Intuitive and efficient computer aided music rearrangement with optimised processing of audio transitions

Daniel Stoller*†[1], Igor Vatolkin‡[2], and Heinrich Müller§[2]

[1]Queen Mary University, London, United Kingdom
[2]Technical University Dortmund, Dortmund, Germany

April 25, 2018

Abstract

A promising approach to create new versions of existing music pieces automatically is to cut out and rearrange sections so that transitions are minimally perceptible and constraints regarding duration or structure are fulfilled. We evaluate previous work and improve on its limitations, particularly the disregard for loudness changes at cuts and the unintuitive control over the musical structure of the output. Our software provides a user-friendly interface, which we make more responsive by greatly accelerating the search for an optimal output track using the A* algorithm. Listening experiments demonstrate an improvement in perceived audio quality.

Keywords. Composition, Machine Composition, Music Analysis, Sound Synthesis

1 Introduction

The dawn of the digital era rapidly increased the accessibility of music and also made sharing music easier. Additionally, it allowed for computer-assisted editing of music, which is more convenient and efficient. In this work, we will focus on the task of music rearrangement as enabled by such digital music editing tools. One context of music rearrangement is the creation of remixes based on existing music pieces that alter or reinterpret their musical ideas. Furthermore, instrumental versions of songs can be created by cutting out the vocal sections. The adaptation of music pieces to video sequences is also

*d.stoller@qmul.ac.uk
†Work undertaken at the Technical University Dortmund.
‡igor.vatolkin@cs.tu-dortmund.de
§mueller@ls7.cs.uni-dortmund.de

an important application, because films, video games, and advertisements often require music accompaniment of a given length and musical structure that supports the visual content. However, music rearrangement is often performed manually by an expert using professional audio software. It is often very time-intensive to achieve high quality results, partly because manipulation occurs at the signal level, where musical abstractions such as melody, instrumentation, and rhythm are not available to guide the process. Automatic solutions for these tasks would not only save time, but also allow less experienced users to perform them.

In this paper, we adopt the concept of rearranging music pieces based on **musical structure segmentation (MSS)** from previous work (Wenner, Bazin, Sorkine-Hornung, Kim, & Gross, 2013). For the input music piece, automatic MSS produces a **song structure estimate** which can optionally be refined by the user to form the **input song structure**. Employing segmentation concepts from (Wenner et al., 2013), we define a **song structure** as a set of **segments** each described by their starting time and their associated **cluster**. Segments are grouped into different clusters according to their musical function or similarity. A more complex approach to defining musical structure such as (Kühn, 1987) could be used, but is beyond the scope of this article. Users can specify a desired song structure of the output piece (**target song structure**) based on the input song structure. The output track is assembled from different sections of these segments, so that jumps between sections are least perceptible to the listener and its **output song structure** aligns with the target song structure.

Examples of possible rearrangements are sketched in Figure 1. Figure 1 (a) describes the segments of the input song structure. Depending on the desired target length, rearranged pieces may be either shorter (b), (c) or longer (d), (e). While (b) and (d) do not adhere to any target song structure and focus only on least perceptible jumps, in (c) and (e) the target song structure was set to a scaled version of the original in (a) so that the global musical structure of the original is preserved.

Based on this concept we have designed and implemented a system for rearranging music incorporating the following main novel contributions:

- an intuitive and effective application of tolerance windows when enforcing musical structure that allows the user to make a trade-off between audio quality and adherence to the desired musical structure,
- an efficient approach to calculating the rearrangement based on the A* algorithm to find the optimal cuts, which typically needs significantly less time than the dynamic programming applied in previous work (Wenner et al., 2013), thus allowing for a more interactive user experience,
- avoiding and smoothing loudness changes at jumps, which results in music pieces of higher quality as demonstrated by extensive listening experiments using a range of music genres,
- a user interface which allows changing the duration and intuitively reordering sections of a music piece in an easy-to-use manner, enabling everyone without prior experience to quickly create new music pieces.

Section 2 provides an overview of the previous methods to music rearrangement and a thorough evaluation of (Wenner et al., 2013) with a discussion of the drawbacks of recent approaches. Section 3 outlines the functionality and the architecture of our music arrangement system. Section 4 specifies the desired rearrangement as an optimisation problem and proposes a solution based on the A* algorithm. Section 5 concerns the improvement of jumps by post-processing of the audio signal. Section 6 compiles the findings of an extensive evaluation. Finally, Section 7 draws conclusions and discusses future work.

2 Related work

In the following, we provide an overview of existing music rearrangement systems and a thorough evaluation of (Wenner et al., 2013) to identify the limitations of recent approaches.

2.1 Existing approaches

Methods applicable to music rearrangement differ in their basic properties and capabilities. Additionally, only few have been developed specifically for this task.

When only a change in duration of a music piece is required, one solution is simply adjusting the playback rate, which simultaneously causes an undesired change in pitch. *Timescale-pitch modifications* like *WSOLA* (Verhelst & Roelands, 1993) can be employed to change tempo and pitch independently, so that the scaling of duration can be performed without changing the pitch. Although yielding good results for small duration changes, higher scaling factors produce sound artefacts and drastically change the listening experience.

Synthesis-based methods (Parker & Behm, 2004), and specifically corpus-based concatenative synthesis (Schwarz, Cahen, & Britton, 2008; Einbond, Trapani, Agostini, Ghisi, & Schwarz, 2014), present another approach. Here, music is created by combining individual sound samples from a sound database according to user criteria. The high granularity results in great control and creative freedom during composition. In our rearrangement scenario however, where we aim for a new but faithful adaption of an existing piece, fitting samples from the database would need to be found, and then properly reassembled. Due to the mismatch in sound quality between the database samples and the original components of the piece, and the complex synthesis that needs to recreate most of the original musical structure, the output would heavily differ in sound.

Consequently, research for music rearrangement in recent years focused on another approach where parts of the original piece are selected and concatenated by searching for sections with a similar hearing impression to minimise the perceptibility of cuts between the selected sections. In other words, the idea is to play back the original while occasionally "jumping" from one position in the piece to another.

One such retargeting approach was devised by (Wenger & Magnor, 2011), developed into a genetic algorithm (Wenger & Magnor, 2012) and later modified to reduce the search space for cut positions to whole bars as identified by a beat tracker (Tauscher, Wenger, & Magnor, 2013). While yielding good results for dance music with a strong, regular

beat, the approach might not generalise to genres with changes in tempo or loudness. Furthermore, the user is not assisted by automatic MSS to construct the desired input song structure, but instead has to create it bar by bar.

Such an automatic MSS is instead employed by (Wenner et al., 2013), whose work is similar, but not only targeted at dance music and also incorporates additional features such as removing vocals or creating infinite music. Because it uses **bixels** (a contraction of the words beat and pixel), i.e. the musical content between two consecutive beats, and not whole bars for self-similarity calculation, irritating jumps from one count of a bar to a different count can occur. A very similar technique is also used by (Sato, Hirai, Nakano, Goto, & Morishima, 2016) to adapt soundtracks to video clips so that their climaxes align.

With a user-defined target song structure, both approaches can suffer from a significantly reduced quality of cuts, as many potentially less perceptible cuts do not lead to a solution satisfying the song structure constraints.

2.2 Evaluation of previous work

Taking into account the considerations from Section 2.1, the two "jump-based" approaches (Wenger & Magnor, 2011; Wenner et al., 2013) appear as the most promising future direction. Although the method in (Wenger & Magnor, 2011) often provides high quality outputs for dance music, our goal is to build an all-purpose music rearrangement system without such a genre restriction. In contrast, the algorithm developed by (Wenner et al., 2013) has a similar scope and thus served as a starting point for our approach.

To identify its limitations, we reimplemented and intensively evaluated Wenner's method on our database "CC1" (see Appendix, Table 1) in a preliminary study. It contains 42 Creative Commons songs sourced from the "Free Music Archive" (WFMU, 2009), selected to cover a range of musical genres. Because the output quality of the method depends on the accuracy of the employed beat tracking system, ground truth data for the beat positions was created manually by "tapping" to the beat to determine the effect of beat accuracy on the system's output quality.

The results of this study have revealed the following list of problems with Wenner's method even when using the correct beat positions from the ground truth. Some of the problems will be tackled in this paper (indicated by references to the related sections).

Frequent repetition of short sections. When extending music, the algorithm sometimes generates pieces that repeat the same short excerpt of the original song several times in direct succession. Although this often produces imperceptible cuts, a lot of repetitions are detrimental to the listener experience. In Section 4.1.2, we will discuss how to penalise the jumps responsible for this effect.

Bixel-based song structure enforcement. Wenner's method features *structure-aware retargeting*, where the user can set a target song structure that the output track should have. It is enforced on a bixel-by-bixel basis: Each bixel in the output can be selected only from input bixels from a specified cluster. This results in pieces with song structures that match the target song structure with a high precision of up to a bixel's length.

However, cuts tend to sound significantly worse than without structure-aware retargeting, since the allowed outputs are severely restricted due to the limited choice of audio material available for output at each point in time. In Section 4.2.4, we propose a method to enforce a song structure with a user-defined level of tolerance, so that slight deviations of the output from the target song structure are allowed and also exploited, if they lead to a result of higher quality.

Inaccuracies caused by differently sized bixels. Pieces with changing tempo feature varying inter-beat intervals. This leads to problems with the bixel-based approach from Wenner in two ways. Firstly, the length of a music piece comprised of a specific number of bixels can vary depending on the bixel selection, making it inaccurate and unintuitive to use the number of bixels as a user-defined target length. We allow users to specify the desired duration in seconds instead using the approach presented in Section 4.2.2. Secondly, enforcing a change in segment clusters after a specified number of beats or bars can produce results with transition times that deviate from the user specification. We tackle this problem in Section 4.2.4 with a dynamic song structure enforcement based on absolute time.

Disregard of time signatures. The algorithm does not consider time signatures during jump selection, so that sometimes the number of beats in a bar is changed. This issue is especially noticeable for music genres with constant time signatures and a clear indication of the current position in the bar, such as Pop.

Cutting off vocals. Another issue noted in (Tauscher et al., 2013) and (Wenner et al., 2013) is jumping in the presence of vocal activity, which occurs often, can jumble up the lyrics and is often perceived as especially jarring as humans tend to focus on the human voice when listening. The original work by Wenner provided a work-around by manually selecting the vocal parts to not jump in these sections, but an automatic solution would be desirable.

Ignorance of melody. Melodies are often repeated throughout a music piece, creating expectations which are sometimes violated by Wenner's method, as the measure used for computing the similarity of two bixels is based on *Mel-Frequency Cepstral Coefficients* (MFCCs) describing timbral, but no melodic aspects, which could be added to alleviate this problem.

Other issues. Rarely, sudden changes occur in the spatial location of instruments in the stereo field. The analysis cannot account for these effects as it involves converting the input signal to mono as the first step. Another issue affects songs featuring large tempo changes, in which unpredictable and very noticeable jumps from slow to fast parts and vice versa are common and significantly degrade the output quality.

3 System overview

In the following, we will outline the functionality and the architecture of our music rearrangement system.

3.1 Functionality

The system allows the automatic construction of a target music piece from the original one with regard to user-defined constraints such as a desired duration. Based on an input song structure, a target song structure can be defined so that the resulting music piece follows the given musical structure. The output track is generated by determining and then assembling a sequence of bixels so that the given user-constraints are fulfilled. Not every original bixel has to appear in the output track, and one bixel may be used several times.

The system features an interactive user interface shown in Figure 2. In the first stage, the user is asked to create the input song structure for the original music piece and is assisted by a method for automatic MSS presented in Section 4.1.1. The original audio with the fixed input song structure is displayed at the top (1).

