost of substituting element a_i with b_j, $w(a_i, \emptyset)$ is the cost for inserting a_i, $w(\emptyset, b_j)$ is the cost for deleting b_j,

The initial conditions are

$$\begin{aligned} d_{0,0} &= 0, & \text{(A.2)} \\ d_{i,0} &= d_{i-1,0} + w(a_i, \emptyset), i \geq 1, & \text{(A.3)} \\ d_{0,j} &= d_{0,j-1} + w(\emptyset, b_j), j \geq 1. & \text{(A.4)} \end{aligned}$$

The entry $d_{i,j}$ gives now the accumulated distance of the best alignment ending with a_i and b_j; in particular, $d(|a|, |b|)$ gives the edit distance for the optimal alignment between sequence a and b. For a detailed description of dynamic programming for string matching see also [Gus97].
In general string matching by DP consists of three stages:

1. Generating a local scoring matrix for the costs between any possible two symbols (*e.g.,* characters) of the two sequences. Instead of a scoring matrix (*e.g.,* PAM matrix as shown in Section A.15) also a scoring function can be used.
2. Generating the DP table d according Equation A.1. The local scoring matrix defined in step 1 is used to calculate the result of the cost function w.
3. If in addition of the edit distance also the alignment itself is of interest a traceback in the DP matrix d from $d(|a|, |b|)$ to $d(0, 0)$ along the path with minimal costs must be performed.

For the traceback we define a set N of the possible neighbours of a cell $\langle i, j \rangle$ as

$$\begin{aligned} N_{i,j} &= \{\langle i-1, j\rangle, \langle i-1, j-1\rangle, \langle i, j-1\rangle \mid 1 \leq i \leq |a|, 1 \leq j \leq |b|\} & \text{(A.5)} \\ N_{i,0} &= \{\langle i-1, 0\rangle \mid 1 \leq i \leq |a|\} & \text{(A.6)} \\ N_{0,j} &= \{\langle 0, j-1\rangle \mid 1 \leq j \leq |b|\} & \text{(A.7)} \\ N_{0,0} &= \{\} & \text{(A.8)} \end{aligned}$$

For a cell $\langle i, j \rangle$ the best neighbour (used for the traceback) can be retrieved by a function

$$n(\langle i, j\rangle) = \arg \min_{b \in N_{i,j}} \{d(b)\},\ 0 \leq i \leq |a|,\ 0 \leq j \leq |b|, i + j > 0, \tag{A.9}$$

where $d(\langle i,j\rangle) = d_{i,j}$ (*i.e.,* an entry of the DP matrix as calculated in step 2).
Using the best neighbour function, we can formalise the resulting alignments of the traceback as:

$$align_a(a,b,i,j) = \begin{cases} align_a(a,b,i-1,j) \circ a_i, & \text{if } n(\langle i,j\rangle) = \langle i-1,j\rangle, \\ align_a(a,b,i-1,j-1) \circ a_i, & \text{if } n(\langle i,j\rangle) = \langle i-1,j-1\rangle, \\ align_a(a,b,i,j-1) \circ \text{gap}, & \text{if } n(\langle i,j\rangle) = \langle i,j-1\rangle, \end{cases} \tag{A.10}$$

$$align_b(a,b,i,j) = \begin{cases} align_b(a,b,i-1,j) \circ \text{gap}, & \text{if } n(\langle i,j\rangle) = \langle i-1,j\rangle, \\ align_b(a,b,i-1,j-1) \circ b_j, & \text{if } n(\langle i,j\rangle) = \langle i-1,j-1\rangle, \\ align_b(a,b,i,j-1) \circ b_j, & \text{if } n(\langle i,j\rangle) = \langle i,j-1\rangle, \end{cases} \tag{A.11}$$

$$align_x(a,b,0,0) = \square \tag{A.12}$$

with $0 \le i \le |a|, 0 \le j \le |b|, i+j > 0$, and $\square$ should denote an empty word.
In both equations 'gap' should indicated a special symbol which does not occur in the sequences a, b: $\text{gap} \notin a \wedge \text{gap} \notin b$. The result of the *align* functions is now a sequence of characters and gap symbols. It follows

$$\max\{i,j\} \le |align_a(a,b,i,j)| = |align_b(a,b,i,j)| \le i+j, \tag{A.13}$$

with $1 \le i \le |a|$ and $1 \le j \le |b|$.

In [Rap01a] Raphael proposes an improvement of the dynamic programming algorithm by merging several states (nodes of a tree) into so-called *superstates*. By reducing the number of nodes, the complexity for searching for the optimal path through all nodes can be significantly reduced.

A.2 The Inter-Onset Interval

Musical notes are usually represented by their duration and distance to the preceding note of a score. If a note does not start immediately at the offset point of the preceding note this distance is indicated by an explicit rest symbol.
Even if no rest is indicated explicitly in the score, performance data might include small rests between notes where the size of these rests depends on articulation markings (*e.g.,* staccato, legato, tenuto), the instrument (*e.g.,* it is impossible to play any rest between successive notes on a bagpipe), the style of music, or the player's emotion and intention.
Tests showed that the duration of notes can be played much more inaccurately than the onset times of notes without being identified as major performance error by human listeners. In most cases therefore the inter-onset interval (IOI) will be used for analysis instead of the typically more inaccurate duration of a note.

Given a list of notes $M = \{m_1, m_2, \ldots, m_{|M|} \mid onset(m_i) < onset m_{i+1}\}$ sorted by ascending onset times,[1] the inter-onset interval for a note m_i is defined as the distance between the onset time of m_i and m_{i+1}:

$$IOI(m_i) = onset_{i+1} - onset_i, \text{ with } i = 1,2,\ldots,|M|-1 \tag{A.14}$$

Depending on the context the IOI can be calculated in performance time units (ms), $IOI_{perf}(m_i)$, or score time units, IOI_{score}. In the following IOI_i will be used equivalent to $IOI(m_i)$.

Given the IOIs for two successive notes m_{i-1} and m_i, a ratio of IOIs – named IOI ratio – can be defined and calculated in several ways. The standard calculation type is the *real* ratio between the two IOIs:

$$IOIratio1(m_i) = \frac{IOI_i}{IOI_{i-1}}, \text{ with } i = 2,3,\ldots,|M|-1 \tag{A.15}$$

Analogous to the inter-onset interval, in the following $IOIratio_i$ will be used equivalent to $IOIratio(m_i)$. As shown in Figure A.1(a) the values for the IOIratio1 are very dense in the interval for $x = (0,1)$ ($IOI_i < IOI_{i-1}$). For the comparison and evaluation of IOI ratios a normalised measure would be of

[1] We assume that notes with equal onset times have been merged to chords before.

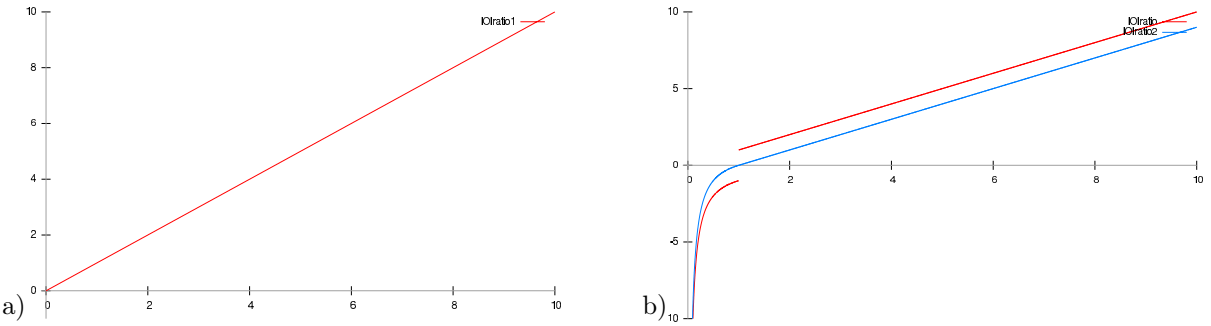


Figure A.1: Types of IOI ratios: a) linear IOI ratio, *IOIratio1*, as defined in Equation A.15 and b) pseudo-semi-log IOI ratio, *IOIratio*, *IOiratio2*, as defined in Equation A.16 and Equation A.18; x-axis = IOI_i/IOI_{i-1}.

advantage. This could be obtained by using $\log(IOIratio1)$ instead of *IOIratio1* itself. Unfortunately the log function also would change the values in the range $(1, \infty)$. Instead of using a log-based calculation a pseudo-semi-log representation for the IOI ratio can be defined:

$$IOIratio_i = \begin{cases} \dfrac{IOI_i}{IOI_{i-1}}, & \text{if } IOI_i \geq IOI_{i-1} \\ \\ -\dfrac{IOI_{i-1}}{IOI_i}, & \text{if } IOI_i < IOI_{i-1} \end{cases} \quad \text{with } i = 2, 3, \ldots, |M| - 1 \tag{A.16}$$

This gives a range of $(-\infty, -1) \cup [1, \infty)$ for the IOI ratio (see Figure A.1(b).
Using the definition in Equation A.16 the comparison of IOI ratios becomes very simple because

$$IOIratio_i = -IOIratio_j \iff \frac{IOI_i}{IOI_{i-1}} = \frac{IOI_{j-1}}{IOI_j}. \tag{A.17}$$

In some special context (*e.g.,* calculation of weight functions as $f(IOIratio)$) a normalised version *IOIratio2* of *IOIratio* can be used:

$$IOIratio2_i = \begin{cases} \dfrac{IOI_i}{IOI_{i-1}} - 1, & \text{if } IOI_i \geq IOI_{i-1} \\ \\ -\dfrac{IOI_{i-1}}{IOI_i} + 1, & \text{if } IOI_i < IOI_{i-1} \end{cases} \quad \text{with } i = 2, 3, \ldots, |M| - 1 \tag{A.18}$$

which gives a continuous range of $(-\infty, \infty)$ to the IOI ratio as defined in Equation A.16.

In cases where the correct duration of the last note $m_{|M|}$ of a sequence M is known (*e.g.,* for patterns) the offset point of this note (*i.e., onset*$(m_{|M|})$ + *duration*$(m_{|M|})$ can be used as the onset time of an additional pseudo note $m_{|M|+1}$ and then the IOI_i and $IOIratio_i$ can be calculated for $i = 1, 2, \ldots, |M|$.

A.3 Parameters And Settings For midi2gmn

The output of the midi2gmn tool can be controlled via some command line parameters and an initialisation file containing settings for the different modules.

A.3.1 Command Line Parameters For midi2gmn

The calling syntax for the current version of midi2gmn is

```
midi2gmn [-help] | [-c"inifilename"] ["inputfilename"]
```

A call with parameter -help will display version and syntax information. If inifilename is omitted fermata.ini will be used as default initialisation file. If the initialisation file does not exist it will be created automatically including default values for all required settings. If no inputfilename is given, first the FILENAME setting of the initialisation file will be evaluated, if no initialisation file exists or if it does not include a FILENAME setting, test.mid will be used as default input filename. If inputfilename is specified without .gmn or .mid as filename extension, .mid will be used as default extension. The output file will be named inputfilename.gmn. If there already exists a file with that name, it will be replaced without any warning! For each input file also a log file including additional information and a one-to-one conversion of MIDI input data in Low-Level GUIDO format (see Appendix A.10) will be created.

A.3.2 The Initialisation File

All user definable settings for the current midi2gmn implementation can be specified in an initialisation file. If not specified with the command line option -c, fermata.ini will be used as default filename for the initialisation file. If the initialisation filename includes no path information midi2gmn searches only in the current working directory for the initialisation file. If the file does not exist it will be created automatically including the required settings and their default value. If no filename is specified as command line parameter and also none in the initialisation file, the user will be prompted for an input filename if midi2gmn runs in INTERACTIVE mode.
A setting is specified as a single line, starting with the setting's name followed by the setting value, where name and value are separated by '='. Remarks and comments must start with a ';' all following characters until the line end are then ignored. The setting names are case sensitive.

In the remainder of this subsection all settings and the default parameters are described.

FILENAME=filename1[,filename2,...,filenameN]
If not specified as a command line parameter filename1,...,filenameN are used as input files. If more than a single filename should be used, the list of filenames must be comma separated without any additional white space between the filenames.

TITLE_OUT=ON|OFF
If set to ON a title tag using the input filename as input will be added to the GUIDO output file.
Default is ON.

INSTR_OUT=ON|OFF
If set to ON trackname events of the (MIDI) inputfile will be converted to \instr tags Default is OFF.

TEXT_OUT=ON|OFF
If set to ON text and lyrics events of the (MIDI) inputfile will be converted to \text tags
Default is OFF.

PLAYDURATION=float
Specifies the relation between played note durations and duration in score. Some score notation programs reduce the score duration of notes by a fixed value (*e.g.,* 80% of score duration). To restore the score duration of the MIDI notes this setting can be used.
If, for example, the MIDI input file includes notes with 80% of the original score duration a value of 0.8

for this setting will restore the original score durations of the notes.
Default is 1.0. For processing live performed MIDI files this setting should be set to 1.0.

DURATION_MAP=OFF|filename
If a filename is given a text file containing all note durations in milliseconds will be created. This setting might only be used for debugging and development. Each line of the created file consists of the onset time position (in MIDI ticks), pitch (in semitone steps, c' = 60), duration (in ms), IOI (in ms), and IOI ratio for a single note.
Default is OFF.

MODE=SILENT|INTERACTIVE
If set to INTERACTIVE midi2gmn will prompt for additional user input in ambiguous or hard, complex situations. If set to SILENT it will use default values in these situations which is required for batch processing.
Default is SILENT.

TEMPO_OUT=ON|HIDDEN|OFF
If set to ON or HIDDEN the GUIDO output file will include tempo profile information (inferred or as specified in the MIDI input file) as tempo tags. If set to ON the tempo tags will be of the form \tempo<"[1/4]=bpm","1/4=bpm"> so they will be displayed in the score and evaluated for MIDI playback (by gmn2midi). If set to HIDDEN the tags will be of the form \tempo<"","1/4=bpm"> which will be evaluated for MIDI playback but not displayed in a score.
Default is ON.

SLUR_OUT=ON|OFF
STACC_OUT=ON|OFF
These settings control the detection and output (as \slur and \stacc) of slurs and staccati.
Default is OFF.

DYNAMICS=ON|OFF
If set to ON an intensity profile will be inferred and exported is \intens tags in the output file.
Default is ON.

ORNAMENT=DETECT|OFF
Controls the ornament detection. If set to DETECT ornaments will be inferred and filtered from the input data. If the inferred ornaments will be included to the output file depends on the ORNAMENT_OUT setting.
Default is DETECT.

ORNAMENT_OUT=ON|OFF
If set to ON all inferred ornaments (*e.g.,* grace, trill, turn) will be exported to the output GUIDO file. If set to OFF the inferred ornaments will be filtered by the ornament detection module (see Section 6.4) but not exported to the GUIDO output file.
Default is ON.

ORNAMENT1-N=feature list
These settings specify typical features for specific ornament types used by the k-NN ornament detection module. Theses settings should not be changed by the user.

pitchname_scale=p1,...,p12
For each scale of the circle of fifths (in the range of zero to seven accidentals) the default pitch name for each of the twelve semitone steps (c to b) can be specified. These pitch names will be used by the pitch spell module if no specific context (*e.g.,* chromatic scale) is determined. The pitch names should be specified by valid GUIDO pitch names without duration information separated by single blank symbols. Experienced users might change the settings for special input data.

DETECTTEMPO=OFF|HYBRID|CLICKTRACK
With this setting the two implemented tempo detection approaches can be selected. Each tempo detection strategy requires specific additional settings.
Default is OFF.

