SHEET MUSIC-AUDIO IDENTIFICATION

Christian Fremerey, Michael Clausen, Sebastian Ewert
Bonn University, Computer Science III
Bonn, Germany
{fremerey,clausen,ewerts}@cs.uni-bonn.de

Meinard Müller
Saarland University and MPI Informatik
Saarbrücken, Germany
meinard@mpi-inf.mpg.de

ABSTRACT

In this paper, we introduce and discuss the task of sheet music-audio identification. Given a query consisting of a sequence of bars from a sheet music representation, the task is to find corresponding sections within an audio interpretation of the same piece. Two approaches are proposed: a semi-automatic approach using synchronization and a fully automatic approach using matching techniques. A workflow is described that allows for evaluating the matching approach using the results of the more reliable synchronization approach. This workflow makes it possible to handle even complex queries from orchestral scores. Furthermore, we present an evaluation procedure, where we investigate several matching parameters and tempo estimation strategies. Our experiments have been conducted on a dataset comprising pieces of various instrumentations and complexity.

1 INTRODUCTION

When listening to an audio recording of a piece of music, an obvious problem is to decide, which bar of a corresponding sheet music representation is currently played. For technical reasons, we tackle this problem from the viewpoint of *sheet music-audio identification*: Given a sequence of bars from the sheet music as a query, the task is to find all temporal sections in the audio recording, where this bar sequence from the query is played.

One application of this task is to find out, whether there are differences between the default bar sequence following the instructions in the sheet music and what is actually played in the audio interpretation. In case there are differences, sheet music-audio identification may also be used to automatically determine the bar sequence that is played in the interpretation, and to identify special parts like cadenzas that have no counterpart in the sheet music.

If the bar sequence played in the audio interpretation is known in advance, sheet music-audio identification can be solved by first performing sheet music-audio synchronization and then using the synchronization results to identify the temporal sections in the audio that correspond to a given query sequence of bars. In case the correct bar sequence is not known, a more direct approach must be taken. Here, sheet music-audio matching as performed in [1] seems to be a reasonable strategy.

In the literature, alignment, identification and retrieval has been a popular field of research for the single-domain cases of either audio or symbolic data, see [2] and the references therein. For the cross-domain case, a lot of effort has been put into the task of off-line and on-line alignment of score data and audio data [3–6]. Here, the assumption is made that the bar sequence of the score is already known. The idea of using cross-domain synchronization results as ground truth or training data for more complicated music information retrieval tasks has already been formulated for the application of automatic transcription of pop music [7].

First important steps towards cross-domain matching and identification of polyphonic musical works have been conducted by the groups of Pickens and Orio [4, 8]. Using either audio transcription techniques [8] or a statistical model for the production of audio data from polyphonic score data [4] a complete audio track (song or movement) is used as a query to find the corresponding work in the score domain. First experiments for approaching the task of cross-domain work identification by querying arbitrary segments of score data have been conducted by Syoto et al. [9] as well as in our previous work [1]. None of the above approaches explicitly handles differences in bar sequence structure or repeats between the score and audio data, even though this is a common and practically relevant issue in real-world digital music libraries.

The paper is structured as follows. Section 2 specifies the task of sheet-music audio identification in more detail and discusses some difficulties and pitfalls. Our two approaches to sheet music-audio identification are presented in Section 3, one using synchronization and the other using matching. Section 4 explains how MIDI events for comparison with the audio data are created from the sheet music data. The synchronization and matching procedures are outlined in Sections 5 and 6. Section 7 describes an evaluation procedure for the matching approach using the more reliable results of the synchronization approach as a ground truth. Experimental results using our test dataset are discussed in Section 8 before the paper concludes with an outlook on future work in Section 9.

We gratefully acknowledge support from the German Research Foundation DFG. The work presented in this paper was supported by the PROBADO project (http://www.probado.de/, grant INST 11925/1-1) and the ARMADA project (grant CL 64/6-1).