After confirming the current song structure, the user can edit the constraints to be imposed on the output track. The desired start and end position t_{start} and t_{end} can be set by dragging the green and red bar in the input display (1). Directly below, points in the black area (2) can be created and moved to model an importance curve that is aligned in time to the original audio. For every bixel, the importance curve influences the probability of including that bixel in the output. Further constraint parameters for the output can be set on the left hand side of the interface (7). Those parameters concern the duration with a tolerance, the influence of the importance curve, how much aspects of loudness and repetition should be considered, and how strongly loudness differences at cuts should be smoothed out by post-processing. This post-processing step can be toggled on and off, along with time-stretching of the final output to exactly fit the desired duration, and whether the target song structure should be imposed on the result. Tolerances when imposing the target song structure on the output can be set, allowing the segment boundaries to deviate from the target positions without or with less penalty. The tolerances enable the user to control the trade-off between the accuracy of the produced result regarding its duration and the quality of the discovered path. The target song structure (3) can be intuitively edited by adding, changing and removing segment boundaries.

A new solution based on these constraints can be produced by pushing the button "Generate Result" on the bottom left. The fourth horizontally aligned display (4) depicts the waveform of the generated music piece along with its output song structure, with white crosses indicating cuts.

The visualisation in the lower right (5) contains a white graph which plots the playback position in the new track on the horizontal axis against the position in the original song that is used in the result at this point. Finally, a list of measures to evaluate the output quality is displayed at the bottom (6), of which some will be introduced in Section 6.

3.2 System architecture

Our music rearrangement system can be represented as a pipeline comprised of individual processing stages. Its general structure is shown in Figure 3 and can be broadly separated into two phases. The **analysis** phase extracts relevant features from the input track, and the **synthesis** phase takes these features to generate a new music piece according to the user-defined parameters.

Following Wenner's approach, we use bixels as building blocks to form new music pieces by concatenating a carefully chosen series of bixels. Because bixels are defined as the musical content between two consecutive beats, our system begins by identifying the locations of the beats with a **beat tracking** system. Beat tracking as a problem in itself is not the focus of this work, so we used the freely available beat tracker by (Davies & Plumbley, 2007), but in principle any beat tracker can be integrated into the system. Alternatively, beat positions can be manually imported from an external file.

The bixels obtained from beat tracking are then analysed with regards to their timbre and loudness in the **preprocessing stage** described in Section 4.1. A method for automatic MSS outlined in Section 4.1.1 provides a basis for the user to create a structural annotation of the original piece.

After confirming the song structure, the synthesis phase begins with the **path optimisation** presented in Section 4.2, searching for the least costly path through the original piece. The cost function for a path is designed to estimate the perceived quality of the resulting cuts. Furthermore, the search is restricted to solutions fulfilling the user constraints such as target length and song structure.

To further increase the final quality of cuts, the pairs of cut positions calculated by the path optimisation are refined based on the local properties of the audio signal at these positions. At first, the **jump synchronisation** described in Section 5.1 aims to correct inaccuracies in the detected beat positions that cause sound artefacts and irregular beat intervals after cutting. To improve the loudness continuity at cuts, the **loudness equalisation** stage which is described in Section 5.2 appropriately amplifies or attenuates the signal at the cut points. Finally, the music piece is **assembled** from sections of the original piece according to the calculated jumps. To further remove any remaining sound artefacts, **crossfading** over a duration of $t_{\text{cross}} = 0.04$ s is applied to the resulting cuts. After assembly, the output track can optionally be exactly time-scaled to the target duration with a timescale-pitch modification method. We use the WSOLA algorithm (Verhelst & Roelands, 1993) for this purpose, but it can be easily replaced by an alternative method.

With the exception of the final assembly stage, the input signal is always processed as a single-channel signal, unless otherwise noted.

The system is implemented in C++ and MATLAB. C++ was selected due to its performance for quickly solving the optimisation problem in Section 4.2, while MATLAB enables fast prototyping using statistical and audio processing tools. The user interface is implemented using the cross-platform C++ framework **JUCE** (ROLI Ltd., 2004), as it provides audio input and output functionality. Additionally, the **dRowAudio** (Rowland, 2010) module is utilised for its waveform display and audio player functions.

4 Rearrangement calculation

This section will show how finding the desired rearrangement can be viewed as solving an optimisation problem that incorporates the user-defined constraints, and propose the A* algorithm for multiple goals for finding this solution. The preprocessing stage of the analysis phase and the path calculation stage of the synthesis phase will be presented, following the order of execution shown in Figure 3.

4.1 Preprocessing

The preprocessing stage extracts musical information about the input track and its bixels needed in the subsequent path optimisation stage. The first subsection is concerned with a method for automatically creating a song structure estimate of the input track. The following three subsections are devoted to developing a measure to estimate the perceptual quality of bixel transitions. The measure assigns a transition cost to every pair of bixels b_i, b_j estimating the transition quality in case b_j will be played directly after b_i in the result. These bixel transition costs will be used in Section 4.2 to determine the optimal sequence of bixels of the new track.

4.1.1 Automatic music structural segmentation

Our music rearrangement system allows users to control the musical structure of the output track. Based on an input song structure of the original piece, a target song structure can be set that constrains the usable bixels to those belonging to a particular cluster ((3) in Figure 2). To support the user in creating an input song structure, we employ a method for automatic MSS described in the following.

The method mostly follows (Wenner et al., 2013), in which a **novelty curve** is computed for the input track based on timbral aspects. A high dissimilarity between the previous and upcoming MFCC features together with a high homogenity within those time spans results in a high novelty rating. Thus, maxima in the novelty curve are selected as segment boundaries. The resulting segments are grouped into clusters according to their timbral similarity with the goal of identifying groups of segments serving the same musical function. *Spectral clustering by normalised cuts* (Shi & Malik, 2000) of the mean MFCC vectors of each segment is used to obtain C clusters. The primary drawback of this approach is that the number of clusters C has to be known in advance. Therefore, we adapt the approach as follows. We automatically find the optimal number of clusters within a user-specified range $[C_{\min}, C_{\max}]$, so that the user does not have to find the most suitable C manually. Clustering is performed for all $C \in [C_{\min}, C_{\max}]$, and the best solution is chosen as the one with the highest average **silhouette** value (Rousseeuw, 1987) serving as a simple evaluation metric. The motivation is that a cluster has a high silhouette value if, in the MFCC space, its bixels are close by and bixels from other clusters are far away. This allows us to avoid having to define a parametric model with a likelihood function for the shape of the clusters, as is neccessary for using information-theoretic measures such as the **Bayesian information criterion** (Celeux & Govaert, 1992).

The song structure estimate provided by the automatic MSS is designed to give the user a reasonable starting point for constructing the desired song structure in most cases. The segmentation accuracy of our method was not measured since segmentation is not

the main focus of this paper and errors in estimation can be corrected by the user. In addition, in some cases the estimate from the automatic MSS might not be useful when the user wants to define the song structure in very unusual ways to perform special rearrangement tasks, such as marking vocal and non-vocal segments for the creation of instrumental pieces. While our method is not guaranteed to recover sections such as verses and choruses consistently, it should give an initial guidance on the timbral development of the music piece and to allow for easier navigation. If segmentation accuracy is very important, a more sophisticated method can easily be integrated into our system due to its modularity.

4.1.2 Unified transition costs

The **bixel transition cost** from an **origin** b_i to a **destination** b_j are represented by entries $T_{i,j}$ in the $N \times N$ **unified cost matrix** $\mathbf{T}$, where N is the number of bixels of the input track. Low matrix entries indicate a high transition quality and vice versa. The transition costs take into account four features: timbre, loudness, importance, and length of backward jumps.

The **timbre transition costs** and the **loudness transition costs** are represented by $N \times N$ matrices $\mathbf{D}'$ and $\mathbf{L}$, respectively, which will be explained in detail in the next two subsections.

Importance was introduced by (Wenner et al., 2013). It is provided by the user (Figure 2 (2)) and is represented by a vector $\mathbf{I}$, where each entry $I_i \in [0, 1]$, $i \in \{1, \ldots, N\}$ influences the probability of including bixel b_i in the output.

The **length of backward jumps** is used to avoid highly repetitive results encountered during the evaluation of previous work (Wenner et al., 2013) in Section 2.2. Repetitions are caused during the path optimisation stage by backward bixel jumps with a short distance and a low cost, which are thus utilised very often in direct succession. To reduce the problem, a **jump penalty**

$$c_{\text{rep}}(\Delta_{\text{jump}}) = \frac{w_{\text{r}}}{\Delta_{\text{jump}}} \tag{1}$$

is introduced, where Δ_{jump} denotes the distance of a backward jump in number of bixels. The variable $w_{\text{r}} \geq 0$ is a user-defined value controlling the strength of repetition avoidance (Figure 2 (7)). The jump penalty has higher values for jumps with a short distance.

The timbre transition costs, the loudness transition costs, and importance are integrated by linearly weighting their individual terms into a single-objective function, yielding a combined cost matrix $\mathbf{T}'$ with entries

$$\begin{aligned} T'_{i,j} = w_{\text{d}} D'_{i,j} + w_{\text{l}} L_{i,j} + w_{\text{p}} I_i \qquad & \text{with} \\ w_{\text{d}} + w_{\text{l}} + w_{\text{p}} = 1 \qquad & \text{and} \\ w_{\text{d}}, w_{\text{l}}, w_{\text{p}} \in [0, 1]. & \end{aligned} \tag{2}$$

The weight variables w_{d}, w_{l}, w_{p} can be manipulated by the user to influence the balance between the criteria for bixel jump selection (Figure 2 (7)).

The jump penalty is integrated into the combined cost matrix by adding it to the

existing costs of backward bixel jumps, yielding the unified cost matrix

$$T_{i,j} = \begin{cases} T'_{i,j} & \text{if } i < j \\ \min\{T'_{i,j} + c_{\text{rep}}(i - j + 1), 1\} & \text{else.} \end{cases} \tag{3}$$

An example of the matrix $\mathbf{T}$ for $w_{\text{r}} = 4$ is depicted in Figure 4, exhibiting costs near one along the diagonal and below, decreasing in the lower left direction, which is where short backward jumps are represented.

The unified cost matrix $\mathbf{T}$ is used for the path optimisation process (Section 4.2.1), and is recomputed only when the weights change.

4.1.3 Transition costs regarding timbre

The matrix $\mathbf{D'}$ of timbre transition costs is based on a similarity matrix $\mathbf{S}$ from (Wenner et al., 2013). The elements of the similarity matrix $\mathbf{S}$ are pairwise Spearman correlations between 40-dimensional MFCC vectors of bixels. The temporally smoothed version $\mathbf{S'}$ is then used to build a matrix $\mathbf{D}$ with

$$D_{i,j} = 1 - S'_{i+1,j}. \tag{4}$$

However, the authors do not define how to compute the row $N + 1$ in $\mathbf{S'}$. We propose padding the borders of $\mathbf{S}$ with $m = 2$ zeros before performing the smoothing operation, leading to low values near the borders of $\mathbf{S'}$. We use the $N + 1$-st row of $\mathbf{S'}$ to compute the last row of our improved transition matrix $\mathbf{D'}$ to penalise jumping from the end of the piece. In addition, we define a matrix $\mathbf{S''}$ built with only valid entries of $\mathbf{S}$ in the temporal smoothing process, and use it to obtain all other rows in $\mathbf{D'}$. Overall, the timbre transition costs are given by

$$D'_{i,j} = \begin{cases} 1 - S'_{i+1+m,j+m} & \text{if } i = N \\ 1 - S''_{i+1,j} & \text{else,} \end{cases} \tag{5}$$

In contrast to (Wenner et al., 2013), we do not assign a cost of one to the diagonal elements representing small backward jumps, because the concept introduced in Section 4.1.2 is a more general and flexible approach for avoiding such repetitions.