CLICKTRACK=n
If DETECTTEMPO=CLICKTRACK, this setting specifies MIDI track n to be scanned for clicknotes. If using standard MIDI files type 0 as input only track 0 can be used as clicktrack. With the settings CLICKCHANNEL

and CLICKFILTER only specific events of the clicktrack can be filtered as clicknotes. For standard MIDI files of type 0 as input files, one of these two settings is required. For type 1 files both are optional.
Default is 1, range is 1 to 255.

CLICKCHANNEL=n
If DETECTTEMPO=CLICKTRACK, this setting selects the events of a specific MIDI channel to be interpreted as click events. If set to 0 events with an arbitrary channel information of the selected clicktrack can be used as clicknotes.
Default is 0, range = 0 to 16.

CLICKFILTER=OFF|PITCHn|CTRLn
If DETECTTEMPO=CLICKTRACK, the value n specifies a specific MIDI pitch or MIDI controller to be interpreted as metronome click information.
Default is OFF, range is 0 to 127.

TACTUSLEVEL=n/d
If DETECTTEMPO=CLICKTRACK, this setting specifies the beat duration of a clicknote (as fraction n/d).
Default is 1/4.

SINGLESTAFF=ON|OFF
If set to ON all voices (as inferred by the voice separation module) will be forced to be written in a single staff by adding a \staff tag to each GUIDO sequence. The setting was introduced for processing guitar score where typically several voices are notated in a single score.
It should be noted that the resulting output will be different from output by forcing the voice separation to use only a single voice (MAXVOICES=1).
Default is OFF.

NOTENUMBERING=ON|OFF
If set to ON the output will contain note numbering information (for each voice/sequence separately) realised by \text tags.
This setting might be useful for debugging or analysis of the output data. Default is OFF.

MERGETRACKS=ON|OFF
As described in Chapter 2 the voice separation module processes each MIDI track or GUIDO sequence separately. This means that if the input data is already separated into several MIDI tracks or GUIDO sequences actually no alternative voice separation can be created. Therefore – if MERGETRACKS is set to ON – all tracks/sequences will be merged before the voice separation module is started.
Default is OFF.

DETECTMETER=OFF|MIDIFILE|DETECT
DETECTKEY=OFF|MIDIFILE|DETECT
These two settings control the key- and time signature detection. If set to MIDIFILE the information as indicated in the input file will be exported to the GUIDO output. If set to DETECT the key- and time signature will be inferred automatically by the correspondent modules. If set to OFF the output file will not contain any \key or \meter tags, the key- and time signature of the input file will be ignored.
It should be noted that the score layout algorithm of the current NoteViewer/NoteServer implementation needs time signature information for inferring system breaks.
Default is MIDIFILE.

QPATTERN=filename|OFF
TPATTERN=filename|OFF
Specifies the filename (including filepath) of the pattern database used for quantisation (QPATTERN) and for hybrid tempo detection (TPATTERN). If set to OFF no pattern will be used for quantisation respectively tempo detection.
Each pattern database can be an arbitrary GUIDO file where each sequence is used as a single pattern. If the specified file does not exist a default pattern database (with a small set of patterns) will be created at the specified location. The user might want to extend the created database by adding additional sequences. The pitch information of these GUIDO files will be ignored. If the filename includes no path information the local working directory will be used. If a performance is recognised as a mechanical performance, a given pattern database will be ignored during quantisation!
Default is qpatternbase.gmn and tpatternbase.gmn.

COLOURVOICESLICES=ON|OFF
If set to ON the slices used for the voice separation will be marked, where all notes of a single slices will be coloured with a specific colour. Successive slices will have different colours. This setting should only be used if no other colour setting is used.
Default is OFF.

MARKQPATTERN=ON|OFF
If set to ON every start note of a matched quantisation pattern will be marked with a green notehead, all other notes quantised by a pattern will have black noteheads. Notes in regions where no quantisation pattern could be applied will appear with read noteheads. This setting should only be used if no other colour setting is used.
Default is OFF.

SIMILARITY=ON|OFF
If set to ON a self-similarity analysis for each voice will be performed. The corresponding similarity matrices and best alignments will be written to an ASCII file. Please see Chapter 3 for more details on the self-similarity module. The self-similarity module requires additional settings.
Default is OFF.

SIM_STEPSIZE=s
SIM_WINDOWSIZE=w
These two settings specify the parameters for window-size and step-size of the self-similarity analysis module (MusicBLAST). The output of this module is written to a log-file.
Default is s = 1, w = 4.

CTRACKSELFSIM=ON|OFF
If set to ON a self-similarity analysis for an inferred clicktrack is performed, which requires that also a tempo profile has been inferred by one of the tempo detection modules. The analysis data including a similarity matrix and alignment data are written to a log file and a GUIDO file for the detected patterns.
Default is OFF.

MIR=filename|OFF
The file which should be used as query for the prototypical implementation of the MIR approach can be specified by filename (including filepath information). The query file must be a valid GUIDO file where the first sequence will be used a query.
The output of the MIR module (including similarity matrices and alignments) will be written to a log file.
Default is OFF.

AttackGridFName=filename
DurationGridFName=filename
IOIGridFName=filename
These settings specify the filenames of GUIDO files containing the information about valid score grid positions (for onset times and durations) used during tempo detection and quantisation (see Section 4.3.3 and Section 5.2.4). Each voice in such a files should contain only a single note. All integer multiples of this note durations will become a valid position in the corresponding grid.
Default is IOIList.ini for all three settings.

IOIratioGridFName=filename
Specifies the filename with the list of known IOI ratios used for the hybrid tempo detection (see Section 4.3.3). See Section A.4 for a description of the file format.
Default is IOIratioList.ini.

MAXVOICES=n
If n> 0, the maximum number of voices which will be created by the voice separation module for each track/sequence of the input file is limited to n. If n $\leq$ 0, no limitation is used.
Default is -1.

EMPTYVOICEIV=n
Equivalent interval for pitch penalty for first note of a voice. The interval n must be specified in semitone steps.
Default is 11, range is $(0 : \infty)$

`PITCHLOOKBACK=n`
`SPLITVOICEDECAY=f`
These two settings control the calculation of the average pitch used as interval for the pitch penalty of the voice separation. `n` gives the number of notes which should be used for average calculation and `f` gives the decay for each step of the average calculation. If $\texttt{n} \leq 1$ no average is calculated. Usually only the setting `PITCHLOOKBACK` should be changed by the user depending on special types of input data (see Equation 2.2.3 for more details).
Default is $\texttt{n} = 1$, $\texttt{f} = 0.8$, range is $\texttt{f} = (0 : 1)$.

`LSEARCHDEPTH=n`
`RWALKTRESH=f`
These two settings control the random walk optimisation behaviour of the voice separation. A large `LSEARCHDEPTH` increases the probability that the optimum solution for a slice is found. `RWALKTRASH` control the number of random walks. If $\texttt{f} = 1$ no random walks will be performed and with $\texttt{f} = 0$ only random walks will be performed (see also Section 2.2.4). Usually this settings should not be changed. With a large `RWALKTRASH` will increase with a very low `RWALKTRASH` not always a good voice separation can be found.
Default is $\texttt{n} = 15$ and $\texttt{f} = 0.8$, ranges: $\texttt{f} = (0,1)$, $\texttt{n} = [10,30]$ (preferred).

`POVERLAP=f`
`PPITCH=f`
`PGAP=f`
`PCHORD=f`
These four settings specify the weight of the corresponding four penalty functions pitch-penalty, gap-penalty, chord-penalty, overlap penalty of the voice separation module (see Equation 2.12).
By changing the relations between the penalty parameters the behaviour of the voice separation module can be changed by the user.
Default is 0.5, range is $[0, \infty)$.

`TIMESIGINTEGRSIZE=n/d`
Specifies the integration size (as fraction `n/d`) for the autocorrelation used for inferring the time signature (see Section 6.1.2).
Default is 8/1.

`MATCHWINDOW=n/d`
Defines the ϵ-window size (as score duration fraction `n/d`) for the autocorrelation on unquantised input data during inferring the time signature (see Equation 6.6).
Default is 1/24.

`LEGATO_TIME=t`
Specifies the maximum time (in ms) of an overlap between two successive notes that can be removed by the pre processing module. Usually this settings needs not to be changed by the user (see also Section 1.4.3).
Default is 152.

`EQUAL_TIME=t`
Specifies the maximum distance (in ms) between two successive notes whose onset times could be merged to an average time position by the pre-processing module. Usually this setting needs not to be changed by the user (see also Section 1.4.3).
Default is 60.

A.4 Storing of Binclass Lists

For storing binclass lists (see Section 4.3.3 and Section 5.2.4) used for tempo detection and quantisation a list file format (also used for the parameters of `midi2gmn`) was chosen. Each entry of the class list is of the form: *prefix*ID=$\{\ p\_1\,[, p\_i]^*\ \}$ where for a single file ID$= 1, 2, \ldots, n$. To keep the parsing simple each file also must include an entry COUNT=n to specify the number of entries (*i.e.,* number of classes).

For storing the IOI ratio class list, *prefix* is set to `IOIr`, parameter p_1 denotes the normalised IOI ratio (see Equation A.18), and p_2 the bias (float) of the corresponding IOI ratio class. During parsing the

bias parameters become normalised so that $\sum\_i = 1^n \text{bias}\_i = 1$. The bias will be updated during tempo detection and quantisation and the new distribution of IOI ratios and durations will be stored after processing a file.

Because the IOI duration class lists represent rhythmical score durations, it was more adequate to store them as GUIDO files which can be edited via ASCII editors and also viewed by the GUIDO NoteViewer. Each class is represented as a single note sequence where the weight of each class is stored by a special \statist tag. This tag can easily be expanded by additional parameters to hold more information about the corresponding class (*e.g.,* mean, variance, number of entries).

A.5 File Format for Patterns

The pattern database used for quantisation and tempo detection should stored in human readable way, easy to read and edit, including score information and statistical information, and it should provide the possibility of a graphical score display of the patterns. Possible file formats would be MIDI including the statistical information as meta text events, proprietor table-based text formats, or GUIDO including the statistical information as special GUIDO tags. Because GUIDO fulfils all these requirements it was chosen as file format for the pattern databases used by the current implementation of our system. This decision also gives the possibility to use arbitrary excerpts of pieces already available in GUIDO format as pattern database without any conversion.
A complete pattern database is represented as a GUIDO segment where each sequence of this segment represents a single pattern. For the statistical information two special tags were defined:

- **\statist** – statistical information for the complete sequence (*i.e.,* a single pattern).
  Parameters:
  cUsed (int) – the total number of usage
  cUsedCur (int) – the number of usage during the last call.
  Range: none – the tag is valid for the complete pattern.

- **\nStat** – statistical information for single notes.
  Parameters:
  mean (float) – the mean of all performed IOIs inferred as the note in the tag range.
  sigma (float) – the variance of all performed IOIs inferred as the note in the tag range.
  Range: mandatory – only a single note should be inside the range.

If the statistical tags are missing they will be calculated by `midi2gmn` and added to the database.

Other GUIDO applications, such as the NoteViewer will gracefully ignore these non-standard tags and display/evaluate only the basic note information and the standard tags in a score. The pattern database can include arbitrary additional GUIDO tags which will be ignored by `midi2gmn`. Also any melodic information (*i.e.,* pitch class, octave, accidentals) is ignored by `midi2gmn` during pattern matching.

A.6 Evaluation of The Son-Clave Performances

Figure A.2 shows the actual performed tempo profile (calculated at each onset time position) of the three live recordings of the son-clave pattern discussed in Section 4.4 and Section 5.4.[2]

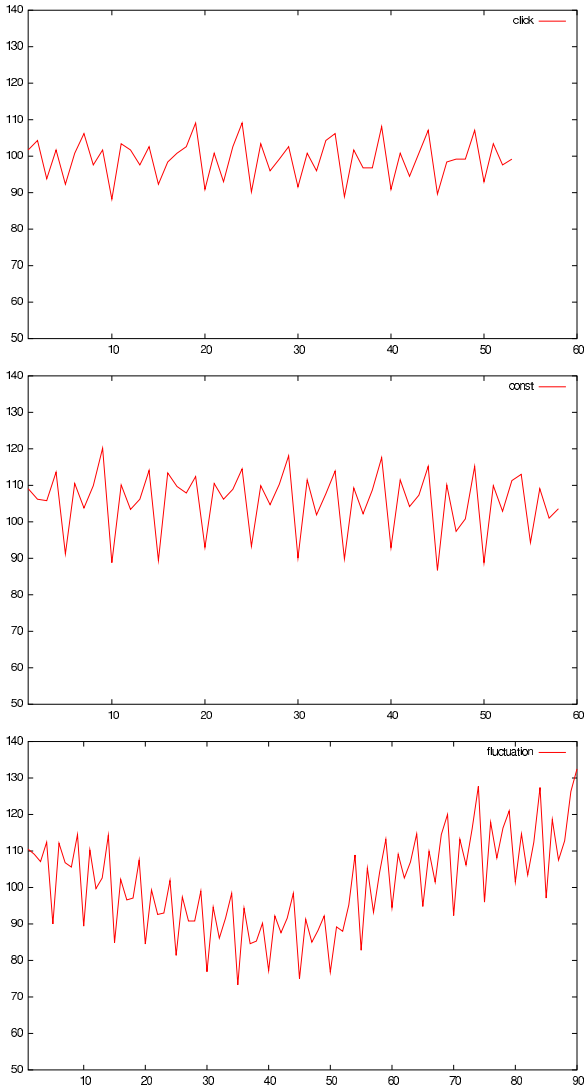


Figure A.2: Resulting tempo profiles for hybrid tempo detection applied to son-clave performance files. Performance 1 (top) was played in synch to a metronome click; performance 2 (centre) was played without metronome information, but with the intention to keep a constant tempo; performance 3 (bottom) was played with the intention to slow down and accelerate again.

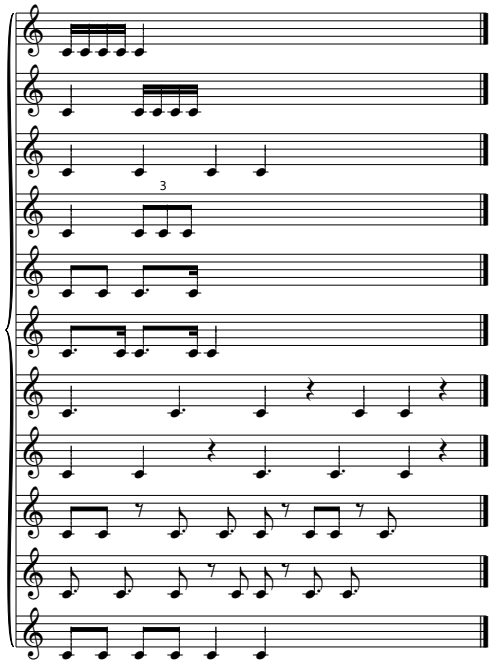


Figure A.3: Pattern database used for tempo detection of son-clave files, each voice represents a single pattern. The son-clave pattern is included in 2-3 and 3-2 version, both in an alla-breve and a 4/4-time signature version (resulting in half durations).

[2]The files were performed by the author.

The two performances played without a metronome click show significant repeating pattern of tempo deviations caused by deviations of the onset times from their mechanical positions.

For the evaluation of the tempo detection module (see Section 4.4) the tempo pattern database shown in Figure A.3 has been used.