2 SHEET MUSIC-AUDIO IDENTIFICATION

In the following, we assume that we are given one scanned sheet music representation and one audio interpretation of the same piece of music. We assign a unique label (p, b) to each bar written in the sheet music, where p is the page number and b is the bar number on the page. Furthermore, B denotes the set of all bar labels of the piece. Sheet music may contain jump directives like repeat signs, alternative endings, dacapos or segnos. Following these directives as they are written in the sheet music, one obtains a sequence $\delta = (\delta_1, \ldots, \delta_n)$, $\delta_i \in B$, indicating the *default sequence* of bars that is to be played when performing the piece. In practice, however, the given audio recording does not always follow this sequence δ. Performers might, for example, choose to ignore or add repeats, or even introduce shortcuts. This leads to a possibly different sequence $\pi = (\pi_1, \ldots, \pi_d)$, $\pi_i \in B \cup \{\uparrow\}$, which we call *performance sequence*. Here, we use the label $\uparrow$ to mark sections that are not written in the sheet music, e.g., cadenzas. Given the performance sequence π, the audio recording can be segmented into time intervals $I_1, \ldots, I_d$ such that time interval I_i corresponds to the section in the audio data where bar π_i is played (or something that is not written in the score in case $\pi_i = \uparrow$).

Given a query sequence of bars $Q = (q_0, \ldots, q_m)$, Q a substring of δ, the task of sheet music-audio identification is to find all time intervals T in the audio data where the query sequence of bars is played. More formally,

$$H(Q) := \{T \mid \exists j : Q = (\pi_j, \pi_{j+1}, \ldots, \pi_{j+m}) \wedge T = I_j \cup I_{j+1} \cup \ldots \cup I_{j+m}\}$$

denotes the set of *hits* w.r.t. Q. Note that in case of repeats that are notated as repeat signs, there can be more than one hit for a given query. Also note that besides the time intervals T there might be other time intervals in the audio data where the same musical content is played, but that belong to a different sequence of bars in the sheet music. We denote this kind of time intervals as *pseudo-hits*.

3 TWO APPROACHES

Given a scanned sheet music representation and an audio recording of the same piece of music, in a first step we use optical music recognition (OMR) software to extract information about musical symbols like staffs, bars and notes from the sheet music scans. Note that the obtained symbolic score data usually suffers from recognition errors. For simplicity, we here assume that the set of bar labels B and the default sequence δ are correctly obtained from the OMR output. Given a query $Q = (q_0, \ldots, q_m)$, which is a substring of δ, we want to find the set of hits $H(Q)$ as specified in Section 2. We now describe two approaches with different preconditions.

For the first approach, we assume that the performance sequence $\pi = (\pi_1, \ldots, \pi_d)$, $\pi_i \in B \cup \{\uparrow\}$, is known. In this case, we are left with the calculation of the corresponding time intervals $I_1, \ldots, I_d$. This can be done by using sheet music-audio synchronization. The set of hits $H(Q)$ can then be computed by finding occurrences of the query sequence in the performance sequence.

In the second approach, the performance sequence π is unknown. In this case, a reasonable strategy is to use sheet music-audio matching to search for sections in the audio recording with a similar musical content compared to the query sequence of bars. These sections may be considered as an approximation of the set of hits $H(Q)$. However, one should be aware of the fact that this method cannot distinguish correct hits from pseudo-hits, and is therefore expected to deliver false positives. In the following, we will refer to such false positives as *content-induced confusion*. Such confusion is also expected to be introduced by query sequences that differ only slightly, either in musical content or by a very small number of bars at the beginning or end of the sequence. This issue becomes particularly relevant, since the presence of OMR errors prohibits using too strict settings for rating similarity in the matching.