4.1.4 Transition costs regarding loudness

The loudness, only represented in the form of the 0-th coefficient of all 40 MFCCs, has little influence on the timbre transition costs of the preceding subsection. As a result, many jumps with extreme loudness differences occur, as mentioned in Section 2.2. Therefore, we propose an additional cost matrix $\mathbf{L}$ specifically for loudness as a supplement to $\mathbf{D'}$.

Loudness modelling. In contrast to simply averaging energy for every bixel, a loudness model reflects more closely the human perception of sound intensity. It takes into account that the perceived loudness of a tone is dependent on its frequency (Olson, 1972), that loud sounds can mask quieter sounds occurring shortly after (Fletcher & Munson,

1937), and other effects. Two widely used models for time-varying sounds have been developed by (Fastl & Zwicker, 2007) and by (Glasberg & Moore, 2002), which are both openly available in the "Loudness Toolbox" for MATLAB (Genesis, 2009). In this paper, we use the Zwicker model, because the Moore model was found to have infeasibly high runtimes in the range of minutes for individual music pieces. Additionally, both models yield highly similar results (Webster & Jiuicek, 2014). The resulting function $l(t)$ represents the instantaneous loudness in sone over time and estimates how jarring a bixel transition would be perceived in terms of loudness changes.

For every bixel i, we define an average loudness l_i^{start} at the beginning and l_i^{end} at the end of the bixel as the weighted mean of $l(t)$ over the first or last 0.2 s of the bixel. The weights add up to 1 and increase linearly towards the bixel boundaries. Hence, we deem the loudness at bixel boundaries as more important due to their proximity to a potential cut point.

Loudness transition costs. The loudness transition costs result from two measures of loudness continuity that estimate how smoothly the loudness change is perceived that occurs between a pair of bixels i and j, $i, j \in \{1, \ldots, N\}$:

$$L_{i,j} = \min\{L_{i,j}^{\text{comp}}, L_{i,j}^{\text{trans}}\}. \tag{6}$$

$L_{i,j}$ are the entries of the final loudness matrix $\mathbf{L}$, $L_{i,j}^{\text{comp}}$ the entries of a normalised comparative loudness matrix $\mathbf{L}^{\text{comp}}$, and $L_{i,j}^{\text{trans}}$ the entries of a normalised transitional loudness matrix $\mathbf{L}^{\text{trans}}$. This results in high costs for jumps that fulfil neither of both requirements for loudness continuity. An example is given in Figure 5.

Transitional loudness is based on the idea that a transition from a bixel b_i to a bixel b_j is continuous in loudness if the end loudness l_i^{end} of bixel b_i is approximately equal to the start loudness l_j^{start} of bixel b_j:

$$L_{i,j}'^{\text{trans}} = |l_i^{\text{end}} - l_j^{\text{start}}|. \tag{7}$$

Comparative loudness assumes that a bixel transition from bixel b_i to a bixel b_j does not produce an irritating loudness change if the musical content that was expected after bixel b_i, namely bixel b_{i+1}, is approximately as loud at the beginning as the start of the bixel b_j which is played instead:

$$L_{i,j}'^{\text{comp}} = |l_{i+1}^{\text{start}} - l_j^{\text{start}}|. \tag{8}$$

We set $l_{N+1}^{\text{start}} = 0$ for the purpose of this calculation, because the listener expects silence after the last bixel and thus a loudness of 0 sone.

$L_{i,j}^{\text{trans}}$ and $L_{i,j}^{\text{comp}}$ result from $L_{i,j}'^{\text{trans}}$ and $L_{i,j}'^{\text{comp}}$, respectively, by normalising their range of values to the $[0, 1]$ interval.

4.2 Path optimisation

The goal of path optimisation is to find the best path from the start to the end bixel through the original song, that fulfils the given set of user constraints. Each path is assigned a cost equal to the sum of the costs of its jumps, designed to estimate the perceived quality of the resulting transitions. After specifying the optimisation problem, we will show an algorithm for finding its solution.

4.2.1 Problem formulation

A **bixel path** can be formally defined as a vector containing the indices of the k bixels visited along the path in their order of occurrence:

$$\mathbf{p} = (p_1, p_2, \ldots, p_k). \tag{9}$$

The task is to find a bixel path $\mathbf{p}$ of length of two or greater which minimises

$$c_{\text{path}}(\mathbf{p}) = \sum_{i=1}^{k-1} T_{p_i, p_{i+1}}, \tag{10}$$

with $\mathbf{T}$ as the unified cost matrix defined by Equation (3) in Section 4.1.2, subject to

(C1) $p_1 = b_{i_{\text{start}}}, p_k = b_{i_{\text{end}}}$, so that the desired start and end times t_{start} and t_{end} fall within the bixels p_1 and p_k respectively,

(C2) $\mathbf{p}$ has a duration within the target duration interval $[\Delta t_{\text{target}} - \Delta t_{\text{tol}}, \Delta t_{\text{target}} + \Delta t_{\text{tol}}] = [\Delta t_{\min}, \Delta t_{\max}]$,

(C3) $\mathbf{p}$ is composed of segments according to a target song structure with tolerances, based on an input song structure for example determined as in Section 4.1.

We tackle the optimisation problem using the A* algorithm for multiple goals on an appropriately defined graph. The solution will be described in two parts. The first part concerns the optimisation problem without constraint (C3), cf. Section 4.2.3. Afterwards this approach is extended to the full version including (C3), cf. Section 4.2.4.

The algorithm can only consider bixel paths up to a certain maximum length. This upper bound should be set depending on the maximum duration $\Delta t_{\max}$, which is equal to the sum of durations of all bixels in a path. A difficulty arises out of the fact that the bixels have variable duration. We will present a method for determining an upper bound on the path length in the following subsection.

4.2.2 Estimating the path length a priori

In this section, we will estimate how many bixels are expected to lead to a solution within the desired target duration with high probability, because the bixels are not all equally long in duration. We represent the duration of the n-th randomly selected bixel with a random, normally distributed variable $B_n \sim \mathcal{N}(\mu_{\text{b}}, \sigma_{\text{b}}^2)$ with the mean μ_{b} and variance σ_{b}^2 determined by the mean and variance of bixel durations in the music piece. When concatenating k randomly chosen bixels, the duration L_k of the resulting bixel path again follows a normal distribution:

$$L_k \sim \mathcal{N}(k \cdot \mu_{\text{b}}, k \cdot \sigma_{\text{b}}^2). \tag{11}$$

We then estimate the probability that a bixel path with $k \geq 2$ bixels, which contains $k-2$ freely selectable bixels (the start and end bixels are determined by the user), has a duration within the target duration range $[\Delta t_{\max}, \Delta t_{\min}]$ specified by the user in the interface (7) shown in Figure 2:

$$P_{\text{len}}(k) = P(\Delta t_{\min} \leq L_{k-2} \leq \Delta t_{\max}). \tag{12}$$

The probability $P_{\text{opt}}(k)$ of the optimal solution having a length of k is then obtained by normalising P_{len} over the interval $[2, \lceil \frac{\Delta t_{\max} - \Delta t_{\text{b}}}{\min_i b_i^{\text{len}}} \rceil]$, where Δt_{b} is the combined duration of the start and end bixels. The upper bound arises from the case in which only the shortest bixel is used to cover at least $\Delta t_{\max}$ seconds. To reduce computational costs, we compute an interval $[k_{\min}, k_{\max}]$ of minimum length while ensuring a given **minimum success probability** p_{succ} of finding the optimal solution within the interval, i.e.

$$\sum_{k=k_{\min}}^{k_{\max}} P_{\text{opt}}(k) \geq p_{\text{succ}}. \tag{13}$$

We construct this interval iteratively, beginning at $\arg\max_k P_{\text{opt}}(k)$ and adding the adjacent position with the highest function value until the sum of all included probabilities exceeds the minimum success probability p_{succ}, for which we will find a suitable default value in Section 6.1.1.

4.2.3 Multiple goal A* algorithm

In the following, we shall obtain the optimal bixel path based on the definitions from Section 4.2.1. (Wenner et al., 2013) employs dynamic programming based on a recursive formulation for the cost of the minimum cost path from bixel $b_{i_{\text{start}}}$ to any bixel b_i with k stops:

$$C_{\text{start}}(i, k) = \min_{j \in \{1,\ldots,N\}} \{C_{\text{start}}(j, k-1) + T_{j,i}\}. \tag{14}$$

Because the runtime complexity $\Theta(N^2 \cdot k_{\max})$ is quadratic in the amount N of bixels in the original piece, long input pieces cause prohibitively long waiting times. We will show a more efficient algorithm that is often considerably faster in practice despite its equivalent asymptotic complexity.

Formulation as a graph problem. The key idea behind our algorithm is that the problem can be represented as a **single-source shortest-paths problem** in a directed, weighted graph $G = (\mathcal{V}, \mathcal{E}, c_{\text{t}})$ with sets of vertices $\mathcal{V}$ and edges $\mathcal{E}$ (Figure 6). A vertex $v_{i,k} \in V$ represents the selection of bixel b_i as the k-th bixel in the bixel path. There is one vertex $v_{i,k} \in \mathcal{V}$ for each $i \in \{1, \ldots, N\}$ and $k \in \{2, \ldots, k_{\max}\}$, and an additional vertex $v_{i_{\text{start}},1}$ for the start bixel. The set of edges $\mathcal{E}$ contains an edge $e = (v_{i,k}, v_{j,k+1})$ from every node $v_{i,k}$ with $k < k_{\max}$ to all $v_{j,k+1}$ with $j \in \{1, \ldots, N\}$. They represent the act of transitioning between bixels in such a bixel path, and the cost $c_{\text{t}}(e)$ is set to the bixel transition cost $T_{i,j}$ from the unified cost matrix $\mathbf{T}$. In such a graph G the bixel path optimal with respect to Equation (14) for a given k is equivalent to the bixel path represented by the shortest path from the start node $v_{i_{\text{start}},1}$ to any of the end nodes $v_{i_{\text{end}},k}$ with $k_{\min} \leq k \leq k_{\max}$.

The dynamic programming approach from (Wenner et al., 2013) is equivalent to the calculation of the shortest path to every node in the graph G in a column-wise manner by considering every one of its predecessor nodes. A shortest path search can accelerate the process by exploring low-cost transitions first to get to the goal, avoiding to consider all transitions. We use the *A* algorithm* (Hart, Nilsson, & Raphael, 1968) for this search, as it visits the node v first that leads to the lowest total distance estimate, which is the sum

of the distance from the start node to v and the estimated minimum remaining distance $h(v)$ from v to the goal as calculated by a **heuristic**. We adapt the algorithm to the case of multiple goals by stopping the process as soon as any goal node is reached, as all paths to other goals are at least as long.

We experimented with hand-crafted heuristics to accelerate path optimisation further. However, the runtime on the standard test set used in the evaluation in Section 6 decreased by only 5.7%. Therefore, we only consider the A* algorithm with the trivial heuristic $h(v) = 0$, i.e. Dijkstra's shortest path algorithm, in the remainder of this paper.

4.2.4 Song structure enforcement with tolerances

We revisit the problem formulation from Section 4.2 and show how to include the song structure constraint (C3) into the path optimisation. After setting up the input song structure of the input track, the user can define a target song structure for the output track ((3) in Figure 2). For any time point in the output, the target song structure specifies which cluster usable segments have to be assigned to, and thus the allowed set of bixels. This allows the user to enforce specific structural constraints on the output.

