A MODEL OF RHYTHM TRANSCRIPTION AS PATH SELECTION THROUGH APPROXIMATE COMMON DIVISOR GRAPHS

Gonzalo Romero-García, Corentin Guichaoua, Elaine Chew
Sorbonne Université, Ircam, CNRS, Ministère de la Culture
Sciences and Technologies of Music and Sound Laboratory, Paris, France
{gonzalo.romero,corentin.guichaoua,elaine.chew}@ircam.fr

ABSTRACT

We apply the concept of approximated common divisors (ACDs) to estimate the tempo and quantize the durations of a rhythmic sequence. The ACD models the duration of the tatum within the sequence, giving its rate in beats per minute. The rhythm input, a series of timestamps, is first split into overlapping frames. Then, we compute the possible ACDs that fit this frame and build a graph with the candidate ACDs as nodes. By building this graph, we transform the quantization problem into one of path selection, where the nodes represent the ACDs and determine the note values of the transcription and the edges represent tempo transitions between frames. A path through the graph thus corresponds to a rhythm transcription. For path selection, we present both an automated method using weights for evaluating the transcription and finding the shortest path, and an interactive approach that gives users the possibility of influencing the path selection.

1. INTRODUCTION

Many techniques have been proposed for the problem of rhythm transcription and quantization: some build a rhythm tree [1, 2], some use probabilistic models and Monte Carlo pruning [3], and others use signal processing methods like autocorrelation [4]. Here, we present yet another approach to the problem using the notion of *approximate common divisors* (ACDs). This mathematical object, stemming from cryptography [5], is related to the rhythmic notion of tatum, which will be used to transcribe a series of onset times into a musical rhythm. A similar technique has been used to find the greatest common divisor of an inter-onset interval histogram [6]. The difference in our approach is that we consider also common divisors that are not the greatest, and we build a graph such that each candidate common divisor maps to a node in the graph.

This approach was first explored in [7] to automate the transcription of a rhythmic sequence in expressive music performances and arrhythmic heartbeats into musical notation in the context of the ERC project COSMOS (`http://cosmos.ircam.fr`). Here, we describe the original problem and algorithmic approach, which addresses a monophonic rhythm input, then propose an extension that can handle polyphonic input.

The main observation driving the algorithm design is that a given rhythmic input may have several transcription possibilities. We model the options using a graph where each path corresponds to a possible transcription. The choice of the path may be automated but it may also be selected interactively with a human in the loop. We will present both approaches and consider how they might coexist and cooperate within the same framework.

This paper is organised as follows: Section 2 gives an introduction to approximate common divisors, explaining how they can be used to quantize a rhythm by creating a temporal grid. After defining ACDs, we first address the problem of transcribing monophonic rhythms in Section 3, then broaden the description to the transcription of polyphonic rhythms in Section 4. While both cases can be approached using the same technique, they differ in how the input is split. Finally, in Section 5, we propose a graphical user interface that allows a user to select the path and thus influence the transcription result.

2. APPROXIMATE COMMON DIVISORS

The notion of an *approximate common divisor* (ACD) has been studied by Howgrave-Graham in [5], with integral ACDs in the context of cryptography. In this paper, we will allow ACDs to take on real values with a slightly different definition adapted to the purpose of rhythm transcription. Let us then provide an exposition on the ACD finding problem, presenting it in a form suitable for the rhythm transcription context, working our way up from the problem of finding (exact) common divisors.

2.1 The ACD finding problem in $\mathbb{R}$

The common divisor finding problem in $\mathbb{N}$ is that of finding the common divisors of a series of N natural numbers,

$$D = (d_1, d_2, ..., d_N) \in \mathbb{N}^N \,. \tag{1}$$

This problem is equivalent to finding the numbers $a \in \mathbb{N}$ such that $\forall n \in \{1, ..., N\}$, $d_n \in a\mathbb{N} \triangleq \{am \in \mathbb{N} : m \in \mathbb{N}\}$. This problem, which is well posed in $\mathbb{N}$, can be extended to $\mathbb{R}$ with some adaptations.

First of all, we will allow both a and the durations d_n to be positive real numbers, *i.e.*: $a > 0$, $d_n > 0$, $\forall n \in \{1, ..., N\}$. Then, we define the a-grid as

$$a\mathbb{Z} \triangleq \{am \in \mathbb{R} : m \in \mathbb{Z}\} \,. \tag{2}$$

This new framework allows us to extend the problem to a series of timestamps,

$$T = (t_0, t_1, ..., t_N) \in \mathbb{R}^{N+1}\,, \tag{3}$$

where $t_0 = 0$ and $\forall n \in \{1, ..., N\}, t_n = t_{n-1} + d_n$. Here, we will work with timestamps, but the definitions work for both timestamps and durations.

In practice, the timestamps are expressed in some time unit, typically MIDI ticks or seconds, and we search for numbers $a > 0$ such that t_n fits into the a-grid $\forall n \in \{1, ..., N\}$. In musical terms, a is the duration of a certain note value that functions as the tatum of the rhythmic sequence and the numbers m_n such that

$$t_n - t_{n-1} = d_n = am_n$$

are the multipliers of a to produce the d_n's.