The used IOI ratio list including the weight for each class after processing the son-clave files:

$$
\begin{aligned}
&\text{IOIr23}=\{8, 0.0189476\}\\
&\text{IOIr22}=\{7, 0.0189476\}\\
&\text{IOIr21}=\{6, 0.0189476\}\\
&\text{IOIr20}=\{5, 0.0189476\}\\
&\text{IOIr19}=\{4, 0.0189476\}\\
&\text{IOIr18}=\{3, 0.0189476\}\\
&\text{IOIr17}=\{2, 0.0189476\}\\
&\text{IOIr16}=\{1.5, 0.0189476\}\\
&\text{IOIr15}=\{1, 0.199168\}\\
&\text{IOIr14}=\{0.5, 0.0188889\}\\
&\text{IOIr13}=\{0.3333, 0.0697128\}\\
&\text{IOIr12}=\{0, 0.123603\}\\
&\text{IOIr11}=\{-0.333, 0.0712626\}\\
&\text{IOIr10}=\{-0.5, 0.0216875\}\\
&\text{IOIr9}=\{-1, 0.192265\}\\
&\text{IOIr8}=\{-1.5, 0.0189476\}\\
&\text{IOIr7}=\{-2, 0.0189476\}\\
&\text{IOIr6}=\{-3, 0.0189476\}\\
&\text{IOIr5}=\{-4, 0.0189476\}\\
&\text{IOIr4}=\{-5, 0.0191987\}\\
&\text{IOIr3}=\{-6, 0.0189476\}\\
&\text{IOIr2}=\{-7, 0.0189476\}\\
&\text{IOIr1}=\{-8, 0.0189476\}
\end{aligned}
$$

IOI list used for tempo detection of son-clave files:

```
{ [\statist<w=0.306810>( c0*1/16 ) ],
[\statist<w=0.139468>( c0*1/8 ) ],
[\statist<w=0.105195>( c0*3/16 ) ],
[\statist<w=0.160687>( c0*1/4 ) ],
[\statist<w=0.145820>( c0*3/8 ) ],
[\statist<w=0.142020>( c0*1/2 ) ] }
```

The IOI list and IOI ratio list have been used only for the few regions, where no pattern had matched.

A.7 Patterns Used For Quantisation

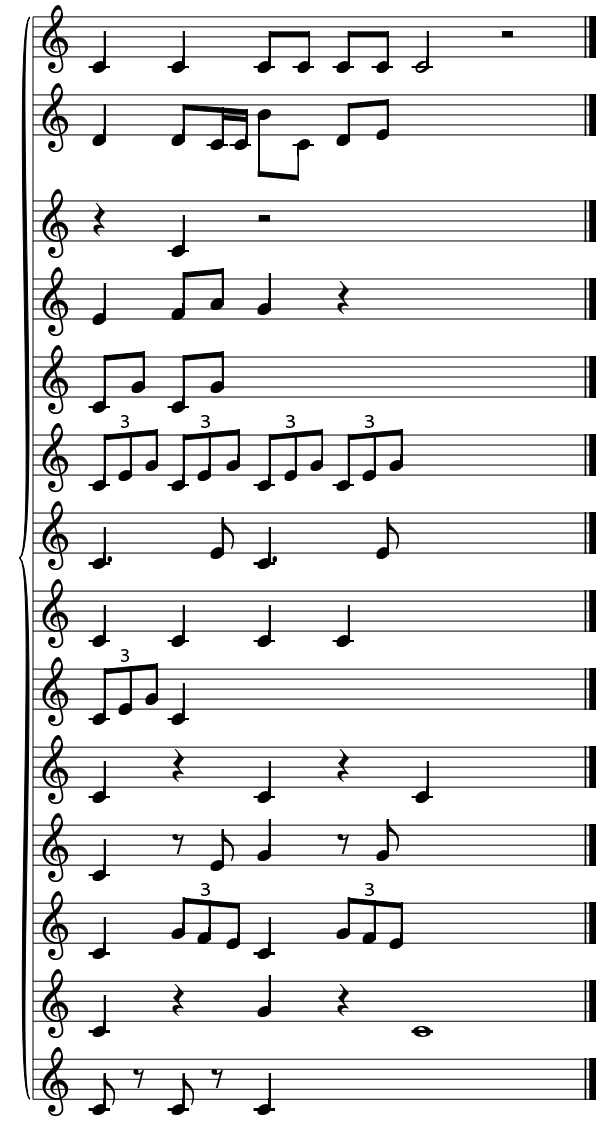


Figure A.4: Pattern database for quantisation of Beethoven *Sonata Nr. 20, Op. 49, 2.*

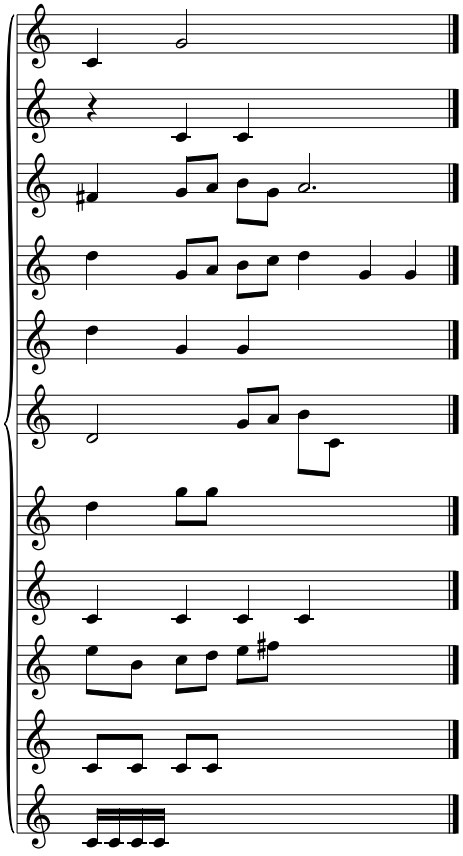


Figure A.5: Pattern database for quantisation of Bach *Minuet in G.*

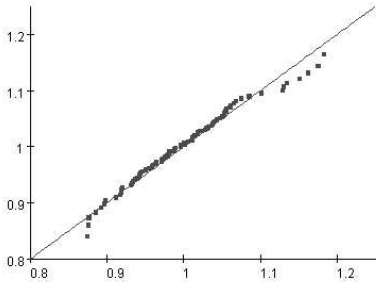
A.8 Statistical Analysis of Performance Data

For testing our assumptions on the typical expected errors in live performed input data we recorded some test files and analysed the distribution of the deviations between intended and performed note durations. The test files were performed by an advanced player (the author) and a non-musician on an electronic keyboard (Roland A-70) and recorded with a software sequencer. For one test (see Figure A.6 and Figure A.7), the two players where asked to play quarter notes on a single key in synch with the metronome clicks of the sequencer at a tempo of 100 bpm.

For the second test, the players where asked to play several repetitions of an up and down scale consisting of five notes (c, d, e, f, g) with one hand and with a constant tempo, but without any metronome clicks. Because the non-musician was not able to perform this task, we analysed only the data set of the advanced musicion (see Figure A.8).

Figure A.6: qq-plot for distribution of the deviations (in units of seconds) between the onset times of mechanical and performed quarter note beats played by an advanced musician with one finger in synch with a metronome click (100bpm); a) left-hand and b) right-hand.

Figure A.7: qq-plot for the distribution of the deviations (in units of seconds) between onset times of mechanical and performed quarter note beats played by a non-musician with one finger in synch with a metronome click (100bpm) a) left-hand and b) right-hand.

a) 

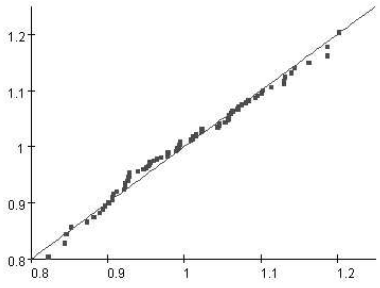
b) 

Figure A.8: qq-plot of the distribution of IOI ratios (1 denotes equal length) of successive quarter notes played by an advanced musician with five fingers (five note up and down scale) without a metronome click and the intention to play with a constant tempo; a) left-hand and b) right-hand.

A.9 Gaussian Window Function

For many distance measures described in this thesis a Gaussian window function is used. A Gaussian window function $W_{Gauss}(x, \sigma)$ can be defined as

$$W_{Gauss}(x, \sigma) = e^{-\frac{1}{2}\cdot(\frac{x}{\sigma})^2}, x \in \mathbb{R}, \sigma \in \mathbb{R}^+ \tag{A.19}$$

or as

$$W_{Gauss}(x_1, x_2, \sigma) = e^{-\frac{1}{2}\cdot(\frac{x_1-x_2}{\sigma})^2}, x_1, x_2 \in \mathbb{R}, \sigma \in \mathbb{R}^+. \tag{A.20}$$

The shape of this function is shown in Figure A.9. The width of the window can be adjusted with the parameter σ where for all $\sigma > 0$: $W_{Gauss}(\sigma, \sigma) = 1/\sqrt{e}$ and $W_{Gauss}(x, x \pm \sigma, \sigma) = 1/\sqrt{e}$.

The first derivation $W'_{Gauss}(x, \sigma) = -\frac{x}{\sigma^2} e^{-0.5\cdot(\frac{x}{\sigma})^2}$ and the second derivation $W''_{Gauss}(x, \sigma) = \frac{1}{\sigma^2}(\frac{x^2}{\sigma^2} - 1)e^{-0.5\cdot(\frac{x}{\sigma})^2}$.

With $W''_{Gauss}(\pm\sigma, \sigma) = 0$ follows that the turning points of $W_{Gauss}(x, \sigma)$ are exactly at $x = \pm\sigma$.

If a penalty p_d for a distance between two values x_1, x_2 should be calculated we use

$$p_d(x_1, x_2, \sigma) = 1 - W_{Gauss}(x_1, x_2, \sigma). \tag{A.21}$$

If the penalty p_r for a relation of two values $x_1, x_2 \neq 0$ should be calculated we use

$$p_r(x_1, x_2, \sigma) = 1 - W_{Gauss}(\log(\frac{x_1}{x_2}), \sigma), \tag{A.22}$$

which results in $W_{Gauss}(\log(\frac{x_1}{x_2}), \sigma) = W_{Gauss}(\log(\frac{x_2}{x_1}), \sigma)$.

The exponential shape of W results in the intended feature that the penalty sum of several small distance values will be smaller than the sum of a few large distance penalties. Also the range of a penalty value will be normalised to the interval $(0, 1]$ for any distance in $(-\infty, +\infty)$.

If for special purpose a higher separation of input values is needed (*e.g.,* for the chord penalty) the shape of the window can be controlled by using

$$W_{k\text{-}Gauss}(x, \sigma, k) = e^{-\frac{1}{2}\cdot(\frac{x}{\sigma})^{2k}}, \quad \text{with } k \in \mathbb{N}. \tag{A.23}$$

The effect of varying parameter k is shown in Figure A.10.

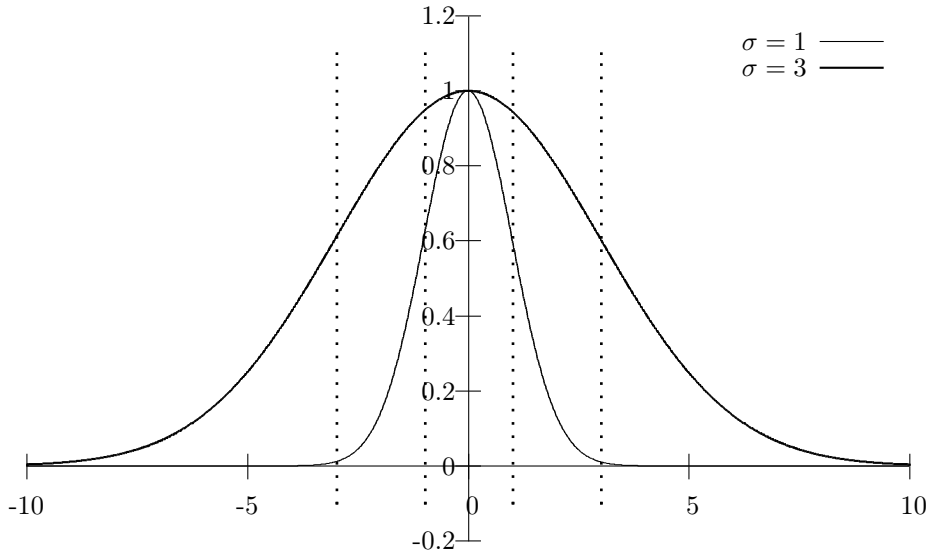


Figure A.9: Shape of a Gaussian window functions for $\sigma = 1$ and $\sigma = 3$. The vertical dotted lines indicate the turning points of the curves at $x = \pm\sigma$.

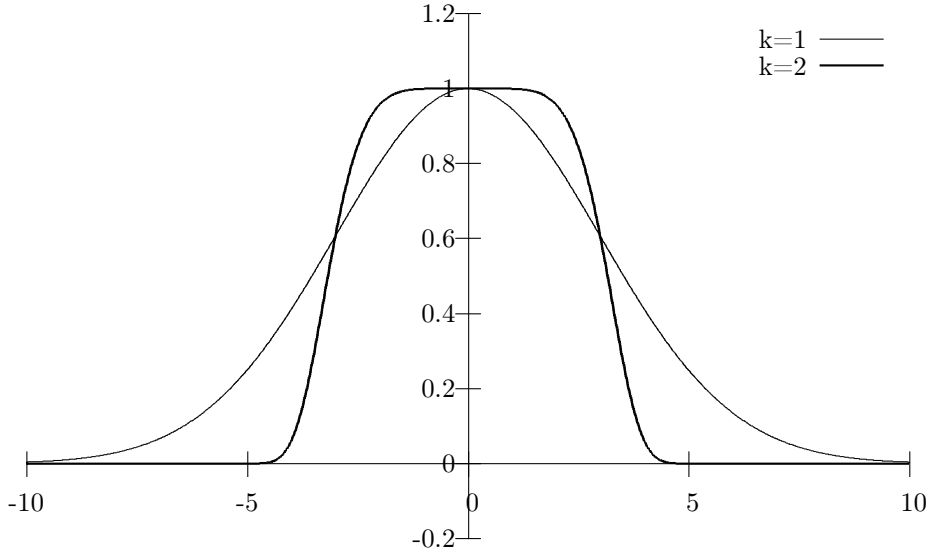


Figure A.10: Shape of a k-Gaussian window functions $W_{k\text{-}Gauss}(x, \sigma, k)$ for $k = 1$ and $k = 2$ ($\sigma = 3$). The intersection points between the functions are at $[\pm\sigma, e^{-0.5\sigma}], \forall\ k \in \mathbb{N}$.