In (Wenner et al., 2013), not only the input song structure is defined on a bixel-wise basis, but the target song structure also defines the cluster to which each bixel in a bixel path $\mathbf{p}$ has to belong in order to yield a valid solution. This is enforced by manipulating the cost matrix to only allow for transitions to bixels belonging to the current cluster, but severely limits the number of valid paths, which often increases the minimum cost of the optimal solution and leads to more perceptible transitions. Additionally, assigning the desired cluster to each bixel manually is time-intensive and unintuitive.

Instead, we allow the user to set segment boundaries at time points and the values of tolerance variables ((1) and (7) in Figure 2). The tolerance variables $\Delta t_{\text{low}}, \Delta t_{\text{high}} \in \mathbb{R}_{\geq 0}$ with $\Delta t_{\text{low}} \leq \Delta t_{\text{high}}$ allow the segment transitions of the resulting song to deviate from the target positions by up to Δt_{low} seconds without and Δt_{high} seconds with additional penalty.

The song structure enforcement is achieved by extending the initial problem definition from Section 4.2.1. We define a function $T(i, j, \Delta t) = f(T_{i,j}, \Delta t)$ that modifies transition costs $T_{i,j}$ between bixels b_i and b_j depending on the current duration Δt of all bixels in the shortest bixel path found so far. In particular, transitions to bixels belonging to currently undesired clusters are given a very high cost c_{inf}, which we set to 100 to ensure they are not used unless absolutely required by some other user constraint. Transition phases at segment boundaries are implemented by allowing the segment transition without penalty if the segment boundary is closer than Δt_{low} seconds. Transitions Δt_{low} to Δt_{high} seconds before and after the segment boundary incur a linearly decreasing and increasing penalty interpolating between zero and c_{inf}, respectively. Setting both Δt_{low} and Δt_{high} to zero makes the algorithm equivalent to the method proposed by (Wenner et al., 2013). Evaluating the function $T(i, j, \Delta t)$ at each calculation of a transition cost leads to paths which fulfil the given song structure constraints up to the given tolerance, if possible. An application example is shown in Figure 7.

5 Jump optimisation

This section concerns the improvement of jumps calculated in Section 4.2 by the jump synchronisation stage and the loudness equalisation stage of the system (Figure 3).

5.1 Jump synchronisation

Jump synchonisation aims to correct inaccuracies in the detected beat positions to prevent sound artefacts at the cuts, especially short transient sounds like snare drum hits that are not smoothed out by crossfading and would be heard twice or not at all.

We hypothesise that the jump times do not have to be aligned perfectly to the actual beat positions, but only need to have the same position relative to the beat. For every given jump (t_1, t_2) from an origin t_1 to a destination t_2, we extract Δt_{sync} seconds of audio before and after t_1 and t_2 respectively. We set $\Delta t_{\text{sync}} = 0.5$ s, so that even for slow songs with 60 beats per minute both audio excerpts cover at least two consecutive beats in duration for a musically meaningful comparison. Assuming that the signals near the jump origin and destination are similar and only delayed relative to each other, we use the **cross-correlation function** (Bracewell & Bracewell, 1986) $r(\Delta l)$ to find the alignment with most similarity. The cross-correlation is computed directly at the level of the audio signal. The amount of time by which the jump times are corrected is

$$\Delta l_{\max} = \arg\max_{\Delta l} r_{\text{w}}(\Delta l), \tag{15}$$

where

$$r_{\text{w}}(\Delta l) = P(S = \Delta l) \cdot r(\Delta l). \tag{16}$$

In case of $\Delta l_{\max} < 0$, jumping earlier is preferred to not interrupt music events at the beat positions with an earlier jump origin. For $\Delta l_{\max} > 0$, an earlier jump destination is set for the analogous reason. S is a random variable modeling the **synchronisation error** for a cut produced by a jump from any beat position to any other beat position, and is used as a weight in Equation (16) to avoid unrealistically large changes to the jump times. For a pair $(b_i^{\text{pos}}, b_j^{\text{pos}})$ of beat positions, the synchronisation error S of a jump is caused by the difference of delays D_i and D_j at each beat position, which in turn result from the deviation of the detected and the unknown actual beat position:

$$P(S = x) = P(D_i - D_j = x). \tag{17}$$

Unfortunately, D_i is difficult to determine empirically with a sufficient precision by comparing the detected beat positions with the ground truth beat data, due to the slight timing inaccuracies of human annotators. We model each D_i as a normally distributed random variable with a mean of $\mu_{\text{e}} = 0$ s, assuming that beats are not detected too early or too late on average. Furthermore, we set the standard deviation to $\sigma_{\text{e}} = 0.1$ s as it was found to prevent most of the problematic cuts during the evaluation in Section 2.2. The influence of σ_{e} on the behaviour of this synchronisation method will be explained in detail below. Then, Equation (17) yields

$$S \sim \mathcal{N}(0, 2\sigma_{\text{e}}^2). \tag{18}$$

Positive values can be interpreted as the amount of seconds that are unnecessarily included into the final music piece around the cut position. Negative values indicate missing audio material, resulting in a slight perceived "jump", as the next beat starts earlier than expected.

5.2 Loudness equalisation

Loudness equalisation aims to smooth out sudden loudness changes during transitions over a longer duration. The maximum duration is influenced by a parameter Δt_{lim}, but the actual duration is also dependent on the loudness ratio between the quieter and the louder signal, creating longer transitions for large loudness changes and vice versa.

Loudness equalisation is applied to every jump (t_1, t_2) with origin t_1 and destination t_2 belonging to the solution from the previous jump synchronisation step in Section 5.1. Parts of the loudness function $l(t)$ from Section 4.1.4 in the neighbourhood of the jump origin and destination, respectively, of equal duration are extracted and aligned with respect to the jump times:

$$l_1(t) = l(t_1 + t),\ l_2(t) = l(t_2 + t),\ t \in [-t_{\text{lim}}, t_{\text{lim}}]. \tag{19}$$

We found $t_{\text{lim}} = 3$ s to be a suitable value high enough to give the impression of a smooth loudness change even for jumps with large loudness differences. Firstly, we determine intersections between l_1 and l_2, where the loudness is equal for both excerpts. Loudness equalisation for the current jump is aborted if a time of equal loudness is closer than Δt_{thresh} to the cut position, because then no significant loudness difference exists when jumping. Determining a well-suited value for Δt_{thresh} is critical, as it represents how sensitive the correction is to short-term loudness changes. It should be considered how many seconds into the past the hearing impression is "integrated" by the human ear when perceiving loudness. (Everest & Pohlmann, 2009, Chapter 4) states this time to be 100 ms, thus we set Δt_{thresh} to 0.1 s, ensuring the average loudness before and after the jump can be reliably computed with an average over a sufficiently large time frame.

If loudness equalisation is not aborted, the loudness functions are used to derive two loudness change functions

$$l_{c,\text{before}}(t),\ t \in [t_{\text{before}}, t_1],\ l_{c,\text{after}}(t),\ t \in [t_2, t_{\text{after}}],$$

where $t_{\text{before}} < t_1$ and $t_{\text{after}} > t_2$ are chosen in a suitable manner which will be described later, cf. Equation (23). The loudness change functions are transformed into amplitude change functions by

$$a_{c,\text{before}}(t) = 10^{0.5 \cdot \log_2 l_{c,\text{before}}(t)},\ t \in [t_{\text{before}}, t_1], \tag{20}$$

$$a_{c,\text{after}}(t) = 10^{0.5 \cdot \log_2 l_{c,\text{after}}(t)},\ t \in [t_2, t_{\text{after}}], \tag{21}$$

according to an approximative rule (Sengpiel, 2018). The values of the amplitude change functions are used as factors by which the amplitude of the original signal before t_1 and after t_2, respectively, is multiplied, thus yielding the desired loudness equalisation.

The loudness change functions are defined so that similar values $l_{c,\text{before}}(t_1)$ and $l_{c,\text{after}}(t_2)$ are achieved at the transition time point and the original values are maintained at t_{before} and t_{after}. The values in between are obtained by an adaptive linear interpolation:

$$l_{c,\text{before}}(t) = s_{\text{before}}(t) \cdot l_{c,\text{before}}(t_1) + (1 - s_{\text{before}}(t)),\ t \in [t_{\text{before}}, t_1],$$

$$l_{c,\text{after}}(t) = s_{\text{after}}(t) \cdot l_{c,\text{after}}(t_2) + (1 - s_{\text{after}}(t)),\ t \in [t_2, t_{\text{after}}],$$

with

$$s_{c,\text{before}}(t) = \frac{\int_{t_{\text{before}}}^{t} l_{\text{diff}}(t' - t_1)\mathrm{d}t'}{\int_{t_{\text{before}}}^{t_1} l_{\text{diff}}(t' - t_1)\mathrm{d}t'},\ t \in [t_{\text{before}}, t_1],$$

$$s_{c,\text{after}}(t) = \frac{\int_{t_2}^{t_{\text{after}}+t_2-t} l_{\text{diff}}(t' - t_2)\mathrm{d}t'}{\int_{t_2}^{t_{\text{after}}} l_{\text{diff}}(t' - t_2)\mathrm{d}t'},\ t \in [t_2, t_{\text{after}}].$$

and (cf. Equation (19))

$$l_{\text{diff}}(t) = |l_1(t) - l_2(t))|,\ t \in [t_{\text{before}} - t_1, t_{\text{after}} - t_2].$$

$l_{c,\text{before}}(t_1)$ and $l_{c,\text{after}}(t_2)$ are defined by

$$l_{c,\text{before}}(t_1) = \frac{l_{\text{conv}}}{l'_{\text{before}}},\ l_{c,\text{after}}(t_2) = \frac{l_{\text{conv}}}{l'_{\text{after}}}.$$

l'_{before} and l'_{after} are determined by

$$l'_{\text{before}} = f_{\text{factor}} \cdot l_{\text{before}} + (1 - f_{\text{factor}}) \cdot l_{\text{mean}},$$

$$l'_{\text{after}} = f_{\text{factor}} \cdot l_{\text{after}} + (1 - f_{\text{factor}}) \cdot l_{\text{mean}},$$

with $l_{\text{mean}} = (l_{\text{before}} + l_{\text{after}})/2$. l_{before} and l_{after} are the mean of the values in the loudness function over Δt_{thresh} seconds before and after the cut position, respectively. The reason for the mean is that a single point on the loudness function only represents the instantaneous loudness at a specific moment. $f_{\text{factor}} \in [0, 1]$ offers the possibility to the user to correct only a portion of the detected loudness difference ((7) in Figure 2).

l_{conv} denotes the loudness to which both signals should converge at the cut position and is defined by

$$l_{\text{conv}} = \min\{l_{\text{mean}}, \min\{l'_{\text{before}}, l'_{\text{after}}\} \cdot f_{\max}\}. \tag{22}$$

The reason for this definition is that the loudness change should not necessarily be performed both before and after the jump to an equal degree due to potential clipping (Davis & Jones, 1989, Chapter 12, pp. 201–203) when amplifying the quieter signal. Instead, the loudness l_{mean} that is desired at the cut position is adjusted to incorporate more of the louder signal and less of the quieter signal, in case more attenuation can be applied than amplification. The possible amplification is controlled by a factor $f_{\max} \geq 1$ which is the inverse of the maximum amplitude around t_1 and t_2, i.e. is small for high amplitudes, and also limits the amplification to a constant factor for silent parts to avoid creating noise.