2.2 Definition of approximate common divisors

In practice, unless algorithmically generated, durations are rarely if ever exact multiples of a non-trivial divisor. We thus introduce some flexibility into the common divisor finding problem through a threshold $\tau > 0$.

We now relax the notion of fitting the a-grid by requiring timestamps to be within the threshold τ of the a-grid, *i.e.*:

$$\epsilon(t_n, a\mathbb{Z}) \triangleq \min_{m \in \mathbb{Z}} |t_n - am| \leq \tau\,, \tag{4}$$

$\forall n \in \{0, ..., N\}$, where $\epsilon(t_n, a\mathbb{Z})$ is the closest distance between the timestamp t_n and the a-grid. Since we require all the timestamps to be within the threshold, τ, of the a-grid, Equation 4 thus implies that the maximum distance between the timestamp series T and the a-grid, $\epsilon_T(a)$, is also within the threshold τ, i.e.,

$$\epsilon_T(a) \triangleq \max_n \epsilon(t_n, a\mathbb{Z}) \leq \tau\,. \tag{5}$$

Figure 1 shows the fit between a timestamp series $T = (0, 0.98, 1.52)$, given in seconds (s), and the 0.5 s-grid with a threshold of 0.05 s. We will use the threshold $\tau = 0.05$ s for all remaining examples.

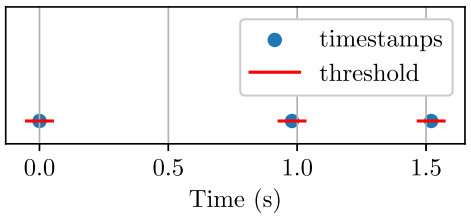


Figure 1. The fit between timestamp series $T = (0, 0.98, 1.52)$ s and a 0.5 s-grid for threshold $\tau = 0.05$ s.

With these definitions in place, we now formally define ACDs for a timestamp series T.

Definition 1 *Let $T = (t_0, t_1, ..., t_N) \in \mathbb{R}^{N+1}$ be a timestamp series. $a > 0$ is an approximate common divisor (ACD) of T with threshold $\tau > 0$ if*

1. *$\epsilon_T(a) \leq \tau$; and,*
2. *ϵ_T has a local minimum at a.*

The first condition ensures that the ACD satisfies the threshold requirement for the timestamp series T. The second condition makes the set of ACDs discrete by selecting the ACD that minimizes the error within each interval $\{a \in (0, \infty) : \epsilon_T(a) \leq \tau\}$.

In practice, since $\frac{a}{2}$ is an upper-bound for $\epsilon_T(a)$, we will see an increasing number of ACDs as a approaches 0, most of which are irrelevant since they are too small. This is why we will choose a lower bound for a, say 0.2 s.

Given a timestamp series $T = (0, 0.98, 1.52)$ s, we plot the function ϵ_T for the range $a \in [0.2, 1.0]$ as shown in Figure 2. A horizontal line marks the threshold $\tau = 0.05$ s and dots the ACDs. We compute $\epsilon_T(a)$ at steps of 1 ms. Since the definition of the ACDs involves local minima, the resolution of the computations can influence the results; in this paper, the precision is set to the millisecond.

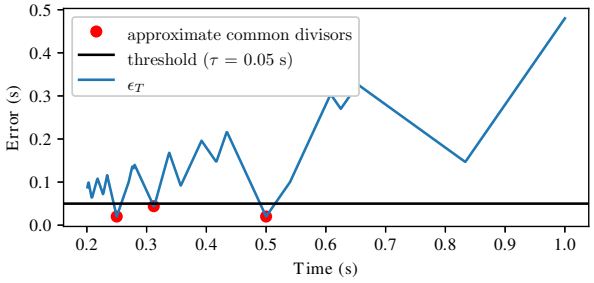


Figure 2. Approximate common divisors of the timestamp series $T = (0, 0.98, 1.52)$ s within the range $a \in [0.2, 1.0]$.

The computational complexity of the ACDs is linear with respect to the length of the timestamp series and the number of candidate values for a. The code uses Python libraries NumPy [8] and SciPy [9] and is available online [1].

2.3 Vectors associated with an ACD

The point of a number a being an ACD of a timestamp series $T \in \mathbb{R}^{N+1}$ with threshold τ is that there is an associated vector

$$M = (M[0], ..., M[N]) \in \mathbb{Z}^{N+1} \tag{6}$$

where $M[n] = \arg\min_{m \in \mathbb{Z}} |t_n - am|$, that consists of the integer values of the grid that fit the timestamps. Setting $\tilde{T} = aM \in \mathbb{R}^{N+1}$, we have

$$||\tilde{T} - T||_\infty = \epsilon \leq \tau\,, \tag{7}$$

where $\epsilon = \epsilon_T(a)$ will be called the approximation error. We can think of $\tilde{T}$ as the approximated timestamp series, which will never be more than ϵ off from the original.

Moreover, we can deduce the *integer durations*

$$\Delta = (M[n] - M[n-1])_{n=1}^{N} \in \mathbb{Z}^N \tag{8}$$

[1] https://github.com/Manza12/TENOR-2022

that satisfy the inequality

$$||\tilde{D} - D||_\infty \leq 2\tau\,, \tag{9}$$

where $\tilde{D} = a\Delta \in \mathbb{R}^N$ are the approximated durations.