Due to the additional information π that is given in the first approach, this approach works much more robust and reliable than the second approach. The required performance sequence π can be created with little effort by manually editing an automatically generated list of jump directives acquired from the available default sequence δ. Therefore, we consider this approach semi-automatic. On the contrary, the second approach is fully automatic, but the results are less reliable. In the optimum case, only content-induced confusion would occur. In practice, however, extra confusion is likely to be introduced by shortcomings of the matching procedure.

The idea followed in this paper is to use the more reliable results of the semi-automatic first approach to create ground truth results for evaluating the less reliable fully automatic second approach. Using this method, we compare different settings of the matching procedure used in the second approach to learn which one works best for the task of sheet music-audio identification.

4 DATA PREPARATION

To compare sheet music data with audio data, we first create MIDI note events from the OMR results. However, OMR results often suffer from non-recognized or misclassified symbols. Especially in orchestral scores with many parts, erroneous or missing clefs and key signatures lead to wrong note pitches when creating MIDI events. Furthermore, orchestral scores can comprise parts for transposing instruments, i.e., the notated pitch is different from the sounding pitch. Such transposition information is not output by current OMR software, but it is essential for creating correctly pitched MIDI events. To be able to handle even complex orchestral scores, a so-called *staff signature* text file is generated from each page and is manually corrected. The staff signature file contains information about the clef, the key signature and the transposition at the beginning of each staff that is found on the page, see Figure 1. It also identifies which staffs belong to the same grand staff. The information from the staff signature files is used to correct errors in the OMR output and to add the missing information about transposing instruments.

There are several choices to be made regarding onset times and tempo, when creating the MIDI events from the OMR results. Since in the OMR output, notes or beams

Clef	Key Signature	Transposition
treble	+2	0
treble	+3	-7
treble	-1	-3
treble	+4	-2
tenor	+2	0
treble	0	-7
tenor	+2	0
alto	+2	0
bass	+2	0
bass	+2	0
bass	+2	0

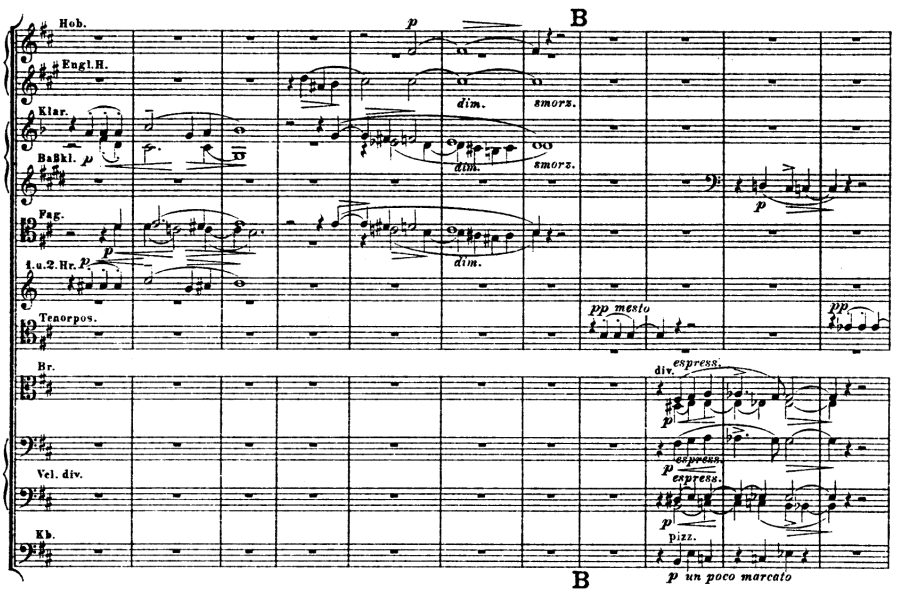

Figure 1. Staff signature annotation for an example grand staff taken from a score of the "Symphony to Dante's Divina Commedia S109 - Inferno" by Franz Liszt. Positive key signature values count the number of sharps, negative values count the number of flats. Transposition values are specified as the amount of semitones the pitch has to be modified with to sound correctly.

are often missed out, the accumulated note durations are not a good estimator for note onset times. This is especially the case for scores with multiple staffs and possibly multiple voices per staff, where the voice onset times might drift apart. Instead we use the horizontal position of notes within each measure as an estimator for the onset time. Even though this does not deliver onset times that perfectly match the musical meter, this method is very robust against surrounding errors and effectively inhibits voices from drifting apart.