The still missing definitions of t_{before} and t_{after} are

$$t_{\text{before}} = t_1 + \Delta t'_{\text{corr}} - \Delta t'_{\text{trans}},\ t_{\text{after}} = t_2 + \Delta t'_{\text{corr}} + \Delta t'_{\text{trans}}. \tag{23}$$

$\Delta t'_{\text{trans}}$ is the duration of the gradual loudness change. It is made dependent on the relative loudness change

$$f_{\text{rel}} = \frac{\min\{l'_{\text{before}}, l'_{\text{after}}\}}{\max\{l'_{\text{before}}, l'_{\text{after}}\}}$$

from the louder to the quieter signal in such a way that the duration $\Delta t'_{\text{trans}}$ of the loudness change in both directions is proportional to the relative loudness change:

$$\Delta t'_{\text{trans}} = (1 - f_{\text{rel}}) \cdot \Delta t_{\text{trans}}.$$

Through subjective listening tests we found $\Delta t_{\text{trans}} = 3$ seconds to be a suitable setting providing loudness transitions smooth enough to not be irritating, but also not unnecessarily long.

To ensure the loudness is changed more or less equally fast across the whole transition, its position is moved $\Delta t'_{\text{corr}}$ away from the cut position in the direction of the louder signal, in case less than the required ideal amount of amplification can be realised, i.e. if $l_{\text{conv}} < l_{\text{mean}}$:

$$\Delta t'_{\text{corr}} = \text{sign}(l'_{\text{after}} - l'_{\text{before}}) \cdot \frac{l_{\text{mean}} - l_{\text{conv}}}{l_{\text{mean}} - \min\{l'_{\text{before}}, l'_{\text{after}}\}} \cdot \Delta t'_{\text{trans}}.$$

6 Evaluation

This section is devoted to an extensive evaluation and consists of two parts. The first part concerns the performance of the algorithmic concepts. The second part evaluates the quality of the generated rearrangements with a listening study.

6.1 Performance of the algorithmic concepts

In the following, we will investigate the path length estimation from Section 4.2.2, the performance of the path optimisation algorithm from Section 4.2.3, and the behaviour of the novel song structure enforcement from Section 4.2.4.

Database and setup. We extended the database CC1, which has an average track duration of 215 s, by including particularly long tracks from the Free Music Archive to evaluate the runtimes of the path optimisation algorithms more extensively in Section 6.1.2. The resulting database CC2 has 59 tracks listed in Appendix, Table 2 with an average track duration of 496 s.

In the upcoming sections, we perform a rearrangement for a list of configurations called the **standard configuration set**. It involves retargeting every song in CC2 to the durations $t_{\text{target}} \in \{60, 120, \ldots, 1200\}$. No tolerance is used ($\Delta t_{\text{tol}} = 0$) to ensure the original length is not within the target range Δt_{range}. For each duration, rearrangement is performed with enabled and disabled song structure enforcement for each duration, where the input song structure used is the song structure estimate from the automatic MSS with the default range of clusters $C_{\text{range}} = [3, 5]$. The target song structure is created by scaling boundary positions in the input song structure proportionally to the target duration t_{target}. The remaining user constraints are set to their default values.

6.1.1 Estimating the path length a priori

The estimated range of possible lengths for the optimal bixel path k_{range} is important for the path optimisation process, as its runtime is linearly dependent on the maximum

number of bixels $k_{\max}$. We employ the path length estimation of Section 4.2.2 multiple times using different minimum success probabilities p_{succ} and determine whether the best solution found in the estimated range k_{range} is the optimal one. On this basis, we measure **empirical success probability** p_{e} as the ratio between the successes and total trials. The results in Table 4 show that, as intended by our design, p_{e} is always at least as high as p_{succ}. Additionally, p_{e} is high even for low settings of p_{succ}, perhaps because of modelling the bixel length B_n as a normal distribution. However, few long bixels commonly present at the start or end of a track act as outliers that heavily increase the estimated variance, so the variation in bixel lengths is overestimated.

We set a default value of $p_{\text{succ}} = 0.6$ for our music rearrangement system, as it offers a satisfying trade-off between accuracy and runtime with an empirical success probability of over 95% despite a relatively low number of 246 goal nodes that have to be evaluated. Future work should explore more suitable distributions to model bixel lengths, especially accounting for outlier bixels at the start and end of songs.

6.1.2 Path optimisation algorithms

In this section, we compare the original approach using dynamic programming (DP) from (Wenner et al., 2013) and the multiple goal A* algorithm (A*) proposed in Section 4.2.3.

All experiments were performed on a computer equipped with an Intel i7 3.4 GHz quad-core processor and 16 GB of RAM.

For the standard configuration set, the multiple goal A* algorithm with an average runtime of 2.63 s successfully improves on the dynamic programming approach with 4.68 s. Referring to the estimation of k_{range} in Section 6.1.1, Figure 8 illustrates the relationship between the minimum success probability p_{succ} and the average runtime of all test cases with a certain setting for p_{succ}. Because the set of goal nodes grows in size with increasing values for p_{succ}, as shown in Table 4, both tested algorithms tend to require increasingly more time.

In the following, we analyse the runtime as a function of the length of the original piece as well as the target duration t_{target} both with and without using song structure enforcement. Due to the variation in performance for different tracks, we approximated each function using *locally weighted regression* with the *loess* function (R Core Team, 2017). Figure 9 contains plots of the resulting runtimes. As expected from the asymptotical theoretical complexity, the runtimes of both algorithms seem to be quadratic in the duration of the original track and linear in the target duration Δt_{range}.

For input tracks of approximately three minutes or less, the algorithms have a similarly low runtime, with DP sometimes being slightly faster due to less data structure overhead. For longer tracks however, the A* algorithm is faster than DP, and this advantage increases with the duration of the input as well as the output track.

Interestingly, enabling the song structure enforcement does not change the runtimes of the A* algorithm to a large extent, while DP becomes faster. This could be due to optimized processor instructions exploiting the structure induced in the problem graph by the song structure enforcement, but further investigation is needed.

While all algorithms share the asymptotic complexity of $O(N^2\, k_{\max})$, the A* algorithm has lower scaling factors associated with N and $k_{\max}$ than DP. Overall, the multiple goal A* algorithm appears to be clearly superior to DP for longer tracks, and on par or only

insignificantly slower for short tracks.

6.1.3 Segment boundary tolerance

In this section, we will use the standard configuration set with the tolerance parameters $\Delta t_{\text{low}} \leq \Delta t_{\text{high}}$, cf. Section 4.2.4, set to values in $\mathcal{P} = \{0, 2, 4, 6, 8, 10\}$ (in seconds) to investigate whether they offer the desired trade-off between transition quality and segmentation accuracy. Note that the configuration with $\Delta t_{\text{low}} = 0$ and $\Delta t_{\text{high}} = 0$ represents the song structure enforcement without any tolerance employed in (Wenner et al., 2013).

We define **segmentation accuracy** as the amount of time the output song structure aligns with the target song structure, divided by the total duration of the output track. Alignment is achieved at any point in time at which the bixel currently used for the output belongs to the same segment cluster as the one defined in the target song structure. The overall audio quality of the output track is estimated with the total cost $c_{\text{path}}(\mathbf{p})$ of the optimal bixel path and the number of jumps in the solution, assuming that the perceived transition quality correlates sufficiently with the costs in the transition cost matrix $\mathbf{T}$ and output tracks with less jumps tend to have better quality. The extent of this correlation will be investigated in Section 6.2.1.

The results are shown in Figure 10 and visualise the average values of the above metrics for the different Δt_{low} and Δt_{high} using heatmaps.

Figure 10 (a) shows that using no tolerance in the lower left leads to almost perfect segmentation accuracy, but it also does not fall below 90% even for the largest tolerance setting. In exchange for lower segmentation accuracy, the average cost $c_{\text{path}}(\mathbf{o}_{k_{\text{opt}}})$ of the optimal bixel path $\mathbf{o}_{k_{\text{opt}}}$ was considerably lower with only a small tolerance $\Delta t_{\text{low}} = \Delta t_{\text{high}} = 2$ than without any tolerance, as Figure 10 (b) demonstrates. However, further increasing the tolerances yields diminishing returns. Assuming the bixel path costs reflect the perceived output quality, even a small segmentation boundary tolerance leads to a significant quality increase. Furthermore, the number of jumps shown in Figure 10 (c) and the number of "wrong jumps" that extend the output piece in case shortening is required and vice versa both decrease on average with increasing tolerances.

Considering the above results, we set $\Delta t_{\text{low}} = 2$ and $\Delta t_{\text{high}} = 4$ as the default for our music rearrangement system.

In conclusion, the concept of segment boundary tolerance was successful in providing a trade-off between segmentation accuracy and output quality, assuming the cost of a bixel path and the number of jumps provide an indication of the subjective output quality.

6.2 Listening study

To evaluate the perceived sound quality offered by our music rearrangement system, we determine the quality of the produced cuts as the critical factor for the overall output quality. Consequently, we conducted a listening study that presented a series of music excerpts with cuts generated by the system and asked the participants to rate their quality according to different aspects. We collected the required data from participants through two online surveys each with a different set of music tracks, where audio snippets could be played as often as desired. In total, we obtained 28 responses for the first and 45

responses for the second survey.

Used database. To determine whether knowing a music piece influences the listener responses, we selected four popular tracks as well as four music pieces from the Free Music Archive (WFMU, 2009) that were unfamiliar to all of the participants. The database EVAL presented in Appendix, Table 3 lists these 8 tracks, where the first four entries represent the popular pieces. Furthermore, the pieces were selected so that a large range of genres is covered. To limit the study duration to about 15 minutes, only the tracks from the database EVAL were used and also split across two different online surveys, each including two popular and two unfamiliar tracks.

Data generation. To generate the examples for the survey, we define the **reduced configuration set** as the set of configurations. It involves doubling and halving the length of every song in database EVAL, without any tolerance Δt_{tol}, with and without song structure enforcement. Although considerably smaller than the standard configuration set, the reduced configuration set was designed to cover the main application scenarios. Like in the standard configuration set, the target song structure is a scaled version of the input song structure and all other parameters take their default values.

Additional questions. Complementing the main questions presented in the following sections, we explore potential effects of the musical education of listeners and their knowledge of the tracks on their responses. In both online surveys, participants were asked to categorise their level of musical education by choosing one of the options "No musical education", "Amateur musician", "Semi-professional", and "Professional musician". The online surveys were structured according to the four original music pieces from the EVAL database. At first, the listeners were able to listen to a preview of the original track and to indicate whether they know the track. Afterwards, a set of audio snippets generated with this track as input was presented before proceeding with the next original track.

6.2.1 Transition quality

Arguably the most important criterion for evaluating a music rearrangement system is the quality of the produced tracks. In our cut-based approach, this is mostly dependent on the quality of the transitions, which will thus be examined here.

Data generation. Song structure enforcement was disabled due to the high number of produced jumps. We disabled postprocessing, as it is evaluated separately in Section 6.2.2. For every remaining configuration in the reduced configuration set, we use different values $\{0, 0.1, \ldots, 0.4\}$ for the weight w_{l} of the loudness matrix $\mathbf{L}$, setting $w_{\text{d}} = 1 - w_{\text{l}}$. Finally, a random configuration from this configuration set was chosen for each track from EVAL to generate the cuts.

Study design. The audio excerpts were presented to each participant, who was asked to give three ratings on a scale from one to five, with higher numbers indicating a higher perceived quality. The three aspects to be rated were loudness continuity, all other factors required for an imperceptible transition, such as timbre, rhythm, and meter, and

the overall quality of the audio snippet as a combination of the former two ratings. We deliberately separated the loudness continuity from all other factors to focus on evaluating the usefulness of the proposed loudness matrix **L**.