A.10 Low-Level GUIDO Specification v1.0

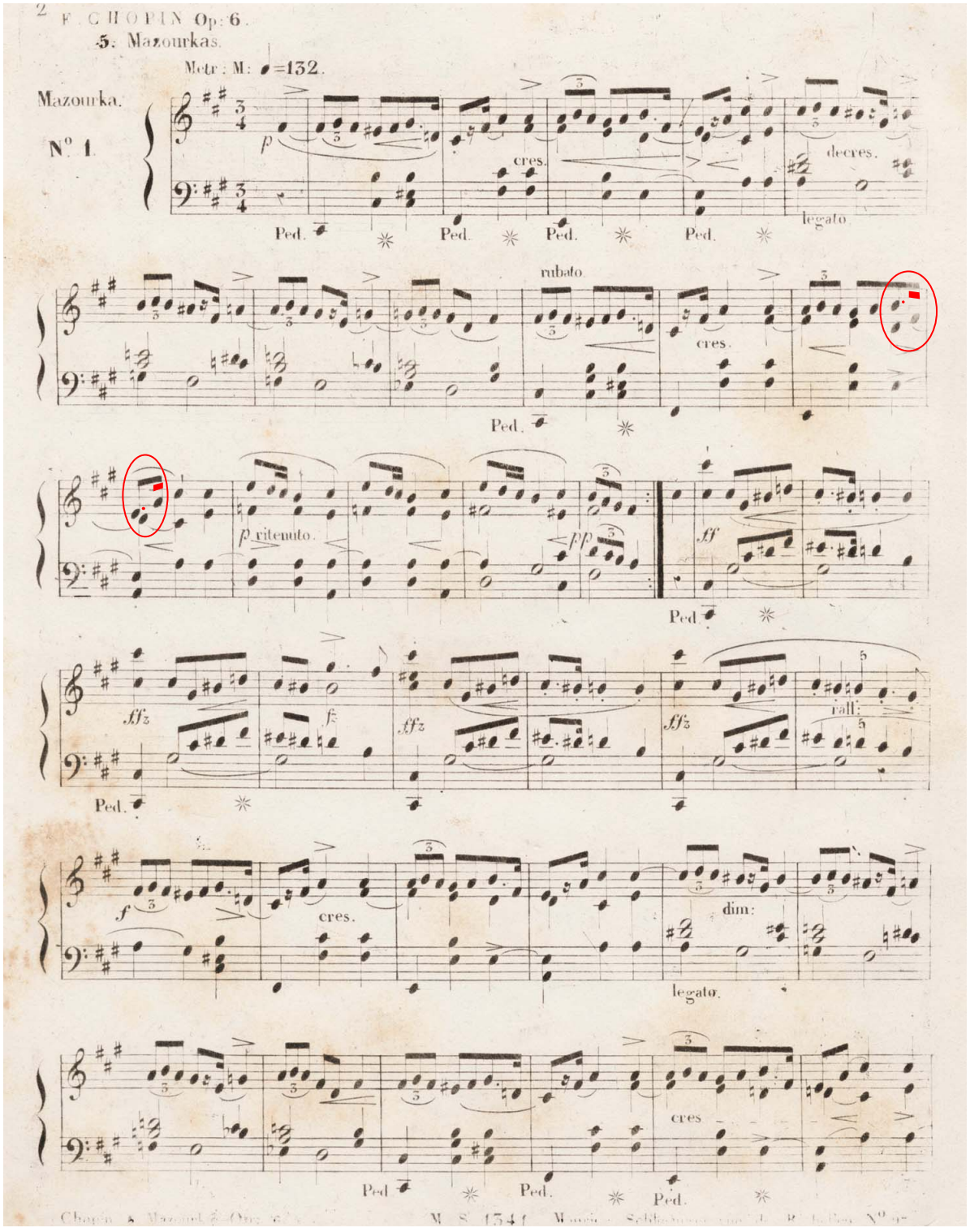
For representing MIDI type score representations where notes are split into note-on and note-off events we introduce here version 1.0 of the Low-Level GUIDO specification. The specification also includes special tags respectively additional parameters for standard tags for representing MIDI specific event types in GUIDO syntax. Please see Section 1.4.1 and `http://www.salieri.org/GUIDO` for more information on GUIDO Music Notation.

tag name	**parameters**	**description**
\staff	<id[,channel][,port]>	select port and channel for a staff id= int/string channel = int [1,16] (opt) port = "MIDI x" | "name" (opt)
\instr	<name|[,type,bank]>	instrument tag with MIDI related parameters type = "MIDI x" | "GM id" x, id = int [0,127] patch number (opt) bank = int [0,16129] | "aa,bb" (opt)
\instr	<name|[,type,bank],key>	instrument tag for percussion sequences key = int [0,127] MIDI pitch for percussion instruments pitch class, accidentals, and octave will be replaced by key during playback
\bankSelect	<bank | bankM, bankL>	bank select for sequence bank = int [0,16129] bankL, bankM = int [0,127]
\noteOn[:id]	<keyno|pitch[,vel]>	note-on event keyno = MIDI key number [0,127] pitch = pitch in GUIDO syntax vel = int [0,127] MIDI intensity id = int id of corresponding noteOff tag
\noteOff[:id]	<keyno|pitch[,vel]>	note-off event keyno = MIDI key number [0,127] pitch = pitch in GUIDO syntax vel = int [0,127] MIDI intensity id = int id of corresponding noteOn tag
\polyAt	<keyno|pitch,vel>	polyphonic after touch keyno = int [0,127] pitch as MIDI key number pitch = pitch in GUIDO syntax vel = int 0,127]
\channelAt	<val>	channel after touch val = int [0,127] after touch strength
\controller	<no,value>	value for a MIDI controller no = int [0,127] value = int [0,127] after touch strength
\RPN	<val | valM, valL>	registered parameter val = int [0,16129] valM,valL =int [0,127]
\dataEntry	<val | valM, valL>	data entry event val = int [0,16129] valM,valL = int [0,127]

\pitchBend	<val|valM, valL>	PitchBend(2 byte) val = int [-8063,+8063] valM,valL = int [0,127]
\channelMode	<val>	channel mode select val= string "all sounds off"|"reset all controllers", ... (case insensitive)
\sysEx	<data>	system exclusive message data="[id]aa,bb,cc,dd,..." xx= int [0,255] | [$00, $ff] | [00H, ffH] id = string “ROLAND”|‘’YAMAHA”|“GENERIC”|... depending on id, a checksum might be calculated and added automatically

A.11 Chopin, *Op. 6, Mazurka 1*, measures 1–36, score

The last two quavers in bar 11 and the first two quaver notes in bar 12 have been played by the performer as a dotted quaver followed by a semi-quaver (see red remarks in the score).



A.12 Chopin, *Op. 6, Mazurka 1*, measures 1–36, merged click-track data (unfiltered)

tempo of MIDI recorder =120bpm
ppq of MIDI file = 480 ticks/quarter

measure	pos [ticks]	#	perfIOI	tempo [bpm]	pIOIratio	sIOIratio	sIOI [beat]	sIOIrDur	scorePos [beat]
	0	1	740	77,84			0,250	0,250	-0,250
1	740	1	268	71,64	-2,76120	-3,00	0,083	0,091	0,000
	1008	1	164	117,07	-1,63410	1,00	0,083	0,051	0,083
	1172	1	173	110,98	1,05490	1,00	0,083	0,088	0,167
	1345	3	217	132,72	1,25430	1,50	0,125	0,105	0,250
	1562	1	247	116,6	1,13820	1,00	0,125	0,142	0,375
	1809	4	283	152,65	1,14570	1,50	0,188	0,143	0,500
	2092	1	147	97,96	-1,92520	-3,00	0,063	0,097	0,688
2	2239	2	295	146,44	2,00680	3,00	0,188	0,125	0,750
	2534	1	156	92,31	-1,89100	-3,00	0,063	0,099	0,938
	2690	3	603	95,52	3,86540	4,00	0,250	0,242	1,000
	3293	4	587	98,13	-1,02730	1,00	0,250	0,243	1,250
3	3880	1	172	111,63	-3,41280	-3,00	0,083	0,073	1,500
	4052	1	116	165,52	-1,48280	1,00	0,083	0,056	1,583
	4168	1	158	121,52	1,36210	1,00	0,083	0,114	1,667
	4326	4	216	133,33	1,36710	1,50	0,125	0,114	1,750
	4542	1	236	122,03	1,09260	1,00	0,125	0,137	1,875
	4778	3	312	138,46	1,32200	1,50	0,188	0,165	2,000
	5090	1	111	129,73	-2,81080	-3,00	0,063	0,067	2,188
4	5201	2	309	139,81	2,78380	3,00	0,188	0,174	2,250
	5510	1	117	123,08	-2,64100	-3,00	0,063	0,071	2,438
	5627	3	559	103,04	4,77780	4,00	0,250	0,299	2,500
	6186	3	579	99,48	1,03580	1,00	0,250	0,259	2,750
5	6765	3	183	104,92	-3,16390	-3,00	0,083	0,079	3,000
	6948	1	142	135,21	-1,28870	1,00	0,083	0,065	3,083
	7090	1	143	134,27	1,00700	1,00	0,083	0,084	3,167
	7233	2	335	128,96	2,34270	2,25	0,188	0,195	3,250
	7568	1	135	106,67	-2,48150	-3,00	0,063	0,076	3,438
	7703	3	562	102,49	4,16300	4,00	0,250	0,260	3,500
6	8265	3	181	106,08	-3,10500	-3,00	0,083	0,081	3,750
	8446	1	100	192	-1,81000	1,00	0,083	0,046	3,833

measure	measure number of score
pos [ticks]	onset time position in MIDI ticks
perfIOI	performed IOI in MIDI ticks
pIOIratio	performed IOI ratio (pseudo-semi-log IOI ratio) as defined in Equation A.16
tempo	local tempo during note i, see Equation 1.1 and Equation 1.6.
sIOI	IOI in score time units
sIOIrDur	IOI in score time units calculated from by sIOI of previous note and performed IOI ratio of current note $\mathrm{s}IOIrDur_i = \begin{cases} \mathrm{sIOI}_{i-1} \cdot \mathrm{pIOIratio}_i, & \text{if } \mathrm{perfIOI}_i > 0 \\ -\frac{\mathrm{sIOI}_{i-1}}{\mathrm{pIOIratio}_i}, & \text{otherwise} \end{cases}$

	8546	2	150	128	1,50000	1,00	0,083	0,125	3,917
	8696	1	314	137,58	2,09330	2,25	0,188	0,174	4,000
	9010	3	134	107,46	-2,34330	-3,00	0,063	0,080	4,188
	9144	3	446	129,15	3,32840	4,00	0,250	0,208	4,250
7	9590	1	217	88,48	-2,05530	-3,00	0,083	0,122	4,500
	9807	1	99	193,94	-2,19190	1,00	0,083	0,038	4,583
	9906	2	140	137,14	1,41410	1,00	0,083	0,118	4,667
	10046	1	336	128,57	2,40000	2,25	0,188	0,200	4,750
	10382	3	131	109,92	-2,56490	-3,00	0,063	0,073	4,938
	10513	3	555	103,78	4,23660	4,00	0,250	0,265	5,000
8	11068	1	226	84,96	-2,45580	-3,00	0,083	0,102	5,250
	11294	1	156	123,08	-1,44870	1,00	0,083	0,058	5,333
	11450	2	189	101,59	1,21150	1,00	0,083	0,101	5,417
	11639	1	325	88,62	1,71960	1,50	0,125	0,143	5,500
	11964	3	440	65,45	1,35380	1,00	0,125	0,169	5,625
	12404	2	855	67,37	1,94320	2,00	0,250	0,243	5,750
9	13259	1	277	69,31	-3,08660	-3,00	0,083	0,081	6,000
	13536	1	176	109,09	-1,57390	1,00	0,083	0,053	6,083
	13712	4	205	93,66	1,16480	1,00	0,083	0,097	6,167
	13917	1	225	128	1,09760	1,50	0,125	0,091	6,250
	14142	4	256	112,5	1,13780	1,00	0,125	0,142	6,375
	14398	1	336	128,57	1,31250	1,50	0,188	0,164	6,500
	14734	2	123	117,07	-2,73170	-3,00	0,063	0,069	6,688
10	14857	1	331	130,51	2,69110	3,00	0,188	0,168	6,750
	15188	3	153	94,12	-2,16340	-3,00	0,063	0,087	6,938
	15341	4	611	94,27	3,99350	4,00	0,250	0,250	7,000
	15952	1	562	102,49	-1,08720	1,00	0,250	0,230	7,250
11	16514	1	166	115,66	-3,38550	-3,00	0,083	0,074	7,500
	16680	1	118	162,71	-1,40680	1,00	0,083	0,059	7,583
	16798	4	137	140,15	1,16100	1,00	0,083	0,097	7,667
	16935	1	219	131,51	1,59850	1,50	0,125	0,133	7,750
	17154	3	219	131,51	1,00000	1,00	0,125	0,125	7,875
	17373	1	311	138,91	1,42010	1,50	0,188	0,178	8,000
	17684	2	111	129,73	-2,80180	-3,00	0,063	0,067	8,188
12	17795	1	321	134,58	2,89190	3,00	0,188	0,181	8,250
	18116	3	119	121,01	-2,69750	-3,00	0,063	0,070	8,438
	18235	3	547	105,3	4,59660	4,00	0,250	0,287	8,500
	18782	3	595	96,81	1,08780	1,00	0,250	0,272	8,750
13	19377	1	313	92,01	-1,90100	-2,00	0,125	0,132	9,000
	19690	1	114	126,32	-2,74560	-2,00	0,063	0,046	9,125
	19804	4	136	105,88	1,19300	1,00	0,063	0,075	9,188
	19940	4	457	126,04	3,36030	4,00	0,250	0,210	9,250
	20397	3	449	128,29	-1,01780	1,00	0,250	0,246	9,500
14	20846	1	294	97,96	-1,52720	-2,00	0,125	0,164	9,750
	21140	1	104	138,46	-2,82690	-2,00	0,063	0,044	9,875
	21244	4	133	108,27	1,27880	1,00	0,063	0,080	9,938
	21377	4	487	118,28	3,66170	4,00	0,250	0,229	10,000
	21864	4	570	101,05	1,17040	1,00	0,250	0,293	10,250
15	22434	1	312	92,31	-1,82690	-2,00	0,125	0,137	10,500
	22746	1	132	109,09	-2,36360	-2,00	0,063	0,053	10,625
	22878	2	182	79,12	1,37880	1,00	0,063	0,086	10,688
	23060	5	528	109,09	2,90110	4,00	0,250	0,181	10,750
	23588	4	717	80,33	1,35800	1,00	0,250	0,340	11,000
16	24305	2	214	89,72	-3,35050	-3,00	0,083	0,075	11,250
	24519	2	188	102,13	-1,13830	1,00	0,083	0,073	11,333
	24707	2	268	71,64	1,42550	1,00	0,083	0,119	11,417
	24975	1	857	67,21	3,19780	3,00	0,250	0,266	11,500
	25832	4	513	112,28	-1,67060	1,00	0,250	0,150	11,750
17	26345	2	454	126,87	-1,13000	1,00	0,250	0,221	12,000
	26799	2	218	132,11	-2,08260	-2,00	0,125	0,120	12,250
	27017	2	201	143,28	-1,08460	1,00	0,125	0,115	12,375
	27218	2	200	144	-1,00500	1,00	0,125	0,124	12,500
	27418	2	190	151,58	-1,05260	1,00	0,125	0,119	12,625
18	27608	2	301	143,52	1,58420	1,50	0,188	0,198	12,750
	27909	2	94	153,19	-3,20210	-3,00	0,063	0,059	12,938
	28003	2	412	139,81	4,38300	4,00	0,250	0,274	13,000
	28415	4	484	119,01	1,17480	1,00	0,250	0,294	13,250
19	28899	2	516	111,63	1,06610	1,00	0,250	0,267	13,500

