MIDI–SHEET MUSIC ALIGNMENT USING BOOTLEG SCORE SYNTHESIS

Thitaree Tanprasert[†] *Teerapat Jenrungrot*[†] *Meinard Müller*[⋆] *TJ Tsai*[†]

[†] Harvey Mudd College, Claremont, CA USA
[⋆] International Audio Laboratories Erlangen, Germany

ABSTRACT

MIDI–sheet music alignment is the task of finding correspondences between a MIDI representation of a piece and its corresponding sheet music images. Rather than using optical music recognition to bridge the gap between sheet music and MIDI, we explore an alternative approach: projecting the MIDI data into pixel space and performing alignment in the image domain. Our method converts the MIDI data into a crude representation of the score that only contains rectangular floating notehead blobs, a process we call bootleg score synthesis. Furthermore, we project sheet music images into the same bootleg space by applying a deep watershed notehead detector and filling in the bounding boxes around each detected notehead. Finally, we align the bootleg representations using a simple variant of dynamic time warping. On a dataset of 68 real scanned piano scores from IMSLP and corresponding MIDI performances, our method achieves a 97.3% accuracy at an error tolerance of one second, outperforming several baseline systems that employ optical music recognition.

Index Terms— alignment, synchronization, MIDI, sheet music

1. INTRODUCTION

This paper tackles the problem of MIDI–sheet music synchronization. Given a symbolic music representation and its scanned sheet music, the objective is to determine the alignment between each time instant in the symbolic representation and its corresponding pixel location in the sheet music.

Many tools for alignment have been developed in the context of audio synchronization. The goal of audio synchronization is to find the temporal alignment between two different audio recordings of the same musical piece. The main technique used to solve this alignment problem is dynamic time warping (DTW) [1, 2]. Many works have proposed ways to extend or improve upon this basic method, including doing the time warping in an online fashion [3, 4], estimating the alignment at multiple granularities [5, 6], handling jumps or partial alignments [7, 8], dealing with fixed memory constraints [9], and taking advantage of multiple recordings [10, 11].

Several previous works have studied the problem of finding correspondences between audio and sheet music. There are two general approaches to the problem. The first approach is to use an existing optical music recognition (OMR) system to convert the sheet music into a MIDI representation, and then to compare the MIDI and audio using some type of chroma representation. This approach has been applied to synchronizing audio and sheet music [12, 13] and other related problems [14, 15]. A different approach has been explored in recent years: convolutional neural networks (CNNs). This approach attempts to learn a multimodal CNN that can embed a short segment of sheet music and a short segment of audio into the same feature space, where similarity can be computed using a simple distance measure (e.g. Euclidean distance). This approach has been explored in the context of sheet music score following and related problems [16, 17, 18]. Dorfer et al. [19] have recently shown promising results formulating the score following problem as a reinforcement learning game.

While audio–sheet music synchronization has been studied in various forms, we are not aware of any previous works that directly study the MIDI–sheet music synchronization problem. As symbolic data often serves as a bridge between audio and sheet music, we believe its study can lead to novel ideas in cross-modal alignment.

This paper proposes a method for aligning a symbolic music representation and its corresponding sheet music. Previous works on cross-modal alignment have focused either on using OMR to bridge the modality gap or indirectly learning a representation in a shared but unknown feature space. In contrast, our approach is to work with an explicitly known sparse, binary representation in the image domain. As we will demonstrate, we can convert both sheet image data and MIDI data into this representation using simple logic coupled with a notehead detector. Based on this common binary representation, we show how the alignment problem can then be solved using a simple variant of DTW.

The paper is organized as follows. Section 2 describes the proposed algorithm. Section 3 explains the experimental setup. Section 4 presents the results. Section 5 concludes the work.

2. SYSTEM DESCRIPTION

There are two inputs to our system: a MIDI performance and its corresponding sheet music. Similar to other recent works [20, 17, 19], we assume that the sheet music is presented as a sequence of image strips, where each image strip contains a single line of music. We focus exclusively on piano music in this work, so each line of music consists of a single grand staff containing an upper staff (treble clef) and a lower staff (bass clef). The image strips may have different dimensions, and the staff lines may appear at different locations on different strips.