In the example showed in the Figure 2 we get

$$\begin{array}{lll} a_1 = 0.5\,\mathrm{s} & a_2 = 0.312\,\mathrm{s} & a_3 = 0.25\,\mathrm{s} \\ \epsilon_1 = 0.02\,\mathrm{s} & \epsilon_2 = 0.044\,\mathrm{s} & \epsilon_3 = 0.02\,\mathrm{s} \end{array}$$

with a_i being the ACDs and ϵ_i being the corresponding errors, $\epsilon_T(a_i)$. We also retrieve the integer vectors

$$\begin{array}{lll} M_1 = (0,2,3) & M_2 = (0,3,5) & M_3 = (0,4,6) \\ \Delta_1 = (2,1) & \Delta_2 = (3,2) & \Delta_3 = (4,2)\,. \end{array}$$

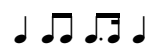
The next section will show how to use ACDs and the resulting vectors to transcribe monophonic rhythms.

3. MONOPHONIC TRANSCRIPTION

Let us first pose the problem of monophonic transcription in a way adapted to the formalism proposed for ACDs: when we have a monophonic rhythm, for example one produced by an instrument playing one note at a time, we can assume for simplicity that the release of each note occurs at the same time as the onset of the following note.

Moreover, since two onsets cannot occur at the same time, integer durations should not be 0. This means that the ACDs should be small enough to prevent this from occurring, say by requiring that the minimum candidate ACD be smaller than the smallest duration we wish to transcribe.

Let us consider, for instance, the rhythm



played at a tempo of ♩ = 60. The exact timestamp series and durations will thus be

$$T = (0, 1, 1.5, 2, 2.75, 3, 4)\,\mathrm{s} \tag{10}$$

$$D = (1, 0.5, 0.5, 0.75, 0.25, 1)\,\mathrm{s}\,, \tag{11}$$

where we consider the onsets of all notes and the release of the last note.

If humans were to play this rhythm, they would deviate slightly from these timestamps. Throughout this section, we consider a human realisation of this rhythm given by

$$T = (0, 1.018, 1.531, 2.061, 2.888, 3.179, 4.286)\,\mathrm{s} \tag{12}$$

$$D = (0, 1.018, 0.513, 0.53, 0.827, 0.291, 1.107)\,\mathrm{s}\,. \tag{13}$$

Let us show how one can transcribe this timestamp series into note values using ACDs.

3.1 Frames of a timestamp series

As mentioned, the ACD definition can result in many very small ACDs, which are not very interesting for music transcription because they imply too fine of a time resolution and make the note values too large. We therefore only consider ACDs above a lower bound of $0.2\,\mathrm{s}$ in the example.

Putting a lower bound on ACDs will imply that, if the threshold is small enough, some timestamp series may not have any ACD. In the musical context, this can be thought of as a player that deviates from a metronome, the metronome playing the role of the grid with the ACD being the time interval between two beats [2]. But this can be overcame by splitting the timestamp series into smaller blocks.

Given a timestamp series $T \in \mathbb{R}^{N+1}$ and a frame length $L \in \mathbb{N}$, $\forall n \in \{0, ..., N - L + 1\}$, the frame F_n of length L given by the vector

$$F_n = (t_n, t_{n+1}, ..., t_{n+L-1}) \in \mathbb{R}^L\,. \tag{14}$$

We now focus on frames of length 3 and on finding their ACDs. It is important to note that we need to shift our frame so that one of its timestamps is 0 in order to adapt the ACDs to that frame. We will then have

$$F_n = (t_n - t_i, t_{n+1} - t_i, ..., t_{n+L-1} - t_i) \in \mathbb{R}^L\,, \tag{15}$$

where $i \in \{n, ..., n + L - 1\}$ is the centering index.

For instance, if we take the first frame of the T defined in (12), $F_0 = (0, 1.018, 1.531)$, we have the ACDs and the integer durations

$$\begin{array}{lll} a_0^0 = 0.51\,\mathrm{s} & a_1^0 = 0.255\,\mathrm{s} & a_2^0 = 0.212\,\mathrm{s} \\ \Delta_0^0 = (2,1) & \Delta_1^0 = (4,2) & \Delta_2^0 = (5,2)\,, \end{array}$$

which give us three ways of transcribing the first two note values that are

$$R_0^0 = \text{♩}\,\text{♪}\qquad R_1^0 = \text{𝅗𝅥}\,\text{♩}\qquad R_2^0 = \text{𝅗𝅥}\,\text{♪}\,\text{♩},$$

where the unit is ♪.

3.2 Consistency across frames

Using the approach of frame-wise transcription, we need to impose some consistency across frames. Because of the way we have defined the frames, consecutive frames have two overlapping timestamps so one duration is shared between the two frames. For a coherent transcription, the common duration should be the same.