	29415	2	242	119,01	-2,13220	-2,00	0,125	0,117	13,750
	29657	2	210	137,14	-1,15240	1,00	0,125	0,108	13,875
	29867	2	186	154,84	-1,12900	1,00	0,125	0,111	14,000
	30053	2	194	148,45	1,04300	1,00	0,125	0,130	14,125
20	30247	2	174	165,52	-1,11490	1,00	0,125	0,112	14,250
	30421	3	184	156,52	1,05750	1,00	0,125	0,132	14,375
	30605	1	411	140,15	2,23370	2,00	0,250	0,279	14,500
	31016	1	184	156,52	-2,23370	-2,00	0,125	0,112	14,750
	31200	5	227	126,87	1,23370	1,00	0,125	0,154	14,875
21	31427	2	426	135,21	1,87670	2,00	0,250	0,235	15,000
	31853	2	199	144,72	-2,14070	-2,00	0,125	0,117	15,250
	32052	2	190	151,58	-1,04740	1,00	0,125	0,119	15,375
	32242	2	171	168,42	-1,11110	1,00	0,125	0,113	15,500
	32413	2	188	153,19	1,09940	1,00	0,125	0,137	15,625
22	32601	2	315	137,14	1,67550	1,50	0,188	0,209	15,750
	32916	2	93	154,84	-3,38710	-3,00	0,063	0,055	15,938
	33009	2	453	127,15	4,87100	4,00	0,250	0,304	16,000
	33462	4	407	141,52	-1,11300	1,00	0,250	0,225	16,250
23	33869	2	390	147,69	-1,04360	1,00	0,250	0,240	16,500
	34259	2	212	135,85	-1,83960	-2,00	0,125	0,136	16,750
	34471	2	213	135,21	1,00470	1,00	0,125	0,126	16,875
	34684	2	220	130,91	1,03290	1,00	0,125	0,129	17,000
	34904	2	329	87,54	1,49550	1,00	0,125	0,187	17,125
24	35233	2	417	82,88	1,26750	1,20	0,150	0,158	17,250
	35650	2	356	97,08	-1,17130	1,00	0,150	0,128	17,400
	36006	2	490	70,53	1,37640	1,00	0,150	0,206	17,550
	36496	1	750	46,08	1,53060	1,00	0,150	0,230	17,700
	37246	1	314	55,03	-2,38850	-2,00	0,075	0,063	17,850
	37560	2	368	46,96	1,17200	1,00	0,075	0,088	17,925
25	37928	1	228	84,21	-1,61400	1,11	0,083	0,046	18,000
	38156	1	142	135,21	-1,60560	1,00	0,083	0,052	18,083
	38298	2	165	116,36	1,16200	1,00	0,083	0,097	18,167
	38463	1	223	129,15	1,35150	1,50	0,125	0,113	18,250
	38686	4	283	101,77	1,26910	1,00	0,125	0,159	18,375
	38969	1	459	94,12	1,62190	1,50	0,188	0,203	18,500
	39428	1	210	68,57	-2,18570	-3,00	0,063	0,086	18,688
26	39638	1	378	114,29	1,80000	3,00	0,188	0,113	18,750
	40016	1	152	94,74	-2,48680	-3,00	0,063	0,075	18,938
	40168	3	673	85,59	4,42760	4,00	0,250	0,277	19,000
	40841	4	513	112,28	-1,31190	1,00	0,250	0,191	19,250
27	41354	1	148	129,73	-3,46620	-3,00	0,083	0,072	19,500
	41502	1	96	200	-1,54170	1,00	0,083	0,054	19,583
	41598	1	175	109,71	1,82290	1,00	0,083	0,152	19,667
	41773	4	223	129,15	1,27430	1,50	0,125	0,106	19,750
	41996	1	234	123,08	1,04930	1,00	0,125	0,131	19,875
	42230	3	310	139,35	1,32480	1,50	0,188	0,166	20,000
	42540	1	121	119,01	-2,56200	-3,00	0,063	0,073	20,188
28	42661	2	327	132,11	2,70250	3,00	0,188	0,169	20,250
	42988	1	123	117,07	-2,65850	-3,00	0,063	0,071	20,438
	43111	3	621	92,75	5,04880	4,00	0,250	0,316	20,500
	43732	3	581	99,14	-1,06880	1,00	0,250	0,234	20,750
29	44313	3	220	87,27	-2,64090	-3,00	0,083	0,095	21,000
	44533	2	145	132,41	-1,51720	1,00	0,083	0,055	21,083
	44678	1	144	133,33	-1,00690	1,00	0,083	0,083	21,167
	44822	2	346	124,86	2,40280	2,25	0,188	0,200	21,250
	45168	1	160	90	-2,16250	-3,00	0,063	0,087	21,438
	45328	3	460	125,22	2,87500	4,00	0,250	0,180	21,500
30	45788	3	224	85,71	-2,05360	-3,00	0,083	0,122	21,750
	46012	1	104	184,62	-2,15380	1,00	0,083	0,039	21,833
	46116	1	139	138,13	1,33650	1,00	0,083	0,111	21,917
	46255	2	313	138,02	2,25180	2,25	0,188	0,188	22,000
	46568	1	128	112,5	-2,44530	-3,00	0,063	0,077	22,188
	46696	3	483	119,25	3,77340	4,00	0,250	0,236	22,250
31	47179	3	221	86,88	-2,18550	-3,00	0,083	0,114	22,500
	47400	1	96	200	-2,30210	1,00	0,083	0,036	22,583
	47496	1	148	129,73	1,54170	1,00	0,083	0,128	22,667
	47644	2	350	123,43	2,36490	2,25	0,188	0,197	22,750
	47994	1	127	113,39	-2,75590	-3,00	0,063	0,068	22,938

	48121	3	527	109,3	4,14960	4,00	0,250	0,259	23,000
32	48648	3	222	86,49	-2,37390	-3,00	0,083	0,105	23,250
	48870	1	130	147,69	-1,70770	1,00	0,083	0,049	23,333
	49000	1	204	94,12	1,56920	1,00	0,083	0,131	23,417
	49204	2	288	100	1,41180	1,50	0,125	0,118	23,500
	49492	1	384	75	1,33330	1,00	0,125	0,167	23,625
	49876	3	751	76,7	1,95570	2,00	0,250	0,244	23,750
33	50627	2	371	51,75	-2,02430	-3,00	0,083	0,123	24,000
	50998	1	156	123,08	-2,37820	1,00	0,083	0,035	24,083
	51154	1	184	104,35	1,17950	1,00	0,083	0,098	24,167
	51338	4	234	123,08	1,27170	1,50	0,125	0,106	24,250
	51572	1	252	114,29	1,07690	1,00	0,125	0,135	24,375
	51824	4	350	123,43	1,38890	1,50	0,188	0,174	24,500
	52174	1	144	100	-2,43060	-3,00	0,063	0,077	24,688
34	52318	2	312	138,46	2,16670	3,00	0,188	0,135	24,750
	52630	1	151	95,36	-2,06620	-3,00	0,063	0,091	24,938
	52781	3	602	95,68	3,98680	4,00	0,250	0,249	25,000
	53383	4	515	111,84	-1,16890	1,00	0,250	0,214	25,250
35	53898	1	190	101,05	-2,71050	-3,00	0,083	0,092	25,500
	54088	1	134	143,28	-1,41790	1,00	0,083	0,059	25,583
	54222	1	168	114,29	1,25370	1,00	0,083	0,104	25,667
	54390	4	208	138,46	1,23810	1,50	0,125	0,103	25,750
	54598	1	232	124,14	1,11540	1,00	0,125	0,139	25,875
	54830	3	330	130,91	1,42240	1,50	0,188	0,178	26,000
	55160	1	129	111,63	-2,55810	-3,00	0,063	0,073	26,188
36	55289	2	339	127,43	2,62790	3,00	0,188	0,164	26,250
	55628	1	174	82,76	-1,94830	-3,00	0,063	0,096	26,438
	55802	3	602	95,68	3,45980	4,00	0,250	0,216	26,500
	56404	2	593	97,13	-1,01520	1,00	0,250	0,246	26,750

A.13 Chopin, *Op. 6, Mazurka 1*, filtered clicktrack data

tempo of MIDI recorder =120bpm
ppq of MIDI file = 480 ticks/quarter

measure	pos [ticks]	#	perfIOI	tempo [bpm]	pIOIratio	sIOIratio	sIOI [beat]	sIOIrDur	scorePos [beat]
	0	1	740	77,84			0,250	0,250	-0,250
1	740	1	605	95,21	-1,22310	1,00	0,250	0,204	0,000
	1345	3	217	132,72	-2,78800	-2,00	0,125	0,090	0,250
	1562	1	247	116,60	1,13820	1,00	0,125	0,142	0,375
	1809	4	430	133,95	1,73680	2,00	0,250	0,217	0,500
2	2239	2	451	127,72	1,05130	1,00	0,250	0,263	0,750
	2690	3	603	95,52	1,33920	1,00	0,250	0,335	1,000
	3293	4	1033	111,52	1,71030	2,00	0,500	0,428	1,250
3	4326	4	216	133,33	-4,78240	-4,00	0,125	0,105	1,750
	4542	1	236	122,03	1,09720	1,00	0,125	0,137	1,875
	4778	3	423	136,17	1,78060	2,00	0,250	0,223	2,000
4	5201	2	426	135,21	1,00950	1,00	0,250	0,252	2,250
	5627	3	559	103,04	1,31460	1,00	0,250	0,329	2,500
	6186	3	578	99,65	1,03040	1,00	0,250	0,258	2,750
5	6764	3	184	104,35	-3,13590	-3,00	0,083	0,080	3,000
	6948	1	285	134,74	1,54350	2,00	0,167	0,129	3,083
	7233	2	335	128,96	1,18310	1,13	0,188	0,197	3,250
	7568	1	136	105,88	-2,47060	-3,00	0,063	0,076	3,438
	7704	3	561	102,67	4,12500	4,00	0,250	0,258	3,500
6	8265	2	431	133,64	-1,30470	1,00	0,250	0,192	3,750
	8696	2	448	128,57	1,04190	1,00	0,250	0,260	4,000
	9144	3	446	129,15	-1,00220	1,00	0,250	0,249	4,250
7	9590	3	456	126,32	1,02010	1,00	0,250	0,255	4,500
	10046	2	467	123,34	1,02410	1,00	0,250	0,256	4,750
	10513	3	555	103,78	1,18840	1,00	0,250	0,297	5,000
8	11068	3	571	100,88	1,02880	1,00	0,250	0,257	5,250
	11639	2	325	88,62	-1,75690	-2,00	0,125	0,142	5,500
	11964	1	440	65,45	1,35380	1,00	0,125	0,169	5,625
	12404	3	855	67,37	1,94320	2,00	0,250	0,243	5,750
9	13259	2	277	69,31	-3,08660	-3,00	0,083	0,081	6,000
	13536	1	381	100,79	1,37180	2,00	0,167	0,114	6,083
	13917	4	225	128,00	-1,68140	-1,33	0,125	0,099	6,250
	14142	1	256	112,50	1,13270	1,00	0,125	0,142	6,375
	14398	4	459	125,49	1,79690	2,00	0,250	0,225	6,500
10	14857	2	484	119,01	1,05220	1,00	0,250	0,263	6,750
	15341	3	615	93,66	1,26860	1,00	0,250	0,317	7,000
	15956	2	558	103,23	-1,10040	1,00	0,250	0,227	7,250
11	16514	1	420	137,14	-1,32860	1,00	0,250	0,188	7,500
	16934	4	220	130,91	-1,90910	-2,00	0,125	0,131	7,750
	17154	1	219	131,51	-1,00460	1,00	0,125	0,124	7,875
	17373	3	422	136,49	1,92690	2,00	0,250	0,241	8,000
12	17795	2	440	130,91	1,04270	1,00	0,250	0,261	8,250
	18235	3	547	105,30	1,24320	1,00	0,250	0,311	8,500
	18782	3	594	96,97	1,08590	1,00	0,250	0,271	8,750
13	19376	3	564	102,13	-1,05320	1,00	0,250	0,237	9,000
	19940	4	457	126,04	-1,23410	1,00	0,250	0,203	9,250
	20397	4	449	128,29	-1,01780	1,00	0,250	0,246	9,500
14	20846	3	531	108,47	1,18260	1,00	0,250	0,296	9,750
	21377	4	487	118,28	-1,09030	1,00	0,250	0,229	10,000
	21864	4	570	101,05	1,17040	1,00	0,250	0,293	10,250
15	22434	4	444	97,30	-1,28380	-1,33	0,188	0,195	10,500
	22878	1	182	79,12	-2,43960	-3,00	0,063	0,077	10,688
	23060	2	539	106,86	2,96150	4,00	0,250	0,185	10,750
	23599	3	705	81,70	1,30800	1,00	0,250	0,327	11,000
16	24304	4	403	95,29	-1,74940	-1,50	0,167	0,143	11,250
	24707	2	268	71,64	-1,50370	-2,00	0,083	0,111	11,417
	24975	2	857	67,21	3,19780	3,00	0,250	0,266	11,500
	25832	1	513	112,28	-1,67060	1,00	0,250	0,150	11,750
17	26345	4	453	127,15	-1,13250	1,00	0,250	0,221	12,000
	26798	2	220	130,91	-2,05910	-2,00	0,125	0,121	12,250
	27018	2	200	144,00	-1,10000	1,00	0,125	0,114	12,375
	27218	2	200	144,00	1,00000	1,00	0,125	0,125	12,500
	27418	2	190	151,58	-1,05260	1,00	0,125	0,119	12,625
18	27608	2	301	143,52	1,58420	1,50	0,188	0,198	12,750
	27909	2	93	154,84	-3,23660	-3,00	0,063	0,058	12,938
	28002	2	413	139,47	4,44090	4,00	0,250	0,278	13,000
	28415	2	484	119,01	1,17190	1,00	0,250	0,293	13,250
19	28899	4	515	111,84	1,06400	1,00	0,250	0,266	13,500