Our proposed method has three main steps. The first step is to convert each image strip into a sparse, binary representation in pixel space (A_i in Figure 1). We perform this conversion by applying a notehead detector and filling in the predicted bounding boxes around each detected notehead. This representation is a very crude representation of the score that only contains rectangular floating notehead blobs. Accordingly, we call this a bootleg representation. The second step is to project the MIDI data into the same bootleg space (B_i in Figure 1). We perform this projection by converting MIDI note onsets into floating notehead blobs that are appropriately placed in pixel space. The third step is to align the bootleg representations with one another using a variant of DTW (Figure 3). Each of these three steps will be described in detail in the next three subsections.

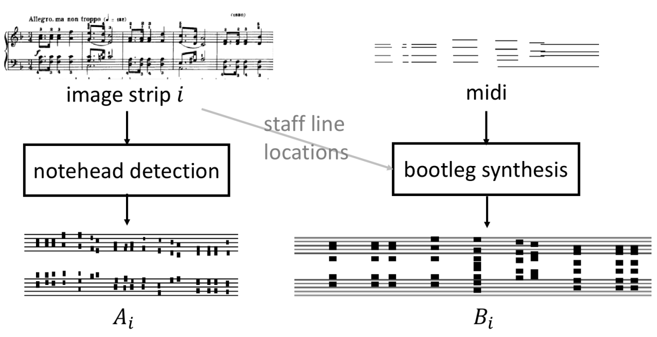

Fig. 1. Projecting data to bootleg space. We convert the image strips and MIDI data into a very crude approximation of the sheet music that only contains floating notehead blobs. The staff lines in A_i and B_i are shown as a visual aid, but are not included in the bootleg representation.

2.1. Notehead Detection

In this section we describe our notehead detector, which is based on the deep watershed detector recently proposed by Tuggener et al. [21] for musical object detection in sheet music. The deep watershed detector is a fully convolutional network [22] modified to predict three outputs: (a) a quantized energy output map which indicates the likelihood of having an object at each pixel location, (b) a class output map which predicts the type of object present at each pixel location (e.g. filled notehead, staff line, treble clef, sharp, quarter rest, etc.), and (c) a bounding box output map which indicates the width and height of an object at that pixel location. Figure 2 illustrates the overall architecture of the deep notehead detection network. The reader is referred to [21] for more details. In [23] and [21], Tuggener et al. show that fully convolutional networks are more suitable for semantic segmentation and detection of tiny objects in sheet music, tasks where (large) object detection methods like Fast R-CNN [24], Faster R-CNN [25], and YOLO [26] fail miserably.

We trained our network on the DeepScores dataset [23]. This dataset contains approximately 300,000 full pages of synthetically generated musical scores and pixel-level ground truth labels for 124 different symbol classes. The inputs to the network are 500x500 grayscale image patches that are randomly sampled from the full page images. The loss function is a linear combination of the losses from the quantized energy output map (cross entropy loss), class output map (cross entropy loss), and bounding box output map (mean squared error).

After training on DeepScores, we fine-tune the network on real scanned sheet music. For fine-tuning, we manually annotated the location and type of approximately 2200 noteheads in 30 different pages of piano music downloaded from IMSLP.[1] These 30 pages of music were selected to maximize diversity across composers and music publishers, and they are a completely separate set from the data used to evaluate alignment. Because we only care about detecting noteheads, we disregard all other musical objects in the fine-tuning process. Because the real scanned music contains a variety of font sizes, we scale each input image to match the staff line spacing in the DeepScores data. After fine-tuning, the notehead detector achieves a training mean average precision (mAP) of 0.4201 for all notehead types (black notehead, half notehead, and whole notehead). For reference, in normal object detection tasks (not tiny objects),