Results. The average ratings over all cuts for loudness continuity, all other musical aspects, and the transition quality were 3.56, 3.35, and 3.27, respectively. Surprisingly, the average overall rating is lower than both of the individual ratings that it is comprised of. A reason could be a misunderstanding of the category "Other", because some of the corresponding aspects like timbre were perhaps unknown.

The average ratings as a function of the loudness weight for the loudness matrix **L** are plotted in Figure 11. The red curve demonstrates that the loudness matrix successfully captures the perception of loudness continuity, as cuts with higher perceived loudness continuity are selected with increasing w_l. The ratings concerning all other musical aspects ("Other") first decreased in the process as expected, but suddenly increased slightly for $w_\mathrm{l} > 0.2$. Possibly, the loudness matrix **L** unintentionally encodes some information about these aspects in addition to loudness continuity.

Next, we directly investigate to what extent the costs for a bixel transition from bixel b_i to b_j defined by the transition cost matrices reflect their perceived transition quality – ideally, there should be a perfectly linear relationship. However, we found no significant correlation between the bixel transition costs D_{ij} concerning timbre and the corresponding rating in the "Other" category, average over users. Although metrical aspects are a confounding factor as they are dependent on the beat tracking accuracy, the complete absence of a relation is still surprising and calls for a revision of the transition costs in **D**. On the other hand, a Spearman rank correlation test showed a significant correlation ($p < 0.05$) with a correlation coefficient $\rho \approx -0.35$ between the cost $L_{i,j}$ of a bixel jump and its average loudness rating. This proves the ability of our proposed loudness matrix **L** to at least partially reflect the actual transition quality regarding loudness.

No significant relationships were found between the musical education of the listener and the ratings ($p > 0.05$). Interestingly, familiarity with the track increased the overall ratings by 0.2 to 0.48 points on average (paired Wilcoxon signed-rank test, $p < 0.05$).

6.2.2 Jump optimisation

In this section we evaluate the influence of the jump optimisation stage on the perceived transition quality of jumps.

Data generation. Again, the reduced configuration set was used to generate the output, this time once with and once without jump optimisation to analyse the differences. For every cut, two audio snippets each containing six seconds of audio centered around the cut position were extracted, of which one was additionally processed with jump optimisation.

Two pairs were randomly selected for every track in the database EVAL with the condition that they were **significantly altered** by the jump optimisation stage, to focus on examples where the jump optimisation had a noticeable effect. We define a bixel jump to be significantly altered if it exhibits an absolute loudness difference $|l_\mathrm{after} - l_\mathrm{before}|$ of at

least 5 sone and/or an absolute detected synchronisation error $|\Delta l_{\max}|$ of at least 0.05 s (cf. Section 5.2). From all 556 generated pairs, 511 were significantly altered, implying that the jump optimisation procedure noticeably affects most cuts and is thus important to evaluate.

Study design. For each pair of audio snippets, we asked the participants to select the more pleasing audio snippet, also allowing a third choice to indicate no noticeable difference. No further information was given, forcing participants to make their choice based on the audio alone.

Results. Figure 12 illustrates the distribution of the responses averaged over all examples, grouped by the level of musical education.

Overall, the quality of the cuts was improved or left unchanged by the jump optimisation procedure in most cases (73%). Particularly interesting is that the percentage of responses indicating the preference of the unedited audio snippet ("Preferred unedited") tended to decrease with increasing reported levels of musical education. However, this effect is not necessarily significant, especially because only four of the 73 participants identified themselves as professional musicians (level of musical education: 4) and consequently, the corresponding results are subject to deviations caused by outliers due to the low sample size.

We investigated the cases in which jump optimisation worsened the audio quality by analysing how the activity of loudness equalisation and jump optimisation respectively affects the response scores. Larger loudness adjustments $|l_{\text{after}} - l_{\text{before}}|$, correlated with more positive responses, while a higher corrected absolute delay $|\Delta l_{\max}|$ during jump optimisation correlated with more negative responses. These correlations were significant and imply that jump optimisation could incorrectly change the jump timings, leading to irritating rhythmic inconsistencies.

We found no significant differences in the average response score when only considering the responses made with knowledge of the respective track used to generate the audio snippets.

7 Conclusions

We have proposed a novel music rearrangement system with an intuitive graphical user interface. Given an existing piece of music, a target length, and a set of user constraints, it allows the production of a new music piece fulfilling these constraints by piece-wise concatenation of sections of the original piece. The rearrangement task is formulated as shortest path finding problem in a graph which represents transitions between bixels and is optimised using a multiple goal A* algorithm, where the jumps between audio sections are smoothed with respect to loudness and timbre of related bixels.

The system was thoroughly evaluated with the help of performance tests and two listening surveys. The results show a decrease of computational complexity compared to the previous approach and an increase in perceived transition quality for different genres in general, especially regarding loudness continuity.

As a further contribution, previous work in the field was intensively evaluated with a self-built database containing a wide variety of music pieces. The results revealed various problems occurring at the cut positions in the produced pieces.

Some of the issues remain for future work, such as cutting off vocals which is especially important as it occurs frequently and listeners tend to focus on vocal sounds. But also other relevant high-level semantic entities of music like harmonic progressions, melodic contours, repeated motifs, etc. may be preserved after the rearrangement. For that goal, a more comprehensive metric of transition quality could be developed. It could also be viable to combine such further musical aspects that influence the transition quality using weights optimised with the help of listening studies to yield a transition cost metric that more accurately reflects human perception. Furthermore, the consideration of semantic music properties may lead to extensions in the user interface, for instance marking of phrases and segments which are not allowed to contain cuts.

References

Bracewell, R. N., & Bracewell, R. N. (1986). *The fourier transform and its applications* (Vol. 31999). McGraw-Hill New York.

Celeux, G., & Govaert, G. (1992). A classification EM algorithm for clustering and two stochastic versions. *Computational Statistics and Data Analysis*, *14*(3), 315–332.

Davies, M. E. P., & Plumbley, M. D. (2007). Context-dependent beat tracking of musical audio. In *IEEE international conference on acoustics, speech, and signal processing (ICASSP)* (Vol. 15, p. 1009-1020).

Davis, G., & Jones, R. (1989). The sound reinforcement handbook. In Y. Corporation (Ed.), (pp. 201–203). Hal Leonard Corporation.

Einbond, A., Trapani, C., Agostini, A., Ghisi, D., & Schwarz, D. (2014). Fine-tuned control of concatenative synthesis with catart using the bach library for Max. In *International computer music conference (ICMC)* (pp. 1037–1042).

Everest, F. A., & Pohlmann, K. (2009). *Master handbook of acoustics.* McGraw-Hill Education.

Fastl, H., & Zwicker, E. (2007). *Psychoacoustics: Facts and models* (Vol. 22). Springer Science & Business Media.

Fletcher, H., & Munson, W. A. (1937). Relation between loudness and masking. *The Journal of the Acoustical Society of America*, *9*(1), 78.

Genesis. (2009). *MATLAB loudness toolbox.* `http://genesis-acoustics.com/en/loudness_online-32.html`. (Accessed: 2018-01-21)

Glasberg, B. R., & Moore, B. C. J., Moore. (2002). A model of loudness applicable to time-varying sounds. *Journal of the Audio Engineering Society*, *50*(5), 331–342.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. In *IEEE transactions on systems science and cybernetics* (Vol. 4, p. 100-107).

Kühn, C. (1987). *Formenlehre der musik.* Deutscher Taschenbuch.

Olson, H. F. (1972). The measurement of loudness. *Audio Magazine*, 18–22.

Parker, J. R., & Behm, B. (2004). Creating audio textures by example: tiling and stitching. In *IEEE international conference on acoustics, speech, and signal processing (ICASSP)* (pp. 317–320).

R Core Team. (2017). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from `http://www.R-project.org/`

ROLI Ltd. (2004). *JUCE framework.* Retrieved from `www.juce.com` (Accessed: 2018-01-21)

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, *20*, 53–65.

Rowland, D. (2010). *dRowAudio - a JUCE module for high level audio application development.* `http://drowaudio.co.uk/docs/`. (Accessed: 2018-01-21)

Sato, H., Hirai, T., Nakano, T., Goto, M., & Morishima, S. (2016). A soundtrack generation system to synchronize the climax of a video clip with music. In *IEEE international conference on multimedia and expo (ICME)* (pp. 1–6).

Schwarz, D., Cahen, R., & Britton, S. (2008). Principles and applications of interactive corpus-based concatenative synthesis. In *Journées d'informatique musicale (JIM).*

Sengpiel, E. (2018). *The human perception of loudness.* `http://www.sengpielaudio.com/calculator-loudness.htm`. (Accessed: 2018-01-21)

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(8), 888–905.

Tauscher, J., Wenger, S., & Magnor, M. (2013). Audio resynthesis on the dancefloor: A music structural approach. In *Vision, modeling and visualization (VMV)* (pp. 41–48).

Verhelst, W., & Roelands, M. (1993). An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech. In *IEEE international conference on acoustics, speech, and signal processing (ICASSP)* (p. 554-557).

Webster, P., & Jiuicek, O. (2014). A brief comparison of loudness evaluation methods. *Acoustic Sheets, Czech Technical University in Prague*, *20*(2), 8–11.

Wenger, S., & Magnor, M. (2011). Constrained example-based audio synthesis. In *International conference on multimedia and expo (ICME)* (pp. 1–6).

Wenger, S., & Magnor, M. (2012). A genetic algorithm for audio retargeting. In *ACM multimedia (ACMMM)* (pp. 705–708).

Wenner, S., Bazin, J.-C., Sorkine-Hornung, A., Kim, C., & Gross, M. (2013). Scalable music: Automatic music retargeting and synthesis. *Computer Graphics Forum (Proceedings of Eurographics 2013)*, *32*(2), 345-354.

WFMU. (2009). *Free music archive.* `http://freemusicarchive.org/`. Retrieved from `http://freemusicarchive.org/` (Accessed: 2018-01-21)

Appendix

Table 1: Database CC1

Genre	Subgenre	Artist	Title
Blues		Arne Bang Huseby	Stormy Blues
Blues		Julia Haltigan	All I Can Think Of Is You
Country		Randy Travis and Brandon	Amazing Grace
Electronic	Ambient	LASERS	Amsterdam
Electronic	Chip	an0va	The First Noel / O Come All Ye Faithful
Electronic	Dance	Broke For Free	Calm The Fuck Down
Electronic	Downtempo	Tours	Enthusiast
Electronic	Dubstep	LittleLight	Illumination
Electronic	Glitch	Oribque	Simple
Electronic	IDM	Pierlo	Barbarian
Electronic	Techno	Foniqz	Spectrum (Subdiffusion Mix)
Electronic	Trip-Hop	_ghost	Lullaby
Experimental	Avantgarde	Jared C. Balogh	Equal Value (Ode To A Squirrel)
Experimental	Drone	The Upsidedown	E-Love
Experimental	Electroacoustic	Origamibiro	Flicker
Experimental	Field Recording	Aaron Ximm	Spring Rain
Experimental	Musique Concrete	Computer Truck	Euritmix Sux My Dix
Folk		Gaby Cardoso	Mujerzuela
Folk	Psych	The New Mystikal Troubadours	Tonight: A Lonely Century
Folk	Singer-Songwriter	Great Lake Swimmers	Gonna Make It Through This Year
Hip-Hop		Blackwell	I Neva
Instrumental		David Lohstana	Petit talible (instrumental version)
Instrumental		The Kyoto Connection	Hachiko The Faithful Dog (short)
International		Los Amparito	El barzón
Jazz		Kevin MacLeod	AcidJazz
Classical		Mozart by CSU	Symphony No. 40 IV
Classical		Beethoven	Moonlight Sonata (short)
Classical		Beethoven	Für Elise
Pop		Fresh Body Shop	Fireballs
Pop		Lilly Wolf	Jealousy
Pop		On returning	Paris (energy of life)
Rock	Alternative	Blind Violet	Deep
Rock	Garage	Artistes d'Origine in Contrôlée	Reveille toi - Dewey
Rock	Indie	Dumbo Gets Mad	Plumy Tale
Rock	Industrial	Nine Inch Nails	7 Ghosts I
Rock	LoFi	Tyrannic Toy	Blackroad
Rock	Metal	Insane Ride	Wrong or Right
Rock	Noise	Lately Kind Of Yeah	Where Is My Jaw?
Rock	Postpunk	Mules	Teenage Freakout
Rock	Postrock	et	Kopeika
Rock	Progressive	Convey	Campaign Speech
Rock	Psych	Flowerheads	06
Rock	Punk	Angstbreaker	Dead Elements
Soul/RnB		Juanitos	Hey