	29414	2	243	118,52	-2,11930	-2,00	0,125	0,118	13,750
	29657	2	210	137,14	-1,15710	1,00	0,125	0,108	13,875
	29867	2	186	154,84	-1,12900	1,00	0,125	0,111	14,000
	30053	2	195	147,69	1,04840	1,00	0,125	0,131	14,125
20	30248	2	174	165,52	-1,12070	1,00	0,125	0,112	14,250
	30422	2	184	156,52	1,05750	1,00	0,125	0,132	14,375
	30606	3	410	140,49	2,22830	2,00	0,250	0,279	14,500
	31016	1	184	156,52	-2,22830	-2,00	0,125	0,112	14,750
	31200	1	228	126,32	1,23910	1,00	0,125	0,155	14,875
21	31428	5	424	135,85	1,85960	2,00	0,250	0,232	15,000
	31852	2	201	143,28	-2,10950	-2,00	0,125	0,119	15,250
	32053	2	189	152,38	-1,06350	1,00	0,125	0,118	15,375
	32242	2	171	168,42	-1,10530	1,00	0,125	0,113	15,500
	32413	2	188	153,19	1,09940	1,00	0,125	0,137	15,625
22	32601	2	409	140,83	2,17550	2,00	0,250	0,272	15,750
	33010	2	452	127,43	1,10510	1,00	0,250	0,276	16,000
	33462	2	407	141,52	-1,11060	1,00	0,250	0,225	16,250
23	33869	4	390	147,69	-1,04360	1,00	0,250	0,240	16,500
	34259	2	212	135,85	-1,83960	-2,00	0,125	0,136	16,750
	34471	2	213	135,21	1,00470	1,00	0,125	0,126	16,875
	34684	2	220	130,91	1,03290	1,00	0,125	0,129	17,000
	34904	2	329	87,54	1,49550	1,00	0,125	0,187	17,125
24	35233	2	417	82,88	1,26750	1,20	0,150	0,158	17,250
	35650	2	356	97,08	-1,17130	1,00	0,150	0,128	17,400
	36006	2	490	70,53	1,37640	1,00	0,150	0,206	17,550
	36496	2	750	46,08	1,53060	1,00	0,150	0,230	17,700
	37246	1	314	55,03	-2,38850	-2,00	0,075	0,063	17,850
	37560	1	368	46,96	1,17200	1,00	0,075	0,088	17,925
25	37928	2	228	84,21	-1,61400	1,11	0,083	0,046	18,000
	38156	1	307	125,08	1,34650	2,00	0,167	0,112	18,083
	38463	2	223	129,15	-1,37670	-1,33	0,125	0,121	18,250
	38686	1	289	99,65	1,29600	1,00	0,125	0,162	18,375
	38975	1	693	83,12	2,39790	2,00	0,250	0,300	18,500
26	39668	1	499	115,43	-1,38880	1,00	0,250	0,180	18,750
	40167	3	674	85,46	1,35070	1,00	0,250	0,338	19,000
	40841	4	513	112,28	-1,31380	1,00	0,250	0,190	19,250
27	41354	1	419	137,47	-1,22430	1,00	0,250	0,204	19,500
	41773	4	223	129,15	-1,87890	-2,00	0,125	0,133	19,750
	41996	1	234	123,08	1,04930	1,00	0,125	0,131	19,875
	42230	3	431	133,64	1,84190	2,00	0,250	0,230	20,000
28	42661	2	450	128,00	1,04410	1,00	0,250	0,261	20,250
	43111	3	624	92,31	1,38670	1,00	0,250	0,347	20,500
	43735	2	578	99,65	-1,07960	1,00	0,250	0,232	20,750
29	44313	3	220	87,27	-2,62730	-3,00	0,083	0,095	21,000
	44533	2	288	133,33	1,30910	2,00	0,167	0,109	21,083
	44821	2	347	124,50	1,20490	1,13	0,188	0,201	21,250
	45168	1	160	90,00	-2,16880	-3,00	0,063	0,086	21,438
	45328	3	460	125,22	2,87500	4,00	0,250	0,180	21,500
30	45788	3	467	123,34	1,01520	1,00	0,250	0,254	21,750
	46255	2	441	130,61	-1,05900	1,00	0,250	0,236	22,000
	46696	3	483	119,25	1,09520	1,00	0,250	0,274	22,250
31	47179	3	465	123,87	-1,03870	1,00	0,250	0,241	22,500
	47644	2	477	120,75	1,02580	1,00	0,250	0,256	22,750
	48121	3	527	109,30	1,10480	1,00	0,250	0,276	23,000
32	48648	3	556	103,60	1,05500	1,00	0,250	0,264	23,250
	49204	2	288	100,00	-1,93060	-2,00	0,125	0,129	23,500
	49492	1	374	77,01	1,29860	1,00	0,125	0,162	23,625
	49866	3	761	75,69	2,03480	2,00	0,250	0,254	23,750
33	50627	2	711	81,01	-1,07030	1,00	0,250	0,234	24,000
	51338	4	234	123,08	-3,03850	-2,00	0,125	0,082	24,250
	51572	1	252	114,29	1,07690	1,00	0,125	0,135	24,375
	51824	4	494	116,60	1,96030	2,00	0,250	0,245	24,500
34	52318	2	462	124,68	-1,06930	1,00	0,250	0,234	24,750
	52780	3	604	95,36	1,30740	1,00	0,250	0,327	25,000
	53384	4	514	112,06	-1,17510	1,00	0,250	0,213	25,250
35	53898	1	324	118,52	-1,58640	-1,50	0,167	0,158	25,500
	54222	1	168	114,29	-1,92860	-2,00	0,083	0,086	25,667
	54390	4	208	138,46	1,23810	1,50	0,125	0,103	25,750
	54598	1	233	123,61	1,12020	1,00	0,125	0,140	25,875
	54831	3	457	126,04	1,96140	2,00	0,250	0,245	26,000
36	55288	2	514	112,06	1,12470	1,00	0,250	0,281	26,250
	55802	3	602	95,68	1,17120	1,00	0,250	0,293	26,500
	56404	3	593	97,13	-1,01520	1,00	0,250	0,246	26,750

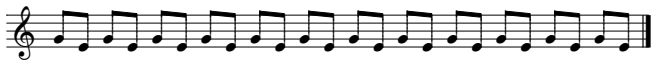
A.14 Evaluation of The Rhythm Perception Experiment

In the context of this thesis we performed a experiment focussing on the relation of melody and context and the perception of rhythm. In the online experiment participants were asked to listen to six MIDI files and select their preferred transcription from a multiple choice list. In addition to the pre-defined answers the participants could also decide to specify an own solution. Below the scores of the presented MIDI files and the corresponding multiple choice answers are shown. The content of tasks 5 and 6 cannot be expressed in a readable graphical score and are therefore presented in GUIDO syntax. Table A.1 to Table A.7 show the evaluation for all participants respectively for the single categories of participants.

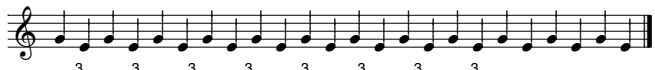
Description of tasks:

Task 1 MIDI file:

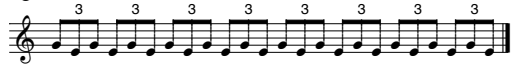
Answer A:



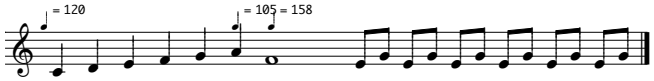
Answer B:



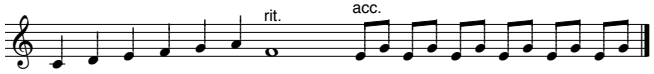
Answer C:



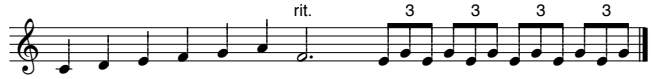
Task 2 MIDI file:



Answer A:

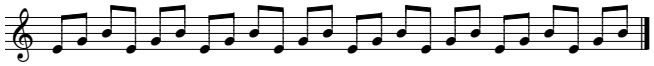


Answer B:

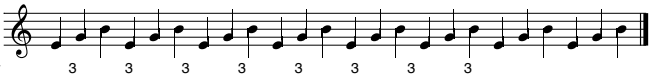


Task 3 MIDI file:

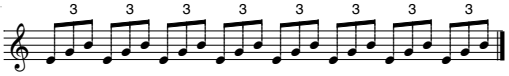
Answer A:



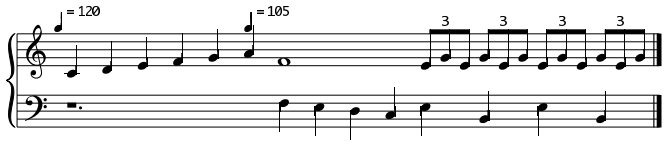
Answer B:



Answer C:



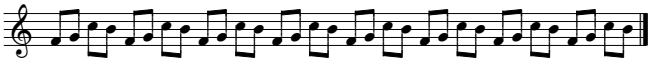
Task 4 MIDI file:



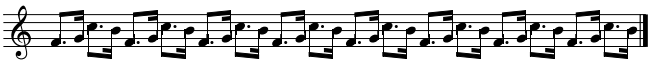
Answer A (transcription of first voice): equal to task 2, answer A
Answer B (transcription of first voice): equal to task 2, answer B

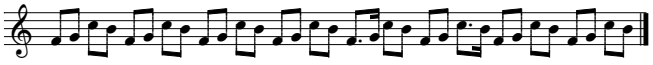
Task 5 MIDI file: [f*7/48 g*5/48 c2*7/48 h1*5/48 f*7/48 g*5/48 c2*7/48 h1*5/48
f*7/48 g*5/48 c2*7/48 h1*5/48 f*7/48 g*5/48 c2*7/48 h1*5/48
f*7/48 g*5/48 c2*7/48 h1*5/48 f*7/48 g*5/48 c2*7/48 h1*5/48
f*7/48 g*5/48 c2*7/48 h1*5/48 f*7/48 g*5/48 c2*7/48 h1*5/48]

Answer A:



Answer B:



Task 6 MIDI file: [f/8 g c2 h1 f/8 g c2 h1 f/8 g c2 h1 f/8 g c2 h1
f*7/48 g*5/48 c2/8 h1 f/8 g c2*7/48 h1*5/48 f/8 g c2 h1]
Answer A: equal to task 5 answer A
Answer B: 

Evaluation:

all	A	B	C	D	answers
question1	70	9	1	2	82
question2	60	21	-	0	81
question3	13	14	52	0	79
question4	18	58	-	1	77
question5	12	45	-	1	58
question6	13	45	-	0	58

Table A.1: Total number of answers of the rhythm perception experiment (D = user input).

other	A	B	C	D	answers
question1	10	3	0	0	13
question2	11	2	-	0	13
question3	1	1	10	0	12
question4	5	7	-	0	12
question5	2	8	-	0	10
question6	2	8	-	0	10

Table A.2: Answers of those participants who did not fit to any of the other categories (D = user input).

teacher	A	B	C	D	answers
question1	9	0	0	0	9
question2	7	1	-	0	8
question3	3	0	3	0	6
question4	1	4	-	0	5
question5	1	2	-	0	3
question6	1	2	-	0	3

Table A.3: Answers of the music teachers (D = user input).

student	A	B	C	D	answers
question1	8	0	1	2	11
question2	8	3	-	0	11
question3	1	2	8	0	11
question4	2	9	-	0	11
question5	2	6	-	0	8
question6	1	7	-	0	8

Table A.4: Answers of the participating students of music (D = user input).

amateur	A	B	C	D	answers
question1	30	3	0	0	33
question2	24	9	-	0	33
question3	4	10	20	0	34
question4	9	23	-	1	33
question5	3	22	-	0	25
question6	3	22	-	0	25

Table A.5: Answers of the amateur musicians (D = user input).

pro	A	B	C	D	answers
question1	9	3	0	0	12
question2	8	4	-	0	12
question3	4	1	7	0	12
question4	1	11	-	0	12
question5	3	4	-	1	8
question6	4	4	-	0	8

Table A.6: Answers of the professional musicians (D = user input).

professor	A	B	C	D	answers
question1	4	0	0	0	4
question2	2	2	-	0	4
question3	0	0	4	0	4
question4	0	4	-	0	4
question5	1	3	-	0	4
question6	2	2	-	0	4

Table A.7: Answers of the professors for music (D = user input).

A.15 PAM Scoring Matrix

PAM 30 Matrix

```
#
# This matrix was produced by "pam" Version 1.0.6 [28-Jul-93]
#
# PAM 30 substitution matrix, scale = ln(2)/2 = 0.346574
#
# Expected score = -5.06, Entropy = 2.57 bits
#
# Lowest score = -17, Highest score = 13
#
   A   R   N   D   C   Q   E   G   H   I   L   K   M   F   P   S   T   W   Y   V   B   Z   X   *
A  6  -7  -4  -3  -6  -4  -2  -2  -7  -5  -6  -7  -5  -8  -2   0  -1 -13  -8  -2  -3  -3  -3 -17
R -7   8  -6 -10  -8  -2  -9  -9  -2  -5  -8   0  -4  -9  -4  -3  -6  -2 -10  -8  -7  -4  -6 -17
N -4  -6   8   2 -11  -3  -2  -3   0  -5  -7  -1  -9  -9  -6   0  -2  -8  -4  -8   6  -3  -3 -17
D -3 -10   2   8 -14  -2   2  -3  -4  -7 -12  -4 -11 -15  -8  -4  -5 -15 -11  -8   6   1  -5 -17
C -6  -8 -11 -14  10 -14 -14  -9  -7  -6 -15 -14 -13 -13  -8  -3  -8 -15  -4  -6 -12 -14  -9 -17
Q -4  -2  -3  -2 -14   8   1  -7   1  -8  -5  -3  -4 -13  -3  -5  -5 -13 -12  -7  -3   6  -5 -17
E -2  -9  -2   2 -14   1   8  -4  -5  -5  -9  -4  -7 -14  -5  -4  -6 -17  -8  -6   1   6  -5 -17
G -2  -9  -3  -3  -9  -7  -4   6  -9 -11 -10  -7  -8  -9  -6  -2  -6 -15 -14  -5  -3  -5  -5 -17
H -7  -2   0  -4  -7   1  -5  -9   9  -9  -6  -6 -10  -6  -4  -6  -7  -7  -3  -6  -1  -1  -5 -17
I -5  -5  -5  -7  -6  -8  -5 -11  -9   8  -1  -6  -1  -2  -8  -7  -2 -14  -6   2  -6  -6  -5 -17
L -6  -8  -7 -12 -15  -5  -9 -10  -6  -1   7  -8   1  -3  -7  -8  -7  -6  -7  -2  -9  -7  -6 -17
K -7   0  -1  -4 -14  -3  -4  -7  -6  -6  -8   7  -2 -14  -6  -4  -3 -12  -9  -9  -2  -4  -5 -17
M -5  -4  -9 -11 -13  -4  -7  -8 -10  -1   1  -2  11  -4  -8  -5  -4 -13 -11  -1 -10  -5  -5 -17
F -8  -9  -9 -15 -13 -13 -14  -9  -6  -2  -3 -14  -4   9 -10  -6  -9  -4   2  -8 -10 -13  -8 -17
P -2  -4  -6  -8  -8  -3  -5  -6  -4  -8  -7  -6  -8 -10   8  -2  -4 -14 -13  -6  -7  -4  -5 -17
S  0  -3   0  -4  -3  -5  -4  -2  -6  -7  -8  -4  -5  -6  -2   6   0  -5  -7  -6  -1  -5  -3 -17
T -1  -6  -2  -5  -8  -5  -6  -6  -7  -2  -7  -3  -4  -9  -4   0   7 -13  -6  -3  -3  -6  -4 -17
W -13 -2  -8 -15 -15 -13 -17 -15  -7 -14  -6 -12 -13  -4 -14  -5 -13  13  -5 -15 -10 -14 -11 -17
Y -8 -10  -4 -11  -4 -12  -8 -14  -3  -6  -7  -9 -11   2 -13  -7  -6  -5  10  -7  -6  -9  -7 -17
V -2  -8  -8  -8  -6  -7  -6  -5  -6   2  -2  -9  -1  -8  -6  -6  -3 -15  -7   7  -8  -6  -5 -17
B -3  -7   6   6 -12  -3   1  -3  -1  -6  -9  -2 -10 -10  -7  -1  -3 -10  -6  -8   6   0  -5 -17
Z -3  -4  -3   1 -14   6   6  -5  -1  -6  -7  -4  -5 -13  -4  -5  -6 -14  -9  -6   0   6  -5 -17
X -3  -6  -3  -5  -9  -5  -5  -5  -5  -5  -6  -5  -5  -8  -5  -3  -4 -11  -7  -5  -5  -5  -5 -17
* -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17   1
```

Figure A.11: Example for a PAM scoring matrix used for BLAST. Source: `http://www.cmbi.kun.nl/bioinf/tools/pam.shtml`.

A.16 Combining Penalties

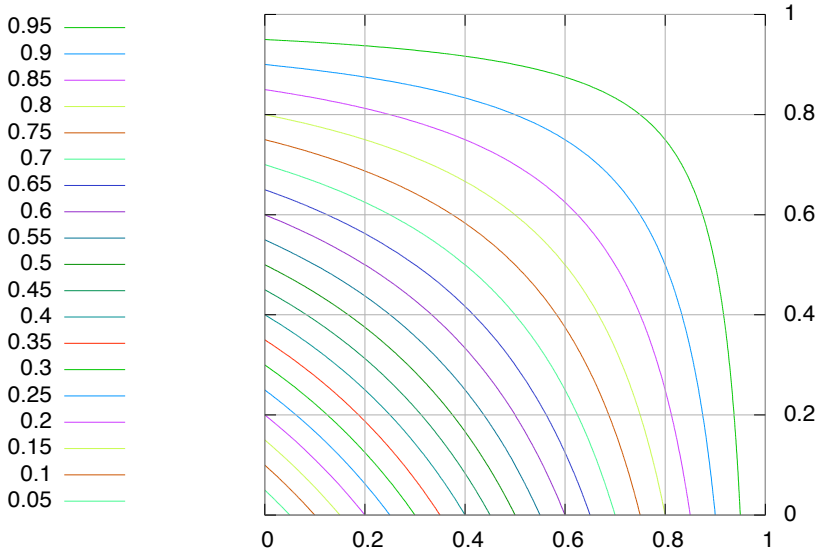


Figure A.12: Combining two penalties $x, y \in (0, 1)$: contour lines for penalty $p(x, y) = x + y - x \cdot y$.