Another parameter that is required to convert sheet music data to MIDI events is the tempo. This parameter is usually not output by OMR systems. If the performance sequence π is known in advance, the mean tempo can be calculated from the duration of the audio track. When π is not known, one might either use a fixed tempo or try to estimate a tempo based on the musical content. Note that the actual tempo used in audio interpretations can easily vary from 40 to 220 beats per minute (quarter notes per minute). We will investigate the effects of different tempo estimation strategies in our experiments in Section 8.

Both the MIDI data and the audio data are converted to sequences of normalized chroma-based features. Each feature is a 12-dimensional vector encoding the local energy distribution among the 12 traditional pitch classes of Western classical music commonly labeled C, C$^\sharp$, D, . . .,B.

5 SYNCHRONIZATION

After transforming both the MIDI data as well as the audio data into sequences of normalized chroma vectors, we use dynamic time warping (DTW) to synchronize the two sequences. Here, the main idea is to build up a cross-similarity matrix by computing the pairwise distance between each score chroma vector and each audio chroma vector. In our implementation, we simply use the inner vector product for the comparison. An optimum-cost alignment path is determined from this matrix via dynamic programming. To speed up this computationally expensive procedure, we use an efficient multiscale version of DTW.

6 MATCHING PROCEDURE

The task of the matching procedure is to find sections in the audio interpretation that are considered similar to a given query of score data. In this paper, we use a variant of the *subsequence dynamic time warping* algorithm for this task. For details we refer to the literature [2]. As in the case of synchronization, both the audio data and the score data are first converted to feature sequences. Each feature vector from the score query is compared to each feature vector from the audio database by means of a suitable *local cost measure*. The results of this comparison are stored in a cost matrix, see Figure 2. Finding candidate matches from this cost matrix means finding paths connecting the bottom row and the top row of the matrix. In particular, we are interested in paths p where the sum of the local cost of the matrix cells covered by the path is as small as possible. Such paths are calculated using dynamic programming by iteratively advancing from the bottom left towards the top right using a constrained set of allowed step directions ensuring that a path never runs backwards in time. For each matrix cell, the minimum cost of any valid path leading to that cell is saved in a so-called *accumulated cost matrix*. Matches are then identified by finding minima in the top row of the accumulated cost matrix.

Given a query bar sequence Q, the matching procedure outputs a set of matches $M(Q) = \{(p_1, c_1), \ldots, (p_N, c_N)\}$, where p_i is a path connecting the top and bottom rows and $c_i \in \mathbb{R}_{\geq 0}$ is the cost of

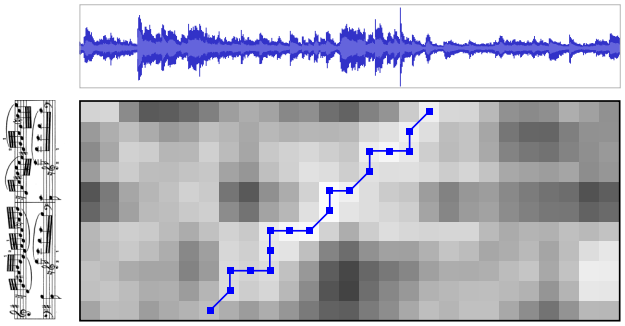

Figure 2. Illustration of the subsequence DTW cost matrix for a score query with a length of two measures accounting for 11 seconds of MIDI data (Beethoven Sonata 3, Opus 2 No 3, Adagio, measures 16–17). An excerpt of 27 seconds of audio data including one correct match is displayed. The optimum-cost path p for the correct match is rendered as a sequence of squares connected by lines.

the path p_i. The results are ranked with respect to the path cost. The choice of allowed step directions can be varied and associated step weights can be introduced to favor certain directions and behaviors. Several settings for step directions and step weights will be discussed in our experiments in Section 8.