Table 2: Database CC2. Contains database CC1

Genre	Subgenre	Artist	Title
Folk		Labib	Saleh Bey
Folk	Singer-Songwriter	Cian Nugent	Double Horse
International		Caligine	Me Piánoune Zaládes
Jazz		Quantum Jazz	If I Can't Dance It's Not My Revolution
Rock	Psych	Noi	Everything Is Changing
Experimental	Avantgarde	Ellen Fullman	Never Gets Out Of Me
Classical		Gio Micheletti	Happy Sinners
Classical		Massimo Mastrangeli	Bagliori di tempesta
Classical		Maya Filipič	Anthos
Classical		Rob Costlow	Bliss
Electronic	Techno	Adrian Sanchez	Adrian Sanchez G Pal Ophra
Electronic	Techno	Dj Rostej	Light Rays
Electronic	Techno	-mystery-	Decadence
Electronic	Techno	Saelynh	Summer in Paradise
Electronic	Techno	Synthager	The Way of Atlant

Table 3: Database EVAL

Genre	Subgenre	Artist	Title
Rock		Beatles	Hey Jude
Pop		Leann Rimes	How Do I Live
Rock	Hard Rock	Motörhead	Ace Of Spades
Classical		Columbia State Univ. Orchestra	Mozart's Symphony No. 40 IV
Hip-Hop		Blackwell	I Neva
International		Los Amparito	El barzón
Folk	Singer-Songwriter	Great Lake Swimmers	Gonna Make It Through This Year
Electronic	Ambient	LASERS	Amsterdam

List of Figures

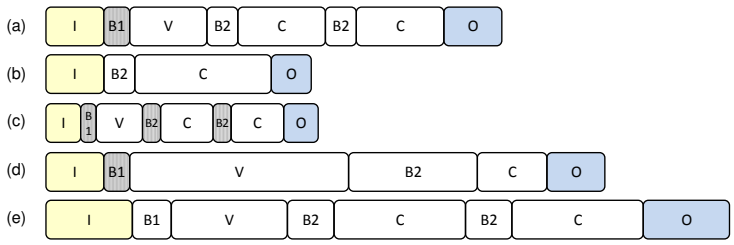

Figure 1: Segments of the original (a) and rearranged music piece (b)-(e). I: intro; B1: bridge 1; V: verse; B2: bridge 2; C: chorus; O: outro.

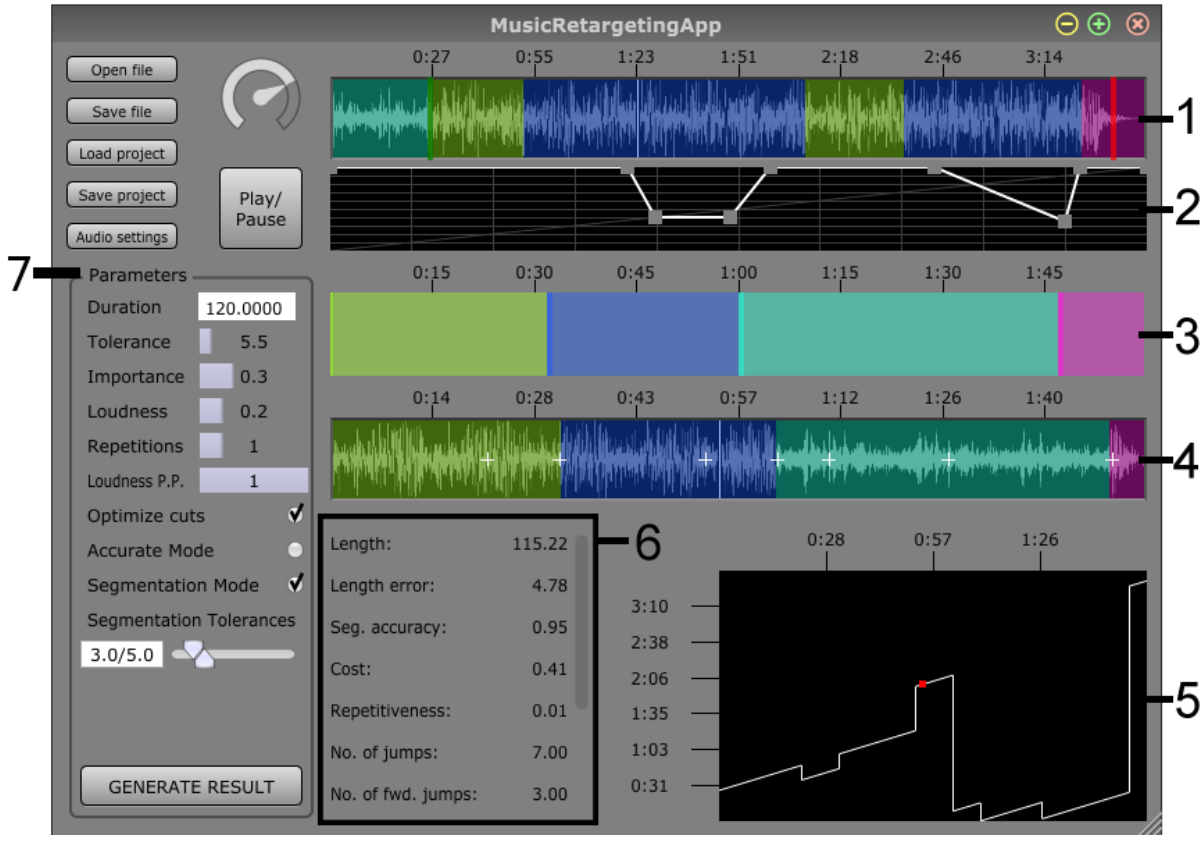

Figure 2: Screenshot of the user interface for the music rearrangement system in the rearrangement stage, enabling the user to generate a new music piece from the original piece based on the different constraints.

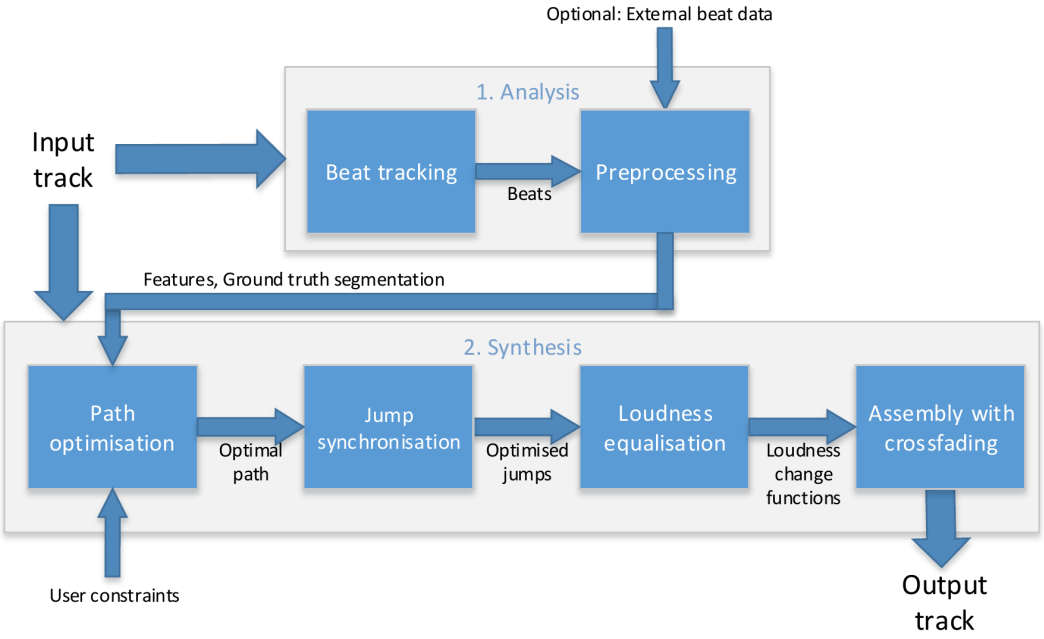

Figure 3: Overview of the proposed music rearrangement system.

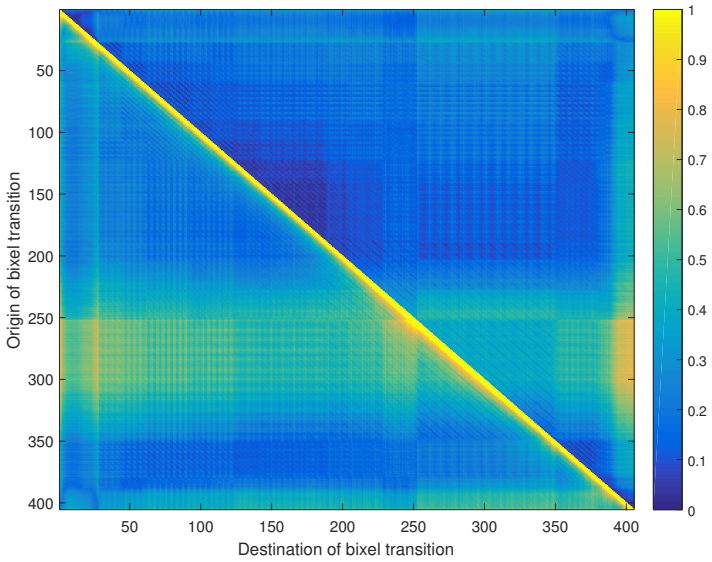

Figure 4: Unified cost matrix $\mathbf{T}$ for the song "El barzón" from "Los Amparito" with a repetition avoidance setting of $w_{\mathrm{r}} = 4$.

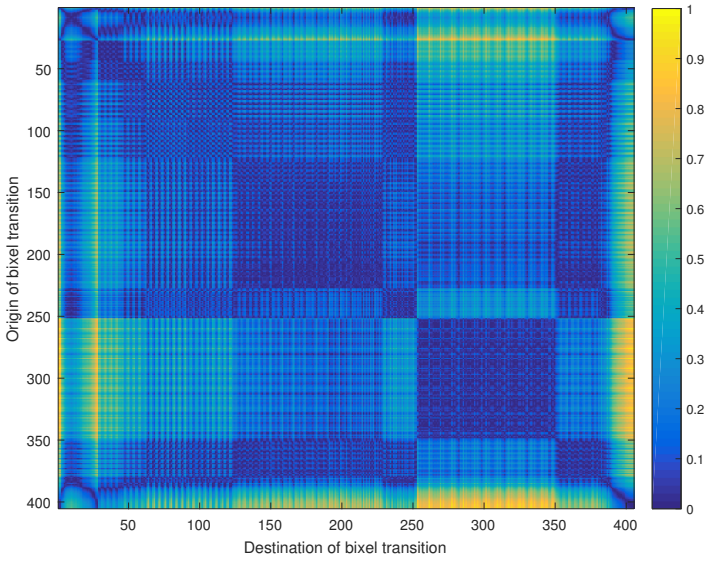

Figure 5: Loudness matrix $\mathbf{L}$ computed according to Equation (6) for the example song "El barzón" from "Los Amparito".