For instance, if we take the shifted second frame of T, $F_1 = (1.018, 1.531, 2.061) - 1.018 = (0, 0.513, 1.043)$, we have the ACDs and the integer durations

$$\begin{array}{ll} a_0^1 = 0.519\,\mathrm{s} & a_1^1 = 0.259\,\mathrm{s} \\ \Delta_0^1 = (1,1) & \Delta_1^1 = (2,2)\,. \end{array}$$

Since the frames F_0 and F_1 share the second and first duration, respectively, they should satisfy the consistency condition expressed as the equation

$$\Delta_{k_0}^0[1] = \Delta_{k_1}^1[0] \tag{16}$$

for them to be consistent. This equation will be satisfied by certain pairs (k_0, k_1) but not by others. This consistency requirement motivates the construction of the graph that is presented in the next section.

[2] Note that a metronome usually plays the tactus rather than the tatum; here, the metronome analogy should be thought of in the sense of the tatum.

3.3 The ACD graph

For each frame, we model each ACD as a node. Then, we add an edge from an ACD of a frame to an ACD of the next frame if the integer durations are consistent. If we do this for the timestamp sequence from (12), we obtain the graph shown in Figure 3.

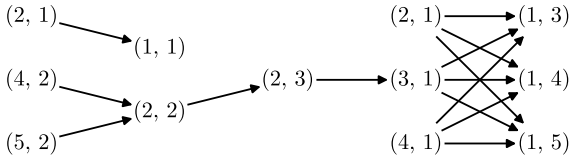


Figure 3. Graph of integer durations Δ.

Figure 3 shows how the consistency condition is fulfilled. It shows also how we can choose a transcription by selecting a path in the graph. Suppose we choose the path

$$(4,2) \to (2,2) \to (2,3) \to (3,1) \to (1,4)\,, \qquad (17)$$

then by merging the common integer durations, we get the rhythm $(4, 2, 2, 3, 1, 4)$. If we set the unit to be 𝅘𝅥𝅯, we will recover the rhythm ♩ ♫ ♬ ♩.

We can get different transcriptions by selecting different paths, which allows transcription to be framed as an interactive task. However, we may be interested in automatic transcription and, in this sense, we may assign weights to the edges in order to have a notion of the *shortest path* and the ensuing notion of the *best transcription*. In the following section we will propose a way of assigning weights to the edges based on the notion of *tempo variation*.

3.4 Assigning weights to the edges

In the first instance, we may choose to assign weights to the edges by weighting them by some function of the error ϵ. This will imply that better transcriptions are the ones that have ACDs that more closely fit the timestamps. Since the error is associated to the nodes instead of the edges, and path finding problems typically have weights assigned to edges rather than nodes, we must decide if the error is associated with the incoming or outgoing edge.

Whereas this is a valid approach, we propose another way of weighting the edges based on tempo variation. To illustrate this concept, we will use Figure 4.

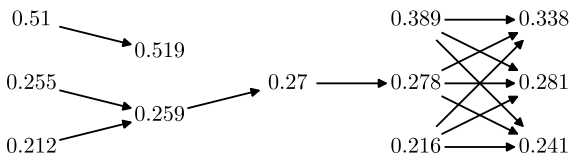


Figure 4. Graph of ACDs.

When we go from a node to another via an edge, we are changing the duration of the ACD and in so doing, we are varying the tempo. For example, if we go from $0.255\,\mathrm{s}$ to $0.259\,\mathrm{s}$ we are decreasing the tempo from 𝅘𝅥𝅯 = 235 to 𝅘𝅥𝅯 = 232, thus reducing the speed by $2\,\%$. This is not a big change, whereas if we consider instead the transition from $0.389\,\mathrm{s}$ to $0.241\,\mathrm{s}$, the tempo increment is $61\,\%$.

The tempo variation between two ACDs, measured in percent, may be defined as

$$\delta(a_1, a_2) = 100\left(\frac{a_1}{a_2} - 1\right) \qquad (18)$$

and we say that the speed from a_1 to a_2 has increased by $\delta(a_1, a_2)\%$ if $\delta(a_1, a_2) > 0$ and decreased by $\delta(a_1, a_2)\%$ if $\delta(a_1, a_2) < 0$. Notice that we divided a_1 by a_2 because we consider speed rather than duration; the two are inversely proportional.

We can then set the weight associated with an edge as a function of $\delta(a_1, a_2)$. In this case, we choose the logarithmic distance between a_1 and a_2, which is defined by

$$d_{\log}(a_1, a_2) \triangleq \left|\log_2\left(\frac{a_1}{a_2}\right)\right| = \left|\log_2\left(\frac{a_2}{a_1}\right)\right|. \qquad (19)$$

This has the property that it is symmetric and returns a value of one for a ratio of $2:1$.

Figure 5 shows the ACD graph of T weighted by the logarithmic distance.

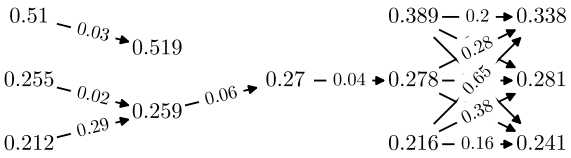


Figure 5. Weighted ACD graph

3.5 Shortest path problem

As we have a weighted graph, we can consider the shortest path problem and its corresponding transcription, the best one according to the parameters. The shortest path problem is defined by giving two nodes of the graph and finding the path that links them and has minimal weight. However, in our case, we may have multiple nodes for the first and last frames which makes ambiguous which nodes to choose. This is easily solved by adding an artificial source at the beginning that connects to all the ACDs of the first frame and an artificial sink connected to all the ACDs of the final frame. We may associate the weight 0 to these edges and then we will have a well posed shortest path problem. This is illustrated in Figure 6.