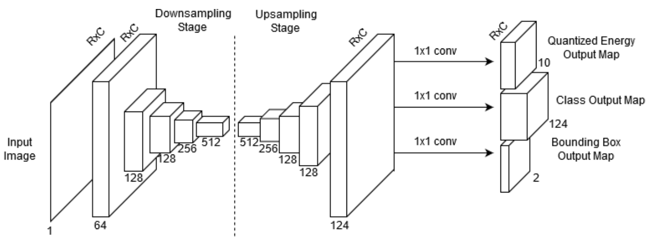

Fig. 2. Architecture of the deep watershed notehead detector. The number below each layer indicates the number of feature maps. In the downsampling and upsampling stages, the length and width of the feature maps change by a factor of two in each successive layer. We train on the DeepScores dataset [23] and fine-tune on a small set of manually labeled noteheads in real scanned music.

the state-of-the-art mAP is around 0.4 to 0.6.[2] Figure 5 (top half) shows an example of the notehead detector predictions on a section of Brahms Intermezzo Op. 117 No. 2.

2.2. Bootleg Synthesis

The second step is to convert the MIDI data into a bootleg representation. As shown in Figure 1 (right side), we accomplish this by converting note onsets into appropriately placed floating notehead blobs. This process consists of three key parts.

The first part is determining the staff line coordinate system. For each image strip, we would like to determine the location of the staff lines in the upper staff and lower staff. We can accomplish this by computing the row sum of image pixels, convolving the result with comb filters of various sizes (each containing 5 regularly spaced impulses), and identifying the comb filter that yields the strongest response at two non-overlapping staff locations. This gives us the staff line coordinate system for the upper and lower staves.

The second part is synthesizing and placing noteheads. Given the coordinate system from an image strip, we convert each MIDI note onset into one or more floating rectangular noteheads. Note that there is ambiguity when converting from a MIDI note number to a location on a staff. For example, a MIDI note number of 68 might appear as a G-sharp or an A-flat, which correspond to two different staff locations. To handle this ambiguity, we can place a larger-than-normal rectangular notehead which overlaps both possible locations. Furthermore, since notes in the middle register could appear in the right hand or left hand staves, we can simply place two different floating noteheads at both possible locations. In the visualization of B_i in Figure 1, for example, you can see that the first chord contains a C4, which produces a notehead in both the upper and lower staves.

The third part is handling timing issues. One issue is the choice of sampling rate. When converting the MIDI data into the bootleg representation, we must choose how much time corresponds to a single pixel column. To avoid extreme time warping, we select this parameter to ensure that the bootleg representation is approximately the same length as the image strips concatenated end-to-end. Another issue is shortening long pauses. When there is a fermata in the score, for example, the MIDI performance may slow down in tempo by a factor of three or four. The sheet music, however, does not reflect this, i.e. the length of the measure in pixels is not elongated by a factor of three or four. To mitigate this issue, we simply shorten long gaps greater than a fixed threshold to the length of the thresh-

[1]https://imslp.org

[2]http://cocodataset.org/#detection-leaderboard

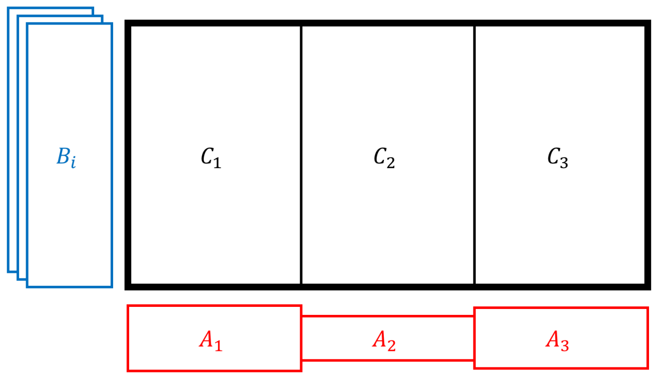

Fig. 3. Aligning MIDI-generated bootleg scores with the sheet image-generated bootleg strips. Each bootleg strip A_i is compared to its corresponding bootleg score B_i to yield a cost matrix block C_i. Note that each B_i is a MIDI-generated bootleg score of the entire piece projected onto the staff line coordinate system of strip A_i.

old. In experiments, we find that system performance is relatively insensitive to this threshold (across an order of magnitude).