7 EVALUATION PROCEDURE

Sheet music-audio matching depends on a multitude of parameters and settings used in the steps of creating MIDI events, creating feature sequences, and performing the matching procedure. In this work, we are interested in finding out which parameters work best for the task of sheet music-audio identification. We do this by evaluating and comparing several parameter sets on a test dataset consisting of a collection of musical *tracks*, with each track being represented by one sheet music representation and one audio interpretation.

In the evaluation, we perform the matching procedure on a set of test queries. For each test query Q, we then evaluate the matching results $M(Q)$ using a set of ground truth hits $H(Q)$ and a suitable confusion measure. To calculate the confusion measure, we first identify which matches output by the matching procedure correspond to ground truth hits. Let $T = [t_0, t_1] \in H(Q)$ be the ground truth hit and $(p, c) \in M(Q)$ be a match whose path p corresponds to the time interval $T' = [t'_0, t'_1]$ in the audio. The match (p, c) is then considered to correspond to the ground truth hit T, if both the durations and the locations roughly coincide. More precisely, with $\Delta := t_1 - t_0$ and $\Delta' := t'_1 - t'_0$ we require that

$$|\Delta' - \Delta| < 0.2\Delta \quad \text{and} \quad |t'_1 - t_1| < 0.2\Delta.$$

In the following, we call a match that corresponds to a ground truth hit a *correct match* and a match that does not correspond to a ground truth hit an *incorrect match*. Let $M(Q) = \{(p_1, c_1), \ldots, (p_N, c_N)\}$ be the set of all matches for a query Q, and let $C \subseteq [1 : N]$ be the set of indices of correct matches and $I \subseteq [1 : N]$ be the set of indices of incorrect matches. The confusion measure we use in this paper is a binary-valued function $\Gamma_{H,M}$ that on input Q takes the value 1 if at least one ground truth hit in $M(Q)$ has no corresponding match or if there is an incorrect match with lower cost than the highest-cost correct match, and 0 otherwise:

$$\Gamma_{H,M}(Q) := \begin{cases} 1 & \text{missed ground truth hit} \\ 1 & \min_{i \in I} c_i < \max_{i \in C} c_i \\ 0 & \text{otherwise.} \end{cases}$$

In other words, $\Gamma_{H,M}(Q) = 0$ if all ground truth hits are found and are ranked higher than any incorrect match. In case of $\Gamma_{H,M}(Q) = 1$ we also speak of *confusion*.

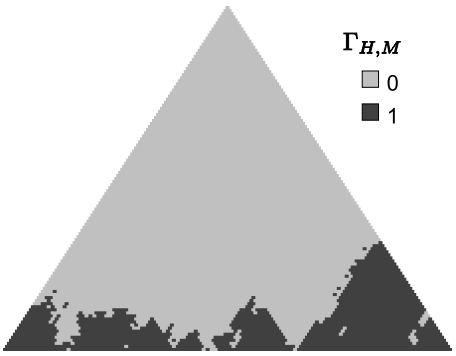

Figure 3. Scape plot for Beethoven's Piano Sonata no.7 op.10 no.3 Rondo (Allegro) using the confusion measure $\Gamma_{H,M}$.

Using the results of sheet music-audio synchronization that have been calculated in a preprocessing step, a set of ground truth hits can be calculated for any input query sequence of bars Q that is a substring of δ. This allows us to test each track using a grid of queries that covers not only the whole track but also a wide range of query lengths. The results can be nicely visualized in a so called *scape plot* [10]. Figure 3 shows a scape plot using the confusion measure $\Gamma_{H,M}$. Time runs from left to right. The lowest row shows the results for the shortest query length. The query length successively increases when moving upwards in the plot. The darker shaded areas indicate confusion.