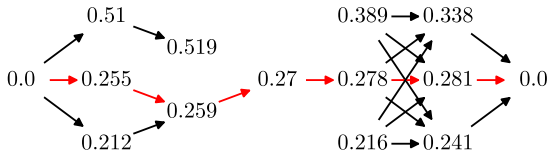


Figure 6. Shortest path outlined in red.

Since the graph is directed and acyclic, the shortest path can be readily computed; indeed, using the topological sorting of a directed acyclic graph, we have an algorithm

in linear time [10]. We can see that we recover the path from (17) that gives the rhythm ♩ ♫ ♬ ♩.

As pointed out, the shortest path depends directly on the weights, and their selection highly influences the outcome. We have presented weights based on the tempo variation or on the error ϵ, but there are many other approaches and combinations thereof, like selecting weights to avoid complex durations that are inelegant to write down, like 5, 11, 13... Conversely, if we value precision above notation clarity or physical reproducibility, we can assign a large weight to the error and allow very small ACDs. The trade-off between precision and clarity then arises as a parametrization problem that could be tuned via statistics or machine learning given labeled data. The subject of human intervention will be covered in Section 5.

3.6 The influence of the frame length

Up to this point, we have been working with frames of length 3. This approach makes sense because we always have a common duration that constraints the paths to be consistent from one frame to the next. However, we may choose a different frame size, for instance if we have very fast rhythms. In this case, consistency will be required of all the integer durations that are common to consecutive frames.

If we have two consecutive frames F_n and F_{n+1} of length $L \in \mathbb{N}$, the condition for the integer durations will be

$$\Delta_n[1 : L-1] = \Delta_{n+1}[0 : L-2]\,. \tag{20}$$

Here, indexing follows the Python convention where the second index is excluded.

Longer frames enforce the restrictions on ACDs and reduce their numbers. For instance, if we set the frame length to $L = 4$ for our previous example, we will have the graph shown in Figure 7, where the virtual initial and final integer durations are set to the null list.

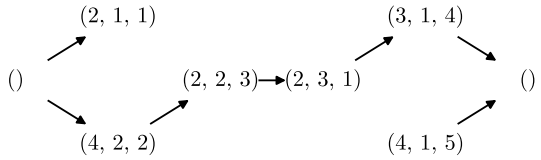


Figure 7. Graph of T with frame length $L = 4$.

We see in this case that there is a single valid path that corresponds to the integer durations we had in (17).

4. POLYPHONIC TRANSCRIPTION

Up to now, we have only considered monophonic rhythms, *i.e.*, rhythms formed by no more than one note at a time. This is useful in the case of singing voice, string instruments (in some cases), winds, etc. but it is insufficient in general and, for instance, for the piano.

When there are several voices playing at the same time, we must adapt our method so as to have meaningful results. First, we can no longer conceive of a frame as being a subvector of the time series T of fixed size L; indeed, we may have L timestamps occurring at almost the same time and then the integer durations would all be zero.

This leads us to an even deeper question: which timestamps should we consider, only onsets or also offsets? This is a very delicate question in music writing since it addresses directly the question of rests and articulation; indeed, since we no longer consider that the offset occurs at the subsequent onset, we have not only to quantize the onset but also the offset.

In order to simplify things, and acknowledging that our approach to the problem will be incomplete, we will focus on transcribing only the onsets. Then, we can account for the offsets in some sense, for instance rounding them toward the closest element in the grid. We will not tackle this problem, leaving it to future work, since it is affected by numerous factors such as performance, pedal, clean transcription and articulation.

That being said, let us consider the excerpt showed in Figure 8 from a Mozart sonata.

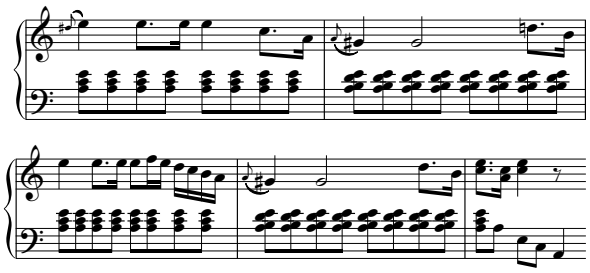


Figure 8. Start of the Mozart's Piano Sonata No. 8 in A minor, K. 310 / 300d.

We shall next attempt to transcribe a human performance of this excerpt, shown as a piano roll in Figure 9, by adapting the technique developed in the previous section.

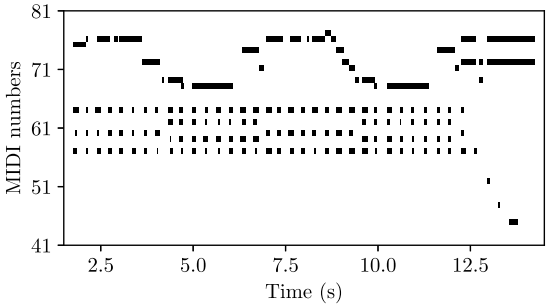


Figure 9. Piano roll of a human performance of the excerpt from Figure 8.