Given the staff line locations, this process generates a bootleg score representation of the entire MIDI performance (B_i in Figure 1). Because each sheet image strip i has a different size and a different staff line coordinate system, we generate one bootleg score B_i for each image strip i. In other words, B_i is the bootleg score representation of the MIDI data projected onto the staff line coordinate system of image strip i. Note that all of the bootleg score representations will have the same length (i.e. number of pixel columns), but each B_i will have a unique height that matches the height of image strip A_i.

2.3. Block DTW

The third step is to align the bootleg representations. Figure 3 shows a graphical depiction of this process. In this example, the sheet music contains three image strips A_1, A_2, and A_3. We first calculate the cost matrix (C_i) between each bootleg image strip (A_i) and its corresponding MIDI-generated bootleg score (B_i). Because A_i and B_i share the same staff line coordinate system, we can compute their similarity with a simple negative inner product. The $(k, l)^{th}$ element of C_i thus indicates the number of overlapping black pixels in the k^{th} pixel column of B_i and the l^{th} pixel column of A_i. We assemble the constituent cost matrices C_i into a single global cost matrix (represented as a bold black rectangle in Figure 3), and then apply DTW with step transitions $\{(1, 1), (1, 2), (2, 1)\}$ and corresponding weights $\{2, 3, 3\}$. The lowest cost path through the global cost matrix is the estimated alignment between the MIDI performance and the sheet music.

3. EXPERIMENTAL SETUP

The experimental setup will be described in three parts: the data, the annotations, and the evaluation.

The data consists of sheet music and MIDI for 22 compositions from 8 different composers. The pieces are all for solo piano, contain no repeats or structural jumps, and span a variety of eras, styles, and lengths. The sheet music is downloaded from IMSLP and contains digital scans of real published musical scores in the public domain. In total, there are 68 sheet music scores. For each composition, we also collected one MIDI performance from online websites.[3] The MIDI performances are symbolic score representations that have been modified to sound like expressive, realistic human performances. Table 1 summarizes the dataset.

Piece	Sheet	Meas	Strips
Brahms Fantasia Op. 117 No. 2	4	86	(20,25)
Brahms Fantasia Op. 116 No. 6	3	64	(12,15)
Chopin Mazurka Op. 30 No. 2	6	64	(9,12)
Chopin Mazurka Op. 63 No. 3	6	76	(10,12)
Chopin Mazurka Op. 68 No. 3	6	60	(8,12)
Clementi Sonata Op. 36 No. 1 mv 3	2	70	(8,8)
Clementi Sonata Op. 36 No. 2 mv 3	2	111	(14,14)
Clementi Sonata Op. 36 No. 3 mv 3	2	82	(11,11)
Debussy Children's Corner mv 1	3	76	(24,25)
Debussy Children's Corner mv 3	3	124	(23,29)
Debussy Children's Corner mv 6	3	128	(25,25)
Mendelssohn Op. 19 No. 2	5	91	(12,14)
Mendelssohn Op. 62 No. 3	3	48	(8,10)
Mendelssohn Op. 62 No. 5	3	59	(12,13)
Mozart Sonata No. 13 mv 3	4	225	(42,50)
Mozart Sonata No. 9 mv 3	3	269	(50,60)
Schubert Impromptu Op. 90 No. 1	2	204	(41,60)
Schubert Impromptu Op. 90 No. 3	2	86	(42,42)
Schubert Op. 94 No. 2	2	92	(17,20)
Tchaikovsky The Seasons - January	2	102	(29,29)
Tchaikovsky The Seasons - June	2	99	(38,40)
Tchaikovsky The Seasons - August	2	198	(24,24)

Table 1. Summary of dataset. For each piece, the table indicates the number of sheet music versions, the number of measures, and the minimum & maximum number of image strips (i.e. lines of music) across the different sheet music versions.

The ground truth consists of beat-level annotations. For the sheet music, we annotated the horizontal pixel location of a subset of beats in the piece, along with the measure number and image strip number. Because pixel-level annotation of beat locations is very time-consuming, we annotated the beats in $N = 40$ measures equally spaced throughout the piece. For the MIDI performances, we estimate the ground truth beat locations using `pretty-midi`[4] and manually correct any errors.