From Figure 3, one can see that longer queries lead to less confusion and better separability of correct and incorrect matches. The plot also reveals where in the track and up to what query lengths the confusion happens. To not only be able to visually compare parameters for each individual track, but to also enable comparisons for the whole dataset, we summarize the results of all queries in one number per track by simply averaging over the complete grid of queries. Subsequently, we calculate the average over all tracks to end up with a single number for each set of parameters. If one parameter set works better than another parameter set, this fact should manifest in a lower average $\Gamma_{H,M}$ value. Note that one should not compare absolute values of the confusion measure for different tracks or datasets, because the absolute values depend on too many uncontrolled factors like the content-induced confusion, the tempo of the audio interpretation, and the content-dependent "uniqueness" of bars. Therefore, we keep datasets fixed, when studying the effects of using

Composer	Work	Instrumentation	#Pages	#Tracks	Duration
Beethoven	Piano Sonatas 1–15	Piano	278	54	5h 01min
Liszt	"A Symphony to Dante's Divina Commedia"	Symphonic Orchestra	145	2	44min
Mendelssohn	Concert in E minor, Op.64	Violin and Orchestra	55	3	26min
Mozart	String Quartetts 1–13	String Quartett	190	46	2h 46min
Schubert	"Die schöne Müllerin", "Winterreise" and "Schwanensang"	Singer and Piano	257	58	3h 04min

Table 1. Information and statistics on the test dataset used for evaluation.

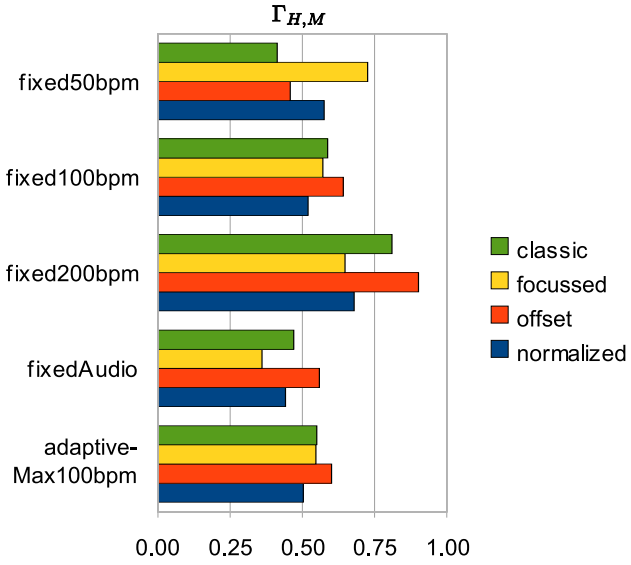

Figure 4. $\Gamma_{H,M}$ values averaged over the complete dataset for every combination of 5 tempo estimation strategies and 4 step direction and cost settings. Lower values are better.

different parameters by comparing the confusion measure values.

8 EXPERIMENTS AND RESULTS

Using the procedures described in the previous sections, there are many aspects whose effect on sheet music-audio identification should be investigated. Due to space limitation, we restrict ourselves to investigating the effects of different tempo estimation strategies in combination with different step settings and cost settings in the subsequence DTW. In particular, we test five tempo estimation strategies: **`fixedXXXbpm`**: Fixed tempo of `XXX` beats per minute, with `XXX` taking the values 50, 100 and 200. **`fixedAudio`**: Fixed mean tempo of the corresponding audio interpretation (estimated via manually annotated π and the duration of the audio file). **`adaptiveMax100bpm`**: The tempo is determined individually for each bar by taking into account the number of different onset times within the bar. The tempo is chosen such that the duration of the bar is 200ms times the number of different onset times. This leads to bars with runs of short-duration notes being slowed down compared to bars with long notes. Additionally, a maximum tempo of 100bpm is used to limit the difference between slow and