4.1 Polyphonic frames

For polyphonic transcription, we must first define a notion of a frame that will be adapted for polyphonic rhythm transcription. Consider, as previously, a timestamp series $T = (t_0, t_1, ..., t_N) \in \mathbb{R}^N$. As several timestamps can now be separated by a few milliseconds, like in the case

of chords, we will set the duration of the frame to $L > 0$ in seconds. Then, we will define the frame F_t starting at $t \in \mathbb{R}$ of length L as

$$F_t = (t_n \in T : t \leq t_n < t + L)\,. \tag{21}$$

In this way, each frame contains the timestamps of T that are within the time interval $[t, t + L)$. The duration L may be tailored to each piece, or even section or segment, and may be pre-computed. It may even be time-varying according to the speed of the piece.

In order to produce a consistent transcription, we shall overlap frames such that they measure common durations. This will be done by selecting overlapping frames with a hop size of $H > 0$. In our case, we choose

$$L = 1.5\,\text{s} \qquad H = \frac{L}{2} = 0.75\,\text{s}\,.$$

In this way, we arrive at a family of frames

$$\{F_{mH} \subseteq T : m \in \mathbb{N}\}\,. \tag{22}$$

Even though the family is formally infinite, as time progresses, the frames will eventually be empty when the input ends, so we can consider only the subfamily of non-empty frames.

4.2 Transcribing frames

Now that we have frames, we can apply the same procedure of extracting the ACDs of the frame as in the monophonic case. By default, we will shift the frame by its first timestamp, that is $F' = F - F[0]$. However, as mentioned before, we can shift the frame on any of its timestamps, for instance the middle one, leading to potentially different results. This approach may give negative numbers in the vector M, but is completely valid from the perspective of the ACD computation, which is defined on $\mathbb{Z}$, and from the transcription paradigm.

As previously, we will recover the vectors $M \in \mathbb{Z}^{N_t}$ and $\Delta \in \mathbb{Z}^{N_t - 1}$ whose size vary for different $t = mH$. It is important to emphasise that now there may be several durations in Δ that are 0, which points to the fact that some timestamps are concurrent. We should abandon the concept of Δ being the duration of the notes since we are only considering onsets; rather, we will think of Δ as a consistency vector that will be used to check if two overlapping frames are consistent.

Figure 10 shows MIDI onsets fitting into a $0.218\,\text{s}$-grid shifted by the first timestamp of the frame.

4.3 The ACD graph

As before, we will recover the ACDs of each frame to build a graph with ACDs as nodes, linking the nodes if they are consistent. Depending on the hop size, we may have more than two overlapping frames, but then we will only connect ACDs between consecutive frames.

The consistency condition between frames will now involve timestamps that are common to consecutive frames.

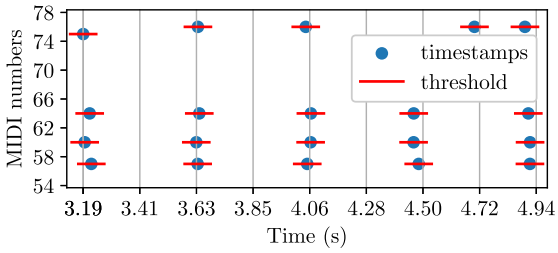


Figure 10. Timestamps of MIDI onsets fitting into a $0.218\,\text{s}$-grid.

For two frames F_{mH} and $F_{(m+1)H}$, the consistency condition will be:

$$\Delta_{mH}[n_i - n_m] = \Delta_{(m+1)H}[n_i - n_{m+1}] \tag{23}$$

$\forall n_i \in \mathbb{N}$ such that $t_{n_i}, t_{n_i+1} \in F_{mH} \cap F_{(m+1)H}$, where $n_m \in \mathbb{N}$ and $n_{m+1} \in \mathbb{N}$ are the first index belonging to F_{mH} and $F_{(m+1)H}$ respectively.

Now that we have established the consistency condition, we can plot the ACD graph of the onsets of the excerpt presented in Figure 9. We added a source node and a sink node to complete the path and, as seen in the Figure 11, there is only one possible path linking them.

Of course, this will not always be the case. The single path solution was a consequence of the fact that the player followed the rhythm very strictly, in part due to the genre of the music. If we take the vector M built by following this path, we will recover the onsets expected by the score.

We will not repeat all the considerations outlined in the previous section regarding path weighting and automating of the transcription model. Rather, in the next section, we will show how transcribing can be done interactively.

5. INTERACTIVE PATH SELECTION

We have modeled both monophonic and polyphonic transcription as path selection problems. Solving either of these problems can be done automatically if we assign weights to the path, but it can be interesting to select the path interactively by means of a user interface. This transforms the rhythm transcription problem into a multiple output problem, and gives humans the responsibility of selecting their preferred transcription from among a set of possibilities.

Figure 12 shows a prospective user interface for selecting the path. This interface has not been implemented; it presents only a possible graphical layout of the essential functions. It is intended, however, to be implemented in a future version of OpenMusic [11, 12].