To evaluate the system performance, we compare the predicted alignment to the ground truth annotations. At the ground truth beat locations in the sheet music, we compare the predicted corresponding times in the MIDI to the ground truth timestamps. Given a fixed error tolerance, we define the error rate to be the percentage of predictions that fall outside of the allowable error tolerance. By considering a range of error tolerances, we can characterize the tradeoff between error rate and error tolerance.

In total, there are 68 MIDI-score pairings, and the resulting alignments are evaluated at $10,913$ ground truth beat locations. Our choice to use scans of real published musical scores has a tradeoff: it places a constraint on the size of the dataset due to the time-consuming nature of annotation, but it is more representative of performance "in the wild" compared to large synthetic datasets like the Multimodal Sheet Music Dataset (MSMD) [18]. We plan to also evaluate our system on the MSMD in future work.

[3] `www.piano-midi.de` and `www.mazurka.org.uk`
[4] `https://github.com/craffel/pretty-midi`

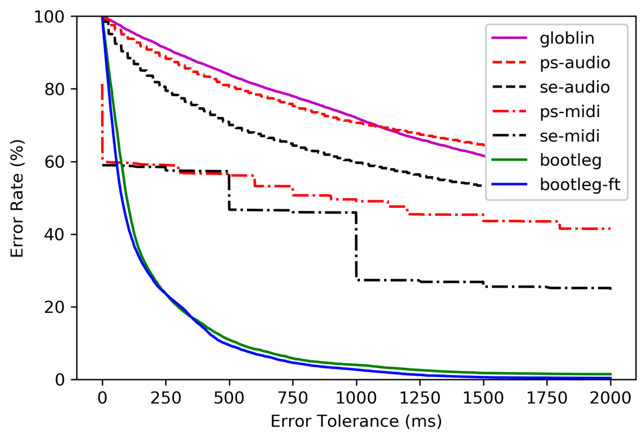

Fig. 4. Comparison of baselines to the bootleg system with and without fine-tuning. The legend lists the systems in order of performance from worst to best. For a description of the baseline systems, see Section 4.

4. RESULTS

We compared our bootleg method (with and without fine-tuning) to five baseline systems. The first baseline system ('globlin') simply assumes a global linear correspondence between the concatenated sheet image strips and the MIDI performance. The second ('ps-audio') and third ('se-audio') baseline systems use two different commercial OMR systems (Photoscore and SharpEye) to convert the sheet music to MIDI, synthesize the MIDI to audio, and then perform audio–audio alignment using DTW on chroma features.[5] The fourth ('ps-midi') and fifth ('se-midi') baseline systems use the same two OMR systems to convert the sheet music to MIDI, estimate the beat locations using `pretty-midi`, and then assume a 1-to-1 correspondence between beat locations in both MIDI files. Note that the OMR system can have many recognition errors but still have perfect alignment if it can simply interpret barlines and beats correctly.[6]

There is one important issue to mention about evaluating the OMR baseline systems. Because PhotoScore and SharpEye do not retain the connection between sheet music pixel location and corresponding MIDI time, there is no reliable way to automatically infer ground truth beat locations in the OMR-generated MIDI. Thus, it was necessary to manually annotate the ground truth beat locations in the OMR-generated MIDI on every sheet music score, so that the predicted alignment can be evaluated. This is a very time-consuming process, and clearly not sustainable for large-scale evaluation. However, the benefit of these results is a fair comparison to commercial OMR systems over a reasonably diverse data set.

There are three things to note about Figure 4. First, the bootleg method ('bootleg') outperforms the baselines by a wide margin. For example, at 1000 ms error tolerance the bootleg systems achieve $3-4\%$ error rate, whereas the best baseline system has 27% error rate. Second, fine-tuning ('bootleg-ft') shows demonstrable improvement. For example, at 1000 ms error tolerance the fine-tuning improves the error rate from 4.0% to 2.7%. The key observation here is that we can achieve observable improvement even with an extremely small set of data for fine-tuning (2200 noteheads). Third, the bootleg systems have low asymptotic error rates. Whereas the best baseline system levels off at 25% error, the fine-tuned bootleg system achieves a 0.4% error rate at 2000 ms error tolerance.