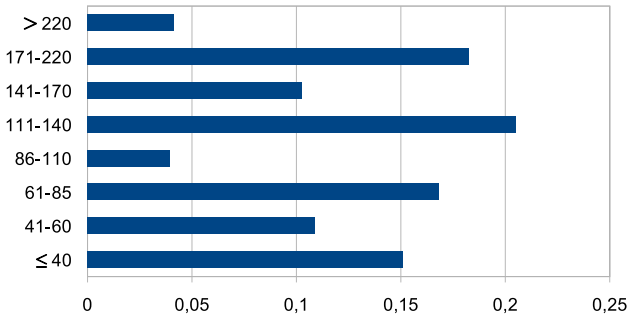

Figure 5. Tempo distribution of the test dataset being weighted the same way as the results in Figure 4

fast bars.

We use four different step and cost settings for the subsequence DTW. **`classic`**: Step vectors $(1,0),(0,1),(1,1)$ and cost weights $1,1,1$. **`focussed`**: Step vectors $(2,1),(1,2),(1,1)$ and cost weights $2,1,1$. **`offset`**: Same as `classic`, but with an additional cost offset of 1 which is added to each cell of the local cost matrix. **`normalized`**: The same as `classic`, but with an additional modification at the stage of calculating the accumulated cost matrix. At each matrix cell, the cost being compared for making the decision about which step vector leading to this cell delivers the minimum accumulated cost are normalized by the accumulated path length up to this cell. This normalization prevents short paths being preferred over long paths, even if the short paths have a higher average cost.

The dataset used for testing consists of 5 sheet music books covering a range of instrumentations and complexities, see Table 1. One audio inpterpretation per track is included. For each track in the dataset, we calculate the $\Gamma_{H,M}$ value for a grid of queries similar to the one used to create the scape plot in Figure 3. We start with a query length of 5 bars and use a hop size of 5 bars to move throughout the track. The query length is successively increased by 5 bars up to a maximum query length of 40 bars.

Figure 4 shows the results for testing all 20 combinations of settings on the test dataset. The $\Gamma_{H,M}$ values illustrated in the figure are average values calculated by first taking the average over all tracks within each scorebook, and then taking the average over all scorebooks. This way, each of the five different types of instrumentation and complexity gets the same weight. Since we are measuring effects that depend on the tempo, we also need to look at the

distribution of tempi of the tracks in the test dataset. Figure 5 shows the distribution of tempi being weighted the same way as the results in Figure 4 and confirms that there is no bias towards slower or higher tempi that might distort our results.

From the results in Figure 4 we can see that both the tempo estimation strategy and the tested step direction and cost settings clearly have an effect on the average amount of confusion. The best overall results are achieved by the setting `focussed` when using the mean tempo of the audio interpretation. This was expected, since this setting is more focussed towards the diagonal direction and, therefore, benefits the most from the fact that the tempo is known. However, in cases where the difference between the estimated tempo and the actual tempo of the interpretation becomes too large, the lack of flexibility leads to confusion, as can be seen for the tempo strategies `fixed50bpm` and `fixed200bpm`.

In the cases, where the tempo of the audio interpretation is assumed to be unknown, the best results are achieved by the setting `classic` using the `fixed50bpm` tempo estimation strategy. Both settings `classic` and `offset` work best when the estimated tempo is low. A possible explanation for this effect is that the accumulating cost lead to a preference of short paths. Shorter paths contain less steps and therefore accumulate less cost. When looking at the cost matrix depicted in Figure 2, one may think of the optimum-accumulated-cost paths tending to make shortcuts towards the top of the cost matrix instead of following the lane of minimum local cost. This effect leads to additional confusion when the estimated tempo of the sheet music data is high compared to the actual tempo of the audio interpretation.