Let us go through an overview of the interface from top to bottom, then left to right.

5.1 The piano roll panel

In this panel, the user may see the actual MIDI file and navigate the frames with the arrows. S/he may change the size of the frame and its position by dragging and dropping

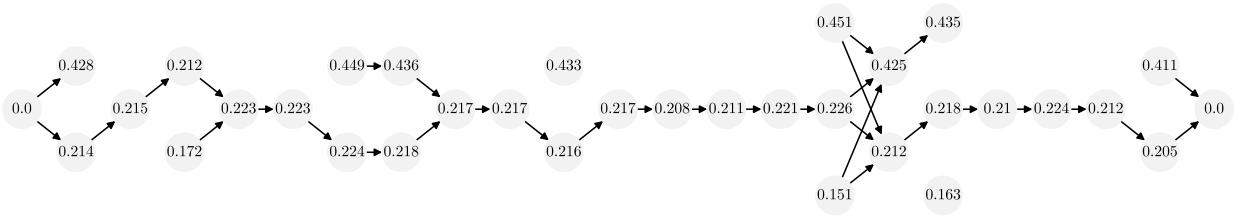


Figure 11. Graph of the excerpt from the onsets of Figure 9.

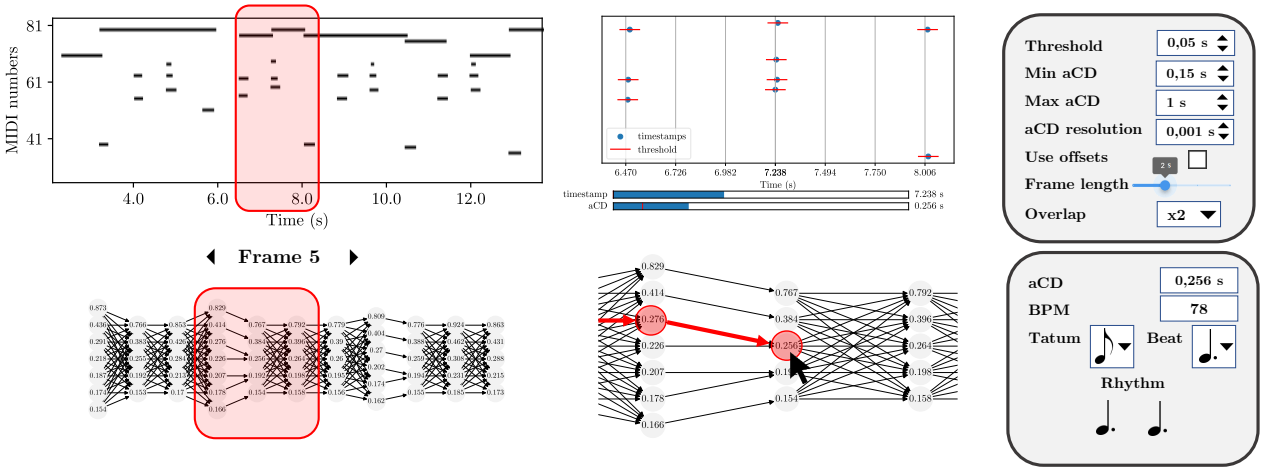


Figure 12. Design of a prospective graphic user interface for interactive path selection.

the limits of the frame. Ideally, the user should also be able to edit the MIDI file by selecting the notes and stretching or compressing them; this way, s/he could influence the transcription by editing the timestamps.

5.2 The full graph panel

This panel shows the full ACD graph and will be related to the frame selected on the piano roll panel. Its purpose is to keep track of the path selected and to allow the user to navigate through frames so that s/he may have a global view.

5.3 The grid panel

This panel presents information from the part of the piano roll where the frame is located. However, instead of MIDI bars, the onsets (and potentially the offsets) will be shown as timestamps so that we may know exactly where each fits in the grid. Timestamps about to be transcribed are plotted in blue and the threshold shown in red. The representation helps the user understanding the scale and the degree of freedom. With the two sliders in the bottom, we can tune which timestamp will be the shift parameter and the resulting grid if we change the ACD.

5.4 The focused graph

This sub-graph of the full ACD graph serves shows the ACD selected for each frame. With the cursor, we may select an ACD. The red line connects the current to the previous ACD. Options in the previous and next frames are shown to convey the available paths when choosing an ACD.

This will be one of the main panels on which to act. Indeed, changing the ACD will trigger updates on other panels, and it will be by selecting the ACDs one by one that the transcription will be made. Nevertheless, we can initialise the graph with some optimal or near optimal path, but the interactive part of the process will be to adapt the graph by changing the ACD.

5.5 The parameters panel

The parameters panel will let the user select all the parameters needed for the ACDs computation:

1. In order to select the timestamps, the user should choose the frame duration with a slider. Also, s/he can select the overlap via a drop-down list. We may also use a checkbox to determine if the offsets are used in the computations. If that is the case, they will also appear in the grid panel.

2. Once the timestamps are selected, the parameters for the computation of the ACD shall be tuned with spinners; these parameters are

   (a) the error threshold,

   (b) the lower bound,

   (c) the upper bound, and

(d) the computation resolution.