[5]We also experimented with doing DTW directly on a piano roll representation of the MIDI data, but found that the results were always worse than synthesizing to audio and aligning chroma features.

[6]Note that the stairstep shape of the 'se-midi' system comes from the fact that SharpEye always renders its OMR-generated MIDI at 120 bpm, so that missing or extra beats correspond to errors at integer multiples of 500 ms.

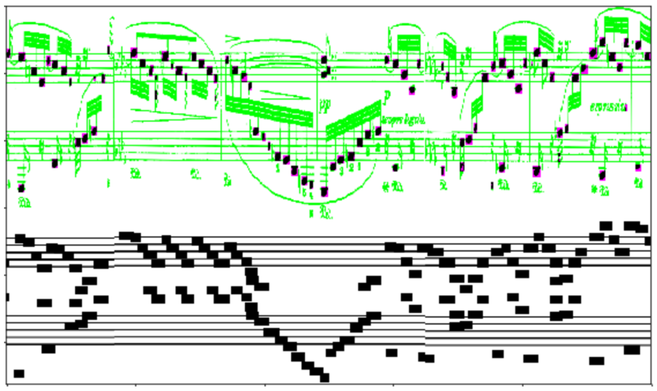

Fig. 5. An example of the predicted alignment produced by the bootleg system. The upper half of the figure shows the original score with the detected noteheads overlaid. The bottom half of the figure shows the aligned MIDI-generated bootleg score. This figure is best visualized in color.

Figure 5 shows a visualization of the predicted alignment for a section of Brahms Intermezzo Op. 117 No. 2. In the upper half, the detected notehead regions are overlaid on top of the original score for ease of visualization. The bottom half contains the aligned MIDI-generated bootleg score. We can see that most noteheads are correctly detected, and the two bootleg representations match well.

By looking at example visualizations, we discovered two weaknesses in our system. The first weakness is that the notehead detector performs poorly on half noteheads and whole noteheads. This is due to the fact that the training data was highly imbalanced towards filled noteheads. The second weakness is that the staff line detection occasionally fails, which leads to poor alignments on the entire strip. Interestingly, the system is fairly robust to clef changes, mainly because clef changes usually only occur in one staff but not both.

5. CONCLUSION

We approach the MIDI–sheet music alignment problem by using an explicit mid-level feature representation called a bootleg score. The bootleg score is a crude representation of the music which only contains rectangular floating notehead blobs. We can project sheet image strips to bootleg space by applying a deep watershed notehead detector and filling in the bounding boxes of detected noteheads. Similarly, we can project the MIDI data to bootleg space by converting MIDI note onsets to appropriately placed rectangular noteheads. Once the MIDI and sheet music have been projected to this bootleg representation, the alignment can be performed using a simple variant of DTW. We evaluate the proposed system on a dataset of 68 real scanned piano scores. Our results indicate that the proposed approach works well for piano music, and it outperforms several baseline systems based on optical music recognition. Future work includes improving notehead detection, expanding to other types of music (e.g. string quartet), and extending this approach to audio–sheet music synchronization.

6. REFERENCES

[1] Roger B. Dannenberg and Ning Hu, "Polyphonic audio matching for score following and intelligent audio editors," in *Proc. of the International Computer Music Conference (ICMC)*, San Francisco, USA, 2003, pp. 27–34.

[2] Sebastian Ewert, Meinard Müller, and Peter Grosche, "High resolution audio synchronization using chroma onset features," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Taipei, Taiwan, Apr. 2009, pp. 1869–1872.

[3] Simon Dixon, "Live tracking of musical performances using on-line time warping," in *Proc. of the 8th International Conference on Digital Audio Effects*. Citeseer, 2005, pp. 92–97.

[4] Robert Macrae and Simon Dixon, "Accurate real-time windowed time warping," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2010, pp. 423–428.