Bibliography

[AGM+90] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, October 1990.

[AMS+97] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI–BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, 25:3389–3402, 1997.

[Bre90] Albert S. Bregman. *Auditory Scene Analysis*. The MIT Press, Cambridge (MA), USA, 1990.

[Bro93] Judith C. Brown. Determination of the meter of musical scores by autocorrelation. *Journal of the Acoustical Society of America*, 94(4):1953–1957, 1993.

[Cam00a] Emilios Cambouropoulos. Extracting 'significant' patterns from musical strings: Some interesting problems., 2000. Invited paper presented at London String Days 2000 workshop, 3-4 April 2000, King's College London, U.K. Download at: `http://www.oefai.at/cgi-bin/get-tr?paper=oefai-tr-2000-14.pdf`.

[Cam00b] Emilios Cambouropoulos. From MIDI to traditional musical notation. In *Proceedings of the AAAI Workshop on AI and Music, 30. July – 3, Aug. 2000, Austin, Texas*, 2000.

[Cam01a] Emilios Cambouropoulos. The local boundary detection model (LBDM) and its application in the study of expressive timing. In *Proceedings of the 2001 International Computer Music Conference (ICMC)*. Computer Music Association, San Francisco, 2001.

[Cam01b] Emilios Cambouropoulos. Automatic pitch spelling: From numbers to sharps and flats. In *Proceedings of the VIII Brazilian Symposium on Computer Music*. Fortaleza, Brasil, 31. July – 3. August 2001.

[CC03] Elaine Chew and Yun-Ching Chen. Determining context-defining windows: Pitch spelling using the spiral array. In *Proceedings of the 4th International Conference on Music Information Retrieval – ISMIR 2003*, 2003.

[CCI+99] E. Cambouropoulos, M. Crochemore, C. S. Iliopoulos, L. Mouchard, and Y. J. Pinzon. Algorithms for computing approximate repetitions in musical sequences. In R. Raman and J. Simpson, editors, *Proceedings of the 10th Australasian Workshop On Combinatorial Algorithms*, pages 129–144, Perth, WA, Australia, 1999.

[CDK00] A. T. Cemgil, P. Desain, and H. J. Kappen. Rhythm quantization for transcription. *Computer Music Journal*, 24(2):60–76, Summer 2000.

[Che02] Elaine Chew. The spiral array: An algorithm for determining key boundaries. In Christina Anagnostopoulou, Miguel Ferrand, and Alan Smaill, editors, *Proceedings of the Second International Conference on Music and Artificial Intelligence, ICMAI 2002, Edinburgh, Scotland, UK, September 12-14, 2002*, volume 2445 of *Lecture Notes in Computer Science*, pages 18–31. Springer, 2002.

[CIR98] T. Crawford, C. S. Iliopoulos, and R. Raman. String-matching techniques for musical similarity and melodic recognition. In W. B. Hewlett and E. Selfridge-Field, editors, *Melodic Similarity: Concepts, Procedures, and Applications*, volume 11 of *Computing in Musicology*, chapter 3, pages 73–100. The Center for Computer Assisted Research in the Humanities and The MIT Press, 1998.

[CK02] Ali Taylan Cemgil and Hilbert J. Kappen. Rhythm quantization and tempo tracking by sequential Monte Carlo. In Thomas G. Dietterich, Sue Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 1361–1368. The MIT Press, 2002.

[CK03] Ali Taylan Cemgil and Hilbert J. Kappen. Monte Carlo methods for tempo tracking and rhythm quantization. *Journal of Artificial Intelligence Research*, 18:45–81, 2003.

[CKDH01] A. T. Cemgil, H. J. Kappen, P. Desain, and H. Honing. On tempo tracking: Tempogram representation and Kalman filtering. *Journal of New Music Research*, 29(4), 2001.

[Cop96] David Cope. *Experiments in Musical Intelligence*, volume 12 of *The Computer Music And Digital Audio Series*. A-R Editions, Madison, Wisconsin, 1996.

[Cop03] David Cope. Computer analysis of musical allusions. *Computer Music Journal*, 27(1):11–28, Spring 2003.

[CST00] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[Dan84] Roger B. Dannenberg. An on-line algorithm for real-time accompaniment. In *Proceedings of the 1984 International Computer Music Conference (ICMC)*, pages 193–198. Computer Music Association, San Francisco, 1984.

[Dan01] Roger B. Dannenberg. Music information retrieval as music understanding. In *Proceedings of the 2nd International Conference on Music Information Retrieval – ISMIR 2001*, pages 139–142, 2001. Invited Address.

[Des92] Peter Desain. A (de)composable theory of rhythm perception. *Music Perception*, 9(4):439–454, 1992.

[Deu80] Diana Deutsch. The processing of structured and unstructured tonal sequences. *Perception and Psychophysics*, 28:381–389, 1980.

[DG02] Simon Dixon and Werner Goebl. Pinpointing the beat: Tapping to expressive performances. In *Proceedings of the 7th International Conference on Music Perception and Cognition (ICMPC)*, July 2002.

[DGW02] S. Dixon, W. Goebl, and G. Widmer. Real-time tracking and visualisation of musical expression. In *Proceedings of the II International Conference on Music and Artificial Intelligence*, pages 58–68, Edinburgh, Scotland, 12 - 15 September 2002.

[DH89] Peter Desain and Henkjan Honing. The quantization of musical time: A connectionist approach. *Computer Music Journal*, 13(3):56–66, 1989. Reprinted In P. M. Todd and D. G. Loy (eds.), *Music and Connectionism*, pages 150–167, The MIT Press. 1991.

[DH91] Peter Desain and Henkjan Honing. Tempo curves considered harmful - a critical review of the representation of timing in computer music. In *Proceedings of the 1991 International Computer Music Conference (ICMC)*, pages 143–149. Computer Music Association, San Francisco, 1991.

[DH02] Roger B. Dannenberg and Ning Hu. Pattern discovery techniques for music audio. In *Proceedings of the 3rd International Conference on Music Information Retrieval – ISMIR 2002*, pages 63–70, 2002.

[Dix01a] Simon Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30(1), 2001.

[Dix01b] Simon Dixon. An empirical comparison of tempo trackers. In *Proceedings of the 8th Brazilian Symposium on Computer Music Fortaleza, Brazil*, pages 832–840, 2001.

[Dix01c] Simon Dixon. An interactive beat tracking and visualization system. In *Proceedings of the 2001 International Computer Music Conference (ICMC)*, pages 215–218. Computer Music Association, San Francisco, 2001.

[DMR87] Roger B. Dannenberg and Bernard Mont-Reynaud. Following an improvisation in real-time. In *Proceedings of the 1987 International Computer Music Conference (ICMC)*, pages 241–248. Computer Music Association, San Francisco, 1987.

[Dov99] Matthew J. Dovey. An algorithm for locating polyphonic phrases within a polyphonic musical piece. In *Proceedings of the AISB'99 Symposium on Musical Creativity*, pages 48–53, 1999.

[Dov01] Matthew J. Dovey. A technique for "regular expression" style searching in polyphonic music. In *Proceedings of the 2nd International Conference on Music Information Retrieval – ISMIR 2001*, pages 179–185. Bloomington, IN, USA, October 2001.

[DPB00] Carolyn Drake, Amandine Penel, and Emmanuel Bigand. Why musicians tap slower than nonmusicians. In P. Desain and L. Windsor, editors, *Rhythm perception and production*, pages 245–248. Swets & Zeitlinger, 2000.

[Dri91] Anthonic Driesse. Real-time tempo tracking using rules to analyze rhythmic qualities. In *Proceedings of the 1991 International Computer Music Conference (ICMC)*, pages 578–581. Computer Music Association, San Francisco, 1991.

[FM00] Ichiro Fujinaga and Karl MacMillan. Realtime recognition of orchestral instruments. In *Proceedings of the 2000 International Computer Music Conference (ICMC)*, pages 241–243. Computer Music Association, San Francisco, 2000.

[Fra82] Paul Fraisse. Rhythm and tempo. In D. Deutsch, editor, *The Psychology of Music*. Academic Press, New York, 1982.

[Fuj96] Ichiro Fujinaga. Exemplar-based learning in adaptive optical music recognition system. In *Proceedings of the 1996 International Computer Music Conference (ICMC)*, pages 55–56. Computer Music Association, San Francisco, 1996.

[Gje94] Robert O. Gjerdingen. Apparent motion in music? *Music Perception*, 11:335–370, 1994.

[GM94] Masataka Goto and Yoichi Muraoka. A beat tracking system for acoustic signals of music. In *Proceedings of the Second ACM International Conference on Multimedia*, pages 365–372, October 1994.

[GM99] Masataka Goto and Yoichi Muraoka. Real-time beat tracking for drumless audio signals: Chord change detection for musical decisions. *Speech Communication*, 27(3–4):311–335, April 1999.

[Got01] Masataka Goto. An audio-based real-time beat tracking system for music with or without drum-sounds. *Journal of New Music Research*, 30(2):159–171, June 2001.

[GUI] GUIDO*Lib*. `http://guidolib.sourceforge.net`.

[Gus97] Dan Gusfield. *Algorithms on Strings, Tries, And Sequences – Computer Science And Computational Biology*. Press Syndicate of the University of Cambridge, 1997.

[Har03] Steven Harford. Automatic segmentation, learning and retrieval of melodies using a self-organizing neural network. In *Proceedings of the 4th International Conference on Music Information Retrieval – ISMIR 2003*, 2003.

[HDL02] N. Hu, R. B. Dannenberg, and Ann L. Lewis. A probabilistic model of melodic similarity. In *Proceedings of the 2002 International Computer Music Conference (ICMC)*. Computer Music Association, San Francisco, 2002.

[Hew97] Walter B. Hewlett. Musedata: Multipurpose representation. In E. Selfridge-Field, editor, *Beyond MIDI: the handbook of musical codes*, pages 402–450. The MIT Press, 1997.

[HHFK98] Holger H. Hoos, Keith Hamel, Kai Flade, and Jürgen Kilian. The GUIDO music notation format – a novel approach for adequately representing score-level music. In *Proceedings of the 1998 International Computer Music Conference (ICMC)*, pages 451–454. Computer Music Association, San Francisco, 1998.

[HHRK01] Holger H. Hoos, Keith Hamel, Kai Renz, and Jürgen Kilian. Representing score level music using the GUIDO music notation format. In Walter B. Hewlett and Eleanor Selfridge-Field, editors, *The Virtual Score*, volume 12 of *Computing in Musicology*, chapter 5, pages 75–94. The Center for Computer Assisted Research in the Humanities and The MIT Press, 2001.

[Hoo99] Holger H. Hoos. *Stochastic Local Search – Methods, Models, Applications*. Infix Verlag, 1999.

[HRG01] Holger H. Hoos, Kai Renz, and Marko Görg. GUIDO/MIR an experimental musical information retrieval system based on GUIDO music notation. In *Proceedings of the 2nd International Conference on Music Information Retrieval – ISMIR 2001*, pages 41–50. Bloomington, IN, USA, October 2001.

[HSF97] Walter B. Hewlett and Eleanor Selfridge-Field. MIDI. In E. Selfridge-Field, editor, *Beyond MIDI: the handbook of musical codes*, pages 41–72. The MIT Press, 1997.

[Hur01] David Huron. Tone and voice: A derivation of the rules of voice-leading from perceptual principles. *Music Perception*, 19(1):1–64, 2001.

[ILMP00] C. S. Iliopoulos, T. Lecroq, L. Mouchard, and Y. J. Pinzon. Computing approximate repetitions in musical sequences. In M. Balík and M. Šimánek, editors, *Proceedings of the Prague Stringology Club Workshop 2000, Bratislava, Slovakia*, pages 49–59, Czech Technical University, Prague, Czech Republic, 2000.

[KH02] Jürgen Kilian and Holger H. Hoos. Voice separation – a local optimisation approach. In *Proceedings of the 3rd International Conference on Music Information Retrieval – ISMIR 2002*, pages 39–46. IRCAM – Centre Pomdidou, Paris, France, 2002.

[KH04] Jürgen Kilian and Holger H. Hoos. MusicBLAST – gapped sequence alignment for MIR. In *Proceedings of the 5th International Conference on Music Information Retrieval – ISMIR 2004*, pages 38–41, 2004.

[KHI+01] Takashi Kadota, Masahiro Hirao, Akira Ishino, Masayuki Takeda, Ayumi Shinohara, and Fumihiro Matsuo. Musical sequence comparison for melodic and rhythmic similarities. In *Proceedings of the 8th International Symposium on String Processing and Information Retrieval (SPIRE2001)*, pages 435–440. IEEE Computer Society, 2001.

[Kil96] Jürgen Kilian. FERMATA - flexible tempo detection for MIDI-file quantization. Diplomarbeit, FG AFS – Fachbereich Informatik, Technische Universität Darmstadt, 1996.

[Lar94] Edward W. Large. The resonant dynamics of beat tracking and meter perception. In *Proceedings of the 1994 International Computer Music Conference (ICMC)*, pages 90–91. Computer Music Association, San Francisco, 1994.

[Lem00] Kjell Lemström. *String Matching Techniques for Music Retrieval*. PhD thesis, University of Helsinki, Department of Computer Science, November 2000. Download at: `http://www.cs.helsinki.fi/u/klemstro/THESIS/`.

[LH76] H. Christopher Longuet-Higgins. Perception of melodies. *Nature*, 263:646–653, 1976. Reprinted in H. C. Longuet-Higgins *Mental Processes. Studies in Cognitive Science*, The MIT Press, 1987. Reprinted in Stephan Schwanauer and David Levitt, editors, *Machine Models of Music*, pp. 472–496, The MIT Press, 1993.

[LHL82] H. Christopher Longuet-Higgins and Christopher S. Lee. Perception of musical rhythms. *Perception*, 11:115–128, 1982.

[LHL84] H. Christopher Longuet-Higgins and Christopher S. Lee. The rhythmic interpretation of monophonic music. *Music Perception*, 1(4):424–441, 1984.

[LHU98] Kjell Lemström, Atso Haapaniemi, and Esko Ukkonen. Retrieving music – to index or not to index. *ACM Multimedia '98*, 1998.

[LJ83] Fred Lerdahl and Ray Jackendoff. *A Generative Theory of Tonal Music.* The MIT Press, Cambridge, 1983.

[LJ99] Edward W. Large and Mari Riess Jones. The dynamics of attending: How people track time-varying events. *Psychological Review*, 106(1):119–159, 1999.

[LK94] Edward W. Large and J. F. Kolen. Resonance and the perception of musical meter. *Connection Science*, 6(1):177–208, 1994.

[LL98] Kjell Lemström and Pauli Laine. Musical information retrieval using musical parameters. In *Proceedings of the 1998 International Computer Music Conference (ICMC)*. Computer Music Association, San Francisco, 1998.

[LP00] Kjell Lemström and Sami Perttu. SEMEX – an efficient music retrieval prototype. In *Proceedings of the 1st International Conference on Music Information Retrieval – ISMIR 2000*, 2000.

[LP02] Edward W. Large and Caroline Palmer. Perceiving temporal regularity in music. *Cognitive Science*, 26:1–37, 2002.

[LPP95] Edward W. Large, Caroline Palmer, and Jordan B. Pollack. Reduced memory representations for music. *Cognitive Science*, 19:53–96, 1995.

[Mac02] W. James MacLean. A Bayesian technique for distinguishing between melody notes and grace notes in recorded performances. In *Proceedings of the 2002 International Computer Music Conference (ICMC)*, pages 368–375, Göteborg, Sweden, September 16-21 2002. Computer Music Association, San Francisco.