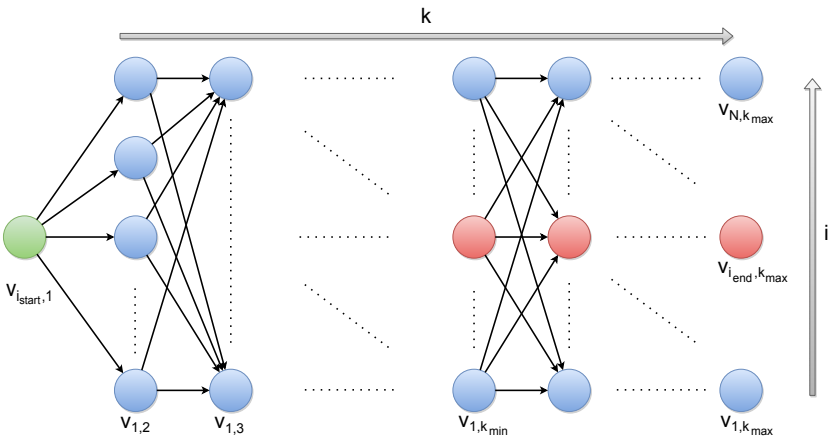

Figure 6: The graph G representing the problem space with the current position in the considered bixel path k on the horizontal and the bixel number i of the included bixel on the vertical axis. The start node $v_{i_{\text{start}},1}$ is coloured in green, while the set of goal nodes $\mathcal{G}_{\text{end}}$ is shown in red.

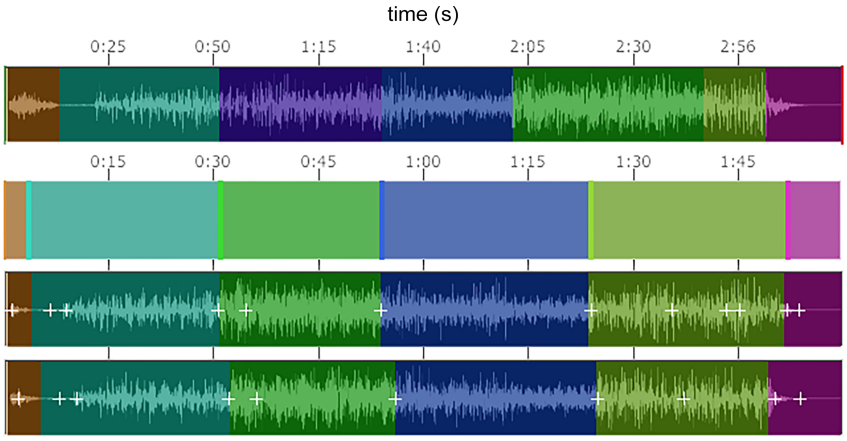

Figure 7: Rearranging "El barzón" from "Los Amparito" with song structure enforcement to 120 s without tolerance. From top to bottom: Ground truth segmentation with seven different clusters, target song structure with desired order of clusters, resulting song with $\Delta t_{\text{low}} = \Delta t_{\text{high}} = 0$, and resulting song with $\Delta t_{\text{low}} = 3$ and $\Delta t_{\text{high}} = 5$. White crosses indicate cuts with bixel jumps.

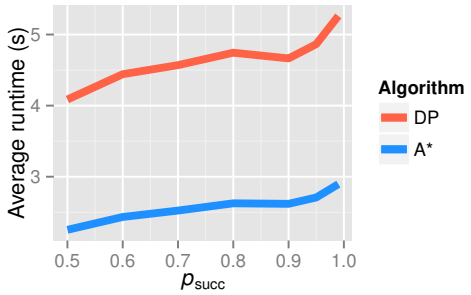

Figure 8: Average runtimes in seconds for the dynamic programming and the A* algorithm presented in section 4.2.3, for different minimum success probabilities p_{succ}.

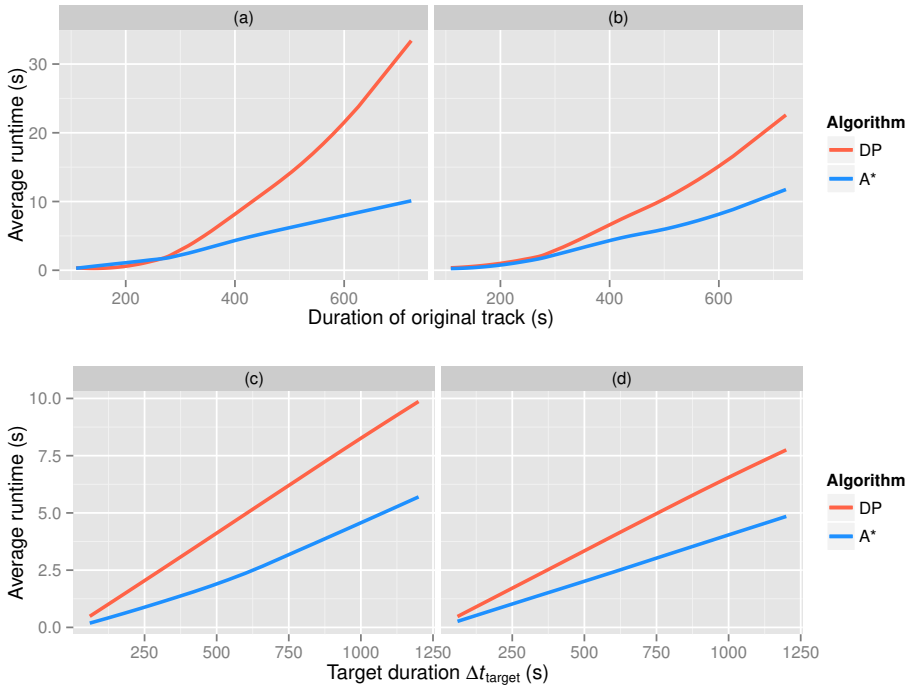

Figure 9: Average runtimes for the path optimisation algorithms. Song structure enforcement is disabled for the plots in (a) and (c) and enabled for the plots in (b) and (d).

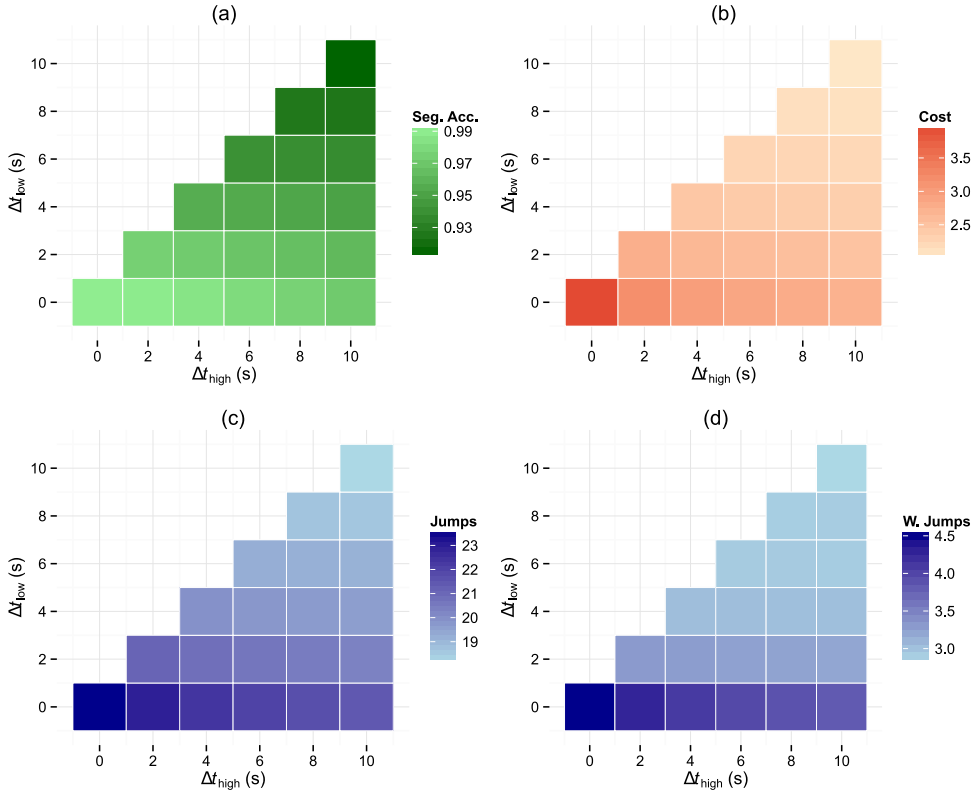

Figure 10: Heatmaps visualising the average (a) segmentation accuracy, (b) cost of the bixel path, (c) number of jumps in the bixel path and (d) number of wrong jumps in the bixel path for various configurations of the segment boundary tolerances Δt_{low} and Δt_{high}. Brighter colour is better.

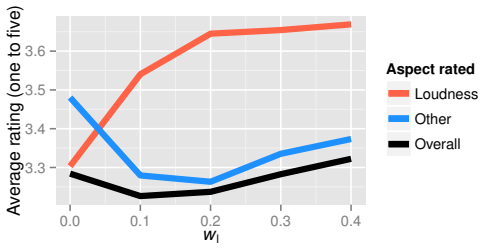

Figure 11: Average ratings of the loudness continuity ("Loudness"), all other musical aspects ("Other") and of the transition quality as a whole ("Overall") for different values of the loudness weight w_l.

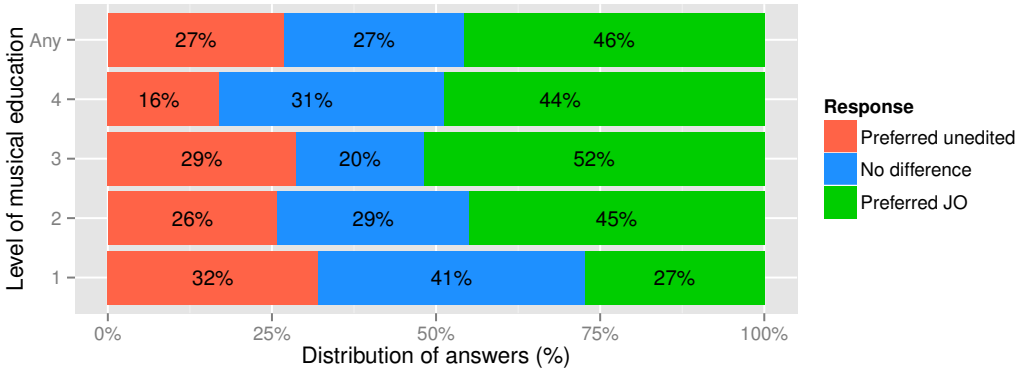

Figure 12: Distributions of responses as an average over all pairs of audio snippets, for different levels of musical education of the participants.

List of Tables

p_{succ}	0.50	0.60	0.70	0.80	0.90	0.95	0.99
p_{e}	0.9267	0.9560	0.9820	0.9904	0.9988	0.9992	0.9996
k_{goal}	201	246	311	389	485	579	664

Table 4: Averages of the empirical success probabilities p_{e} and the number of goal nodes $k_{\text{goal}} = k_{\max} - k_{\min} + 1$ for different minimum success probabilities p_{succ}, cf. Section 4.2.2.