[5] Meinard Müller, Henning Mattes, and Frank Kurth, "An efficient multiscale approach to audio synchronization," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, Victoria, Canada, Oct. 2006, pp. 192–197.

[6] Stan Salvador and Philip Chan, "FastDTW: Toward accurate dynamic time warping in linear time and space," in *Proc. of the KDD Workshop on Mining Temporal and Sequential Data*, 2004.

[7] Christian Fremerey, Meinard Müller, and Michael Clausen, "Handling repeats and jumps in score-performance synchronization," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, Utrecht, The Netherlands, 2010, pp. 243–248.

[8] Meinard Müller and Daniel Appelt, "Path-constrained partial music synchronization," in *Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Las Vegas, Nevada, USA, Apr. 2008, vol. 1, pp. 65–68.

[9] Thomas Prätzlich, Jonathan Driedger, and Meinard Müller, "Memory-restricted multiscale dynamic time warping," in *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Shanghai, China, 2016, pp. 569–573.

[10] Andreas Arzt and Gerhard Widmer, "Real-time music tracking using multiple performances as a reference," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, Málaga, Spain, 2015, pp. 357–363.

[11] Siying Wang, Sebastian Ewert, and Simon Dixon, "Robust and efficient joint alignment of multiple musical performances," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 11, pp. 2132–2145, 2016.

[12] Frank Kurth, Meinard Müller, Christian Fremerey, Yoon-Ha Chang, and Michael Clausen, "Automated synchronization of scanned sheet music with audio recordings," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, Vienna, Austria, Sept. 2007, pp. 261–266.

[13] Verena Thomas, Christian Fremerey, Meinard Müller, and Michael Clausen, "Linking sheet music and audio – challenges and new approaches," in *Multimodal Music Processing*, Meinard Müller, Masataka Goto, and Markus Schedl, Eds., vol. 3 of *Dagstuhl Follow-Ups*, pp. 1–22. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2012.

[14] Christian Fremerey, Meinard Müller, Frank Kurth, and Michael Clausen, "Automatic mapping of scanned sheet music to audio recordings," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, Philadelphia, USA, Sept. 2008, pp. 413–418.

[15] Christian Fremerey, Michael Clausen, Sebastian Ewert, and Meinard Müller, "Sheet music-audio identification," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, Kobe, Japan, Oct. 2009, pp. 645–650.

[16] Matthias Dorfer, Andreas Arzt, Sebastian Böck, Amaury Durand, and Gerhard Widmer, "Live score following on sheet music images," in *Late Breaking Demo at the International Conference on Music Information Retrieval (ISMIR)*, 2016.

[17] Matthias Dorfer, Andreas Arzt, and Gerhard Widmer, "Learning audio-sheet music correspondences for score identification and offline alignment," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, Suzhou, China, 2017, pp. 115–122.

[18] Matthias Dorfer, Jan Hajič Jr., Andreas Arzt, Harald Frostel, and Gerhard Widmer, "Learning audio-sheet music correspondences for cross-modal retrieval and piece identification," *Transactions of the International Society for Music Information Retrieval*, vol. 1, no. 1, pp. 22–33, 2018.

[19] Matthias Dorfer, Florian Henkel, and Gerhard Widmer, "Learning to listen, read, and follow: Score following as a reinforcement learning game," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2018, pp. 784–791.

[20] Matthias Dorfer, Andreas Arzt, and Gerhard Widmer, "Towards score following in sheet music images," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, New York City, New York, USA, 2016, pp. 789–795.

[21] Lukas Tuggener, Ismail Elezi, Jürgen Schmidhuber, and Thilo Stadelmann, "Deep watershed detector for music object recognition," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2018, pp. 271–278.

[22] Jonathan Long, Evan Shelhamer, and Trevor Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.

[23] Lukas Tuggener, Ismail Elezi, Jürgen Schmidhuber, Marcello Pelillo, and Thilo Stadelmann, "DeepScores – a dataset for segmentation, detection and classification of tiny objects," in *Proc. of the IEEE International Conference on Pattern Recognition*, 2018.

[24] Ross Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 91–99.

[26] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.