[Mar01] Matija Marolt. SONIC: Transcription of polyphonic piano music with neural networks. In *Proceedings of Workshop on Current Research Directions in Computer Music*, pages 217–224, November 15-17 2001.

[Mar04] Matija Marolt. A connectionist approach to transcription of polyphonic piano music. *IEEE Transactions on Multimedia*, 6(3):439–449, June 2004.

[MB01] Colin Meek and William P. Birmingham. Thematic extractor. In *Proceedings of the 2nd International Conference on Music Information Retrieval – ISMIR 2001*, pages 119–128. Bloomington, IN, USA, October 2001.

[MD97] Susan L. McCabe and Michael J. Denham. A model of auditory streaming. *Journal of the Acoustical Society of America*, 101(3):1611–1621, 1997.

[MD01] Dominic Mazzoni and Roger B. Dannenberg. Melody matching directly from audio. In *Proceedings of the 2nd International Conference on Music Information Retrieval – ISMIR 2001*, 2001.

[Mer03] David Meredith. Method of computing the pitch names of notes in MIDI-like music representations, 2003. UK Patent application filed on 11 April 2003. Application number 0308456.3. Download at: `http://www.titanmusic.com/papers/public/ps13-patent-1.pdf`.

[Meu02] Benoit Meudic. Automatic meter extraction from MIDI files. In *Proceedings of the Journèes d'Informatique Musicale 2002*, pages 247–256, 2002.

[MLW03] David Meredith, Kjell Lemström, and Geraint Wiggins. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music, 2003. Cambridge Music Processing Colloquium 2003, Department of Engineering, University of Cambridge. Download at: `http://www.titanmusic.com/papers/public/cmpc2003.pdf`.

[MO03] Massimo Melucci and Nicola Orio. A comparison of manual and automatic melody segmentation. In *Proceedings of the 3rd International Conference on Music Information Retrieval – ISMIR 2002*, pages 7–14, 2003.

[Mor99] Kenneth John Morrison. *A Polymetric Interpretation of the Swing Impulse: Rhythmic Stratification in Jazz*. PhD thesis, University of Washington, 1999.

[MRRC82] Bernard Mont-Reynaud, Loron Rush, and Chris Chafe. Toward an intelligent editor of digital audio: Recognition of musical constructs. *Computer Music Journal*, 6(1):30–41, 1982.

[MS90] Marcel Mongeau and David Sankoff. Comparison of musical sequences. *Computers and the Humanities*, 24:161–175, 1990.

[MWL01] David Meredith, Geraint A. Wiggins, and Kjell Lemström. Pattern induction and matching in polyphonic music and other multidimensional data sets. In *Proceedings of the 5th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI2001)*, volume X, pages 61–66, July 22–25 2001.

[NN04] Han-Wen Nienjuys and Jan Nieuwenhuizen. GNU LilyPond - the music typesetter. The newest tutorial and reference manual available at `http://www.lilypond.org`, August 2004.

[Ohy94] Ken'ichi Ohya. A rhythm perception model by neural rhythm generators. In *Proceedings of the 1994 International Computer Music Conference (ICMC)*. Computer Music Association, San Francisco, 1994.

[Par94a] Richard Parncutt. A model of beat induction accounting for perceptual ambiguity by continuously variable parameters. In *Proceedings of the 1994 International Computer Music Conference (ICMC)*, pages 83–84. Computer Music Association, San Francisco, 1994.

[Par94b] Richard Parncutt. A perceptual model of pulse salience and metrical accent in musical rhythms. *Music Perception*, 11(4):409–464, 1994.

[PB01] Bryan Pardo and William P. Birmingham. Following a musical performance from a partially specified score. In *Proceedings of MTAC 2001*. IEEE, 2001.

[PB02] Bryan Pardo and William P. Birmingham. Improved score following for acoustic performances. In *Proceedings of the 2002 International Computer Music Conference (ICMC)*. œicmcpub, 2002.

[PE03] Mitchell Parry and Irfan Essa. Rhythmic similarity through elaboration. In *Proceedings of the 4th International Conference on Music Information Retrieval – ISMIR 2003*, pages 251–252. Johns Hopkins University, October 26-30 2003.

[PK90] Caroline Palmer and Carol L. Krumhansl. Mental representations for musical meter. *Journal of Experimental Psychology: Human Perception and Performance*, 16(4):728–741, 1990.

[PK02] Jouni Paulus and Anssi Klapuri. Measuring the similarity of rhythmic patterns. In *Proceedings of the 3rd International Conference on Music Information Retrieval – ISMIR 2002*, pages 150–156, 2002.

[PL93] Jeffrey Pressing and Peter Lawrence. Transcribe: A comprehensive autotranscription program. In *Proceedings of the 1993 International Computer Music Conference (ICMC)*, pages 343–345. Computer Music Association, San Francisco, 1993.

[Pov77] Dirk-Jan Povel. Temporal structure of performed music. some preliminary observations. *Acta Psychologica*, 41:309–320, 1977.

[Puc91] Miller Puckette. Combining event and signal processing in the MAX graphical programming environment. *Computer Music Journal*, 15(3):68–77, 1991.

[Rap01a] Christopher Raphael. Coarse-to-fine dynamic programming. *IEEETPAMI: IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23, 2001.

[Rap01b] Cristopher Raphael. Automated rhythm transcription. In *Proceedings of the 2nd International Conference on Music Information Retrieval – ISMIR 2001*, pages 99–107. Bloomington, IN, USA, October 2001.

[Rap02] Christopher Raphael. Automatic transcription of piano music. In *Proceedings of the 3rd International Conference on Music Information Retrieval – ISMIR 2002*, pages 15–19, 2002.

[Ren02] Kai Renz. *Algorithms and Data Structures for a Music Notation System based on* GUIDO *Music Notation*. PhD thesis, Fachbereich Informatik Technische Universität Darmstadt, Germany, 2002.

[RG94] Simon Roberts and Mike Greenhough. The detection of rhythmic repetition using a self-organised neural network. In *Proceedings of the 1994 International Computer Music Conference (ICMC)*, pages 125–128. Computer Music Association, San Francisco, 1994.

[RG02] Pierre-Yves Rolland and Jean-Gabriel Ganascia. Pattern detection and discovery: The case of music data mining. In D.J. Hand, N. M. Adams, and R. J. Bolton, editors, *Pattern Detection and Discovery. Lecture Notes in Artificial Intelligence (LNAI) Vol. 2447*, pages 190–198, 2002.

[RGM94] David Rosenthal, Masataka Goto, and Yoichi Muraoka. Rhythm tracking using multiple hypothesis. In *Proceedings of the 1994 International Computer Music Conference (ICMC)*, pages 85–88. Computer Music Association, San Francisco, 1994.

[Rip03] Gregory M. Rippin. *Who is Listening? Human and Computer Influence in Interactive Music Systems*. Master's Thesis, Department of Music and Performing Arts Professions in the Steinhardt School of Education, New York University, April 2003. Download at: `http://homepages.nyu.edu/~gmr222/interactive/WhoIsListening.pdf`.

[RL94] Robert Rowe and Tang-Chun Li. Pattern processing in music. In *Proceedings of the 1994 International Computer Music Conference (ICMC)*, pages 60–62. Computer Music Association, San Francisco, 1994.

[Rob96] Simon C. Roberts. *Interpreting Rhythmic Structures Using Artificial Neural Networks*. PhD thesis, Department of Physics and Astronomy University of Wales, College of Cardiff, September 1996.

[Rol99] Pierre-Yves Rolland. Discovering patterns in musical sequences. *Journal of New Music Research*, 28(4):334–350, 1999.

[Row93] Robert Rowe. *Interactive music systems: machine listening and composing*. The MIT Press Cambridge, MA, USA, 1993.

[Row01] Robert Rowe. *Machine Musicianship*. The MIT Press Cambridge, MA, USA, 2001.

[Sep01] Jarno Seppänen. *Computational models of meter recognition*. Master of Science thesis, Tampere University of Technology, 2001.

[SF97] Eleanor Selfridge-Field. DARMS, its dialects, and its uses. In E. Selfridge-Field, editor, *Beyond MIDI: the handbook of musical codes*, chapter 11, pages 163–174. The MIT Press, 1997.

[SMJ89] Don L. Scarborough, Ben O. Miller, and Jacqueline A. Jones. Connectionist models for tonal analysis. *Computer Music Journal*, 13(3):49–65, 1989. Reprinted in P M. Todd, D. G. Loy (eds.), *Music and Connectionism*, pages 54–60, The MIT Press, 1991.

[Smo94] Stephen W. Smoliar. Modelling music perception: A critical view. *Connection Science*, 6(2–3):209–222, 1994.

[SMW98] L. A. Smith, R. J. McNab, and I. H. Witten. Sequence-based melodic comparison: a dynamic programming approach. In W. B. Hewlett and E. Selfridge-Field, editors, *Melodic Similarity: Concepts, Procedures, and Applications*, volume 11 of *Computing in Musicology*, chapter 3, pages 101–117. The Center for Computer Assisted Research in the Humanities and The MIT Press, 1998.

[SP93] Dale R. Stammen and Bruce Pennycook. Real-time recognition of melodic fragments using the dynamic timewarp algorithm. In *Proceedings of the 1993 International Computer Music Conference (ICMC)*, pages 232–235. Computer Music Association, San Francisco, 1993.

[SP00] Ilya Shmulevich and Dirk-Jan Povel. Complexity measures of musical rhythms. In P. Desain and L. Windsor, editors, *Rhythm perception and production*, pages 239–244. Swets & Zeitlinger, 2000.

[SS68] Herbert A. Simon and Richard K. Summer. Pattern in music. In B. Kleinmutz, editor, *Formal Representations of Human Judgement*. Wiley, 1968. Reprinted in Stephan Schwanauer and David Levitt, editors, *Machine Models of Music*, pp. 84–110, The MIT Press, 1993.

[TAD+02] R. Timmer, R. Ashley, P. W. M. Desain, H. J. Honing, and W. L. Windsor. Timing of ornaments in the theme of Beethoven's Paisiello variations: Empirical data and a model. *Music Perception*, 20(1):3–33, 2002.

[Tan93] Andranick Tanguiane. An artificial perception model and its application to music recognition. In *Proceedings of the 1993 International Computer Music Conference (ICMC)*, pages 284–291. Computer Music Association, San Francisco, 1993.

[TC02] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions On Speech And Audio Processing*, 10(5):293–302, July 2002.

[Tem01] David Temperley. *The Cognition of Basic Musical Structures*. The MIT Press, Cambridge (MA), USA, 2001.

[TME99] Daniel Taupin, Ross Mitchell, and Andreas Egler. MusiXTEX - using TEX to write polyphonic or instrumental music, April 1999.

[TNS03] Haruto Takeda, Takuya Nishimoto, and Shigeki Sagayama. Automatic transcription of multiphonic MIDI signals. In *Proceedings of the 4th International Conference on Music Information Retrieval – ISMIR 2003*, pages 263–264, 2003.

[Tod85] Neil Todd. A model of expressive timing in tonal music. *Music Perception*, 3(1):33–58, 1985.

[TS] David Temperley and Daniel Sleator. The Melisma music analyzer. Download at: `http://www.links.cs.cmu.edu/music-analysis`.

[TSH02] Belinda Thom, Christian Spevak, and Karin Höthker. Melodic segmentation: Evaluating the performance of algorithms and musical experts. In *Proceedings of the 2002 International Computer Music Conference (ICMC)*. Computer Music Association, San Francisco, 2002.

[Vap95] Vladimir Naumovich Vapnik. *The nature of Statistical Learning Theory*. Springer Verlag, 1995.

[WAD+00] L. Windsor, R. Aarts, P. Desain, H. Heijink, and R. Timmers. On time: the influence of tempo structure and style on the timing of gracenotes in skilled musical performance. In P. Desain and L. Windsor, editors, *Rhythm perception and production*, pages 217–224. Swets & Zeitlinger, 2000.

[WAD+01] L. Windsor, R. Aarts, P. Desain, H. Heijink, and R. Timmers. The timing of grace notes in skilled musical performance at different tempi: A preliminary case study. *Psychology of Music*, 29:149–169, 2001.

[Wid95] Gerhard Widmer. Modelling the rational basis of musical expression. *Computer Music Journal*, 19(2):77–96, Summer 1995.

[Win68] Terry Winograd. Linguistics and the computer analysis of tonal harmony. *Journal of Music Theory*, 12(1):2–49, 1968. Reprinted in M. A. K. Halliday and J. R. Martin, editors, *Readings in Systemic Linguistics*, pp. 257–270, London, Batsford Academic, 1981. In Stephan Schwanauer and David Levitt, editors, *Machine Models of Music*, pp. 113–153, The MIT Press, 1993.

[Yes76] Maury Yeston. *The stratification of musical rhythm*. Yale University Press, 1976.

[Zan04] Damián H. Zanette. Zipf's law and the creation of musical context. *submitted for publication*, 2004. `http://www.arxiv.org/abs/cs.CL/0406015`.

[ZP03] Patrick Zano and Giovanni De Poli. Estimation of parameters in rule systems for expressive rendering in musical performance. *Computer Music Journal*, 27(1):29–46, Spring 2003.

List of Figures

List of Symbols / Abbreviations

attackpoint	onset time of a note
bpm	beats per minute
BSA	Boundary Search Algorithm
crotchet	note with a length of a quarter note
DP	dynamic programming
DTW	dynamic time warping
gcd	greatest common divisor
GTTM	Generative Theory of Tonal Music
GUI	graphical user interface
HMM	Hidden Markov Model
IOI	inter-onset interval
IOI ratio	ratio of two inter-onset intervals (IOIs)
LBDM	local boundary detection model
MIDI	Musical Instruments Digital Interface
minim	note with the duration of two crotchets
MIR	music information retrieval
onset time	start position of a note
offset	endpoint of a note, equal to onset time + duration
OCR	optical character recognition
OMR	optical music recognition
ppq	parts (MIDI ticks) per quarter
quaver	note with the duration of an eighth note
semi-quaver	note with the duration of a sixteenth note
SMF	Standard MIDI File
SVM	support vector machine

Index

Curriculum vitae

1968	Gebohren in Bensheim/Bergstraße
1975–1979	Müller-Guttenbrunn Schule (Grundschule) Fürth/Odenwald
1979–1988	Martin-Luther-Schule (Gymnasium) Rimbach; Abschluss: Abitur
1988–1989	Grundwehrdienst in Mannheim
WS 1989/90 – WS 1996/97	Studium der Informatik an der TU Darmstadt
Oktober 1996	Diplom Informatik
seit 1994	Mitarbeit im Rahmen des SALIERI-Projektes am FG Automatentheorie und Formale Sprachen (Prof. Dr. H. K.-G. Walter)
seit 1999	Lehrtätigkeiten im Rahmen der Vorlesung 'Einführung in die Computermusik', FB Informatik, TU Darmstadt
1996–2002	Geschäftsführer und Inhaber der Bäckerei und Konditorei Café Jakob
2003–2004	Mitarbeiter am FG Automatentheorie und Formale Sprachen (Prof. Dr. H. K.-G. Walter)
WS 2004/05	Lehrauftrag 'Einführung in die Computermusik', FB Informatik, TU Darmstadt

Erklärung

Hiermit erkläre ich, die vorliegende Arbeit zur Erlangung des akademischen Grades Dr. rer. nat. mit dem Titel "Inferring Score Level Musical Information From Low-Level Musical Data" selbstständig und ausschließlich unter Verwendung der angegebenen Hilfsmittel erstellt zu haben. Ich habe bisher noch keine Promotionsversuche unternommen.

Darmstadt, den

(Jürgen F. Kilian)