These parameters will play an important role in which ACDs are recovered.

5.6 The rhythm panel

In this panel, users can see the outcome of their choice; by selecting an ACD in the focused graph, both the ACD and the BPM (beats per minute) will be printed in non editable text boxes. The ACD corresponds to the tatum, which can be selected in a drop-down list. The beat, that will usually consist of several tatums, can also be freely chosen in another drop-down list.

In addition, when the ACD is selected, the timestamps will be transcribed into rhythms via a label in the panel. It should be noted that this rhythm corresponds to onsets and should be thought of as a musical grid rather than the durations of the notes, at least when offsets are not taken into account.

6. CONCLUSIONS AND FUTURE WORK

In this work, we have shown how an extension of the notion of common divisors to a continuous framework leads valuable contributions to rhythm transcription. Their linear computational complexity makes them a suitable and efficient tool for quantizing musical rhythms for large amounts of data.

We have also proposed a flexible framework where we considered transcription by splitting the timestamp series into frames and computing ACDs separately. However, when this is done, adjacent frames need to be consistent, and we have chosen to model this via a graph in which ACDs are nodes and edges represent the consistency between frames.

Once this graph is set up, the transcription problem then turns into a one of path selection. Using this paradigm, we proposed two complementary ways of solving the transcription problem: by assigning weights and determining the shortest path, or by allowing humans to intervene in the process by selecting a path that results in a more desirable transcription.

Regarding this last option, we presented a prospective graphical user interface and gave an exposition of its likely elements. This interface will allow the user to directly steer the transcription and may be implemented in musically-oriented frameworks.

In conclusion, we have proposed a framework that is simple in essence but highly parameterizable. Indeed, there are many ways in which we may affect either the ACD computation or the graph creation and weighting, for example by tuning the weights, limiting the tempo variation or by allowing only certain integer durations.

A remaining challenge is the organisation of the durations into groups and measures. By considering the tatum instead of the tactus, we may enter into a low-level quantization that does not account for how the durations are grouped together. The grouping of durations to form rhythm trees has been studied by [2] and is of major importance in OpenMusic [12].

A way to tackle this may be to build a rhythm tree on top of the ACD graph. How this may be done is outside the scope of this paper but several possibilities can be explored, like choosing different duration groupings and evaluating if they would form a tactus. A challenge for both these approaches is the handling of ties, which could also be broken through the user interface.

In the future, the tool we proposed could be incorporated into existing frameworks like OpenMusic either as a function that performs transcription automatically or as a user interface that allows composers and music editors to interact with the transcription. We would then be able to adapt criteria for weighting the graph to tune the standalone part of the algorithm. To that end, we may use several techniques, but one interesting option would be to record data from musicians transcribing rhythms in order to tap into their experience so as to make the output of the algorithm as human friendly and readable as possible.

Acknowledgments

This result is part of the COSMOS project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 788960.).

7. REFERENCES

[1] C. Agón, G. Assayag, J. Fineberg, and C. Rueda, "Kant: a Critique of Pure Quantification," in *In Proceedings of the International Computer Music Conference (ICMC)*, Aarhus, Denmark, 1994, pp. 52–59.

[2] A. Ycart, F. Jacquemard, J. Bresson, and S. Staworko, "A Supervised Approach for Rhythm Transcription Based on Tree Series Enumeration," in *In Proceedings of the International Computer Music Conference (ICMC)*, Utrecht, Netherlands, 2016, pp. 369–376.

[3] A. T. Cemgil and B. Kappen, "Monte Carlo methods for tempo tracking and rhythm quantization," *Journal of Artificial Intelligence Research*, vol. 18, no. 1, pp. 45–81, 2003.

[4] J. C. Brown, "Determination of the meter of musical scores by autocorrelation," *The Journal of the Acoustical Society of America*, vol. 94, no. 4, pp. 1953–1957, 1993.

[5] N. Howgrave-Graham, "Approximate Integer Common Divisors," in *Cryptography and Lattices*, ser. Lecture Notes in Computer Science, J. H. Silverman, Ed. Berlin, Heidelberg: Springer, 2001, pp. 51–66.

[6] J. Seppänen, "Computational Models of Musical Meter Recognition," Master's thesis, Tampere University, Tampere, 2001.

[7] G. Romero-García, "Rhythm Transcription and Characterization for Performed Music and Arrhythmia Sequences," Master's thesis, Sorbonne Université, Paris, 2020.

[8] C. R. Harris, K. J. Millman, S. J. van der Walt, and others, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[9] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, and J. Bright, "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nature methods*, vol. 17, no. 3, pp. 261–272, 2020.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Section 22.4: Topological sort," in *Introduction to Algorithms*, 2nd ed. Cambridge, Mass: MIT Press, 2001, pp. 549–552.

[11] C. Agón, "OpenMusic : un langage visuel pour la composition musicale assistée par ordinateur," PhD Thesis, Paris 6, Paris, 1998.

[12] J. Bresson, C. Agon, and G. Assayag, "OpenMusic: visual programming environment for music composition, analysis and research," in *Proceedings of the 19th ACM international conference on Multimedia (MM '11)*, New York, 2011, pp. 743–746.