The setting `normalized` delivers better results than the `classic` and `offset` settings for every tempo estimation strategy except for the `fixed50bpm`. For that strategy, however, it clearly falls behind and leads to even worse results than in the `fixed100bpm` case. A possible explanation is that, in contrast to the settings `classic` and `offset`, the setting `normalized` does not prefer shorter paths over longer paths. This seems to be an advantage when the estimated tempo is not too low, but in the `fixed50bpm` case, the lack of a driving force towards keeping the path connecting the bottom and top rows short causes paths to become much more sensitive to noise and local dissimilarities.

The `adaptiveMax100` yields only a tiny improvement over the `fixed100bpm` estimation. The reason for that probably is that the difference between the two strategies usually affects only the slower pieces. A test run using only the slower pieces might lead to a bigger advantage for the adaptive strategy.

9 CONCLUSIONS

We introduced and discussed the task of sheet music-audio identification, which is identifying sections of an audio recording where a given query sequence of bars from the sheet music is played. Two approaches to solving the task have been described, a semi-automatic approach using synchronization and a fully automatic approach using matching techniques. We proposed a workflow that allows for evaluating the matching approach by using results from the more reliable synchronization approach. This workflow includes contributions that make it possible to perform synchronization and matching even for complex orchestral scores. We introduced the idea of using scape plots to visualize results of matching or retrieval tasks that are performed on a grid of test queries covering a complete track of music over a wide range of query lengths. Finally, we performed an evaluation using a subsequence DTW based matching technique for the task of sheet music-audio identification. Results were presented and discussed for different sets of settings and tempo estimation strategies.

In our future work, we would like to investigate more aspects of sheet music-audio identification to answer questions like the following: Which features work best? What is the optimum feature resolution? Can the results be improved by using a harmonic model on the MIDI events created from the sheet music? What influence do OMR errors have on the results? Besides comparing the amount of confusion, we are also interested in comparing the temporal accuracy of matches.

10 ACKNOWLEDGEMENTS

We would like to express our thanks to the Bavarian State Library in Munich for their cooperation and for providing the sheet music scans.

11 REFERENCES

[1] C. Fremerey, M. Müller, F. Kurth, and M. Clausen: "Automatic Mapping of Scanned Sheet Music to Audio Recordings," *Proc. ISMIR, Philadelphia, USA*, pp. 413–418, 2008.

[2] M. Müller: *Information Retrieval for Music and Motion*, Springer, 2007.

[3] F. Soulez, X. Rodet, and D. Schwarz: "Improving Polyphonic and Poly-instrumental Music to Score Alignment," *Proc. ISMIR, Baltimore, USA*, pp. 143–148, 2003.

[4] N. Orio: "Alignment of Performances with Scores Aimed at Content-Based Music Access and Retrieval," *Proc. ECDL, Rome, Italy*, pp. 479–492, 2002.

[5] C. Raphael: "Aligning Music Audio with Symbolic Scores Using a Hybrid Graphical Model," *Machine Learning*, Vol. 65 No. 2–3 pp. 389–409, 2006.

[6] R.B. Dannenberg and C. Raphael: "Music Score Alignment and Computer Accompaniment," *Communications of the ACM*, Vol. 49 No. 8 pp. 38–43, 2006.

[7] R.J. Turetsky and D.P.W. Ellis: "Ground-Truth Transcriptions of Real Music from Force-Aligned MIDI Syntheses," *Proc. ISMIR, Baltimore, USA*, pp. 135–141, 2004.

[8] J. Pickens, J.P. Bello, G. Monti, T. Crawford, M. Dovey, and M. Sandler: "Polyphonic Score Retrieval Using Polyphonic Audio Queries: A Harmonic Modeling Approach," *Proc. ISMIR, Paris, France*, pp. 140–149, 2002.

[9] I.S.H. Suyoto, A.L. Uitdenbogerd, and F. Scholer: "Searching Musical Audio Using Symbolic Queries," *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 16 No. 2 pp. 372–381, 2008.

[10] C. Sapp: "Comparative Analysis of Multiple Musical Performances," *Proc. ISMIR, Philadelphia, USA*, pp. 497–500, 2008.