AUDIO-TO-SCORE ALIGNMENT AT NOTE LEVEL FOR ORCHESTRAL RECORDINGS

Marius Miron, Julio José Carabias-Orti, Jordi Janer
Music Technology Group, Universitat Pompeu Fabra
marius.miron,julio.carabias,jordi.janer@upf.edu

ABSTRACT

In this paper we propose an offline method for refining audio-to-score alignment at the note level in the context of orchestral recordings. State-of-the-art score alignment systems estimate note onsets with a low time resolution, and without detecting note offsets. For applications such as score-informed source separation we need a precise alignment at note level. Thus, we propose a novel method that refines alignment by determining the note onsets and offsets in complex orchestral mixtures by combining audio and image processing techniques. First, we introduce a note-wise pitch salience function that weighs the harmonic contribution according to the notes present in the score. Second, we perform image binarization and blob detection based on connectivity rules. Then, we pick the best combination of blobs, using dynamic programming. We finally obtain onset and offset times from the boundaries of the most salient blob. We evaluate our method on a dataset of Bach chorales, showing that the proposed approach can accurately estimate note onsets and offsets.

1. INTRODUCTION

Audio-to-score alignment concerns synchronizing the notes in a musical score with the corresponding audio rendition. An additional step, alignment at the note level, aims at adjusting the note onsets, in order to further minimize the error between the score and audio. In the context of orchestral music, this task is challenging; first, because of the complex polyphonies, and, second, because of the timing expressivity of classical music.

As possible applications of note alignment, deriving the exact locations of the note onsets and offsets could improve tasks as score-informed source separation [6], [2], [7].

State-of-the-art score alignment methods use Nonnegative matrix factorization (NMF) [14], [11], template adaptation through expectation maximization [9], dynamic time warping (DTW) [3], and Hidden Markov Models (HMM) [4, 6]. The method described in [11, p. 103] is the only one addressing explicitly the topic of fine note alignment as a post-processing step. A factorization is performed to obtain the onsets of the anchor notes. The basis vectors are trained with piano pitches models, and the onsets are obtained from the activations matrix. Furthermore, an additional step is performed in order to look for onsets between anchors.

However, the methods listed above have certain limitations. First, accurately detecting the offset of the note is a challenging problem and none of these methods claim to solve it. Second, the scope of the NMF-based systems is solely piano recordings. Third, except [11], the algorithms consider a large window to evaluate detected onsets. Note that the MIREX Real-time Audio-to-Score Alignment task considers a 2000 ms window size.

With respect to image processing techniques deployed in music information research, a system to link audio and scores for makam music is presented in [13]. In this case, Hough transform is used for picking the line corresponding to the most likely path from a binarized distance matrix. Additionally, the same transform is used in [1] to find repeating patterns for audio thumbnailing.

In this paper we propose a novel method for audio-to-score alignment at the note level, which combines audio and image processing techniques. In comparison to classical audio-to-score alignment methods, we aim to detect the offset of the note, along with its onset. Additionally, we do not assume a constant delay between score and audio, thus we do not use any information regarding the beats, tempo or note duration, in order to adjust the onsets. Therefore, our method can align notes when dealing with variable delays, as the ones resulting from automatic score alignment or the ones yielded by manually aligning the score at the beat level.

The proposed method is based on two stages. First, the audio processing stage involves filtering the spectral peaks in time and frequency for every note. Consequently, the filtering occurs in the time interval restricted for each note and in the frequency bands of the harmonic partials corresponding to its fundamental frequency. Furthermore, we decrease the magnitudes of the peaks which are overlapping in time and frequency with the peaks from other notes. Using the filtered spectral peaks, we compute the pitch salience for each note using the harmonic summation algorithm described in [10]. Second, we detect the boundaries of the note using an image processing algorithm. The pitch salience matrix associated to each note is binarized. Then, blobs, namely boundaries and shapes, are detected using

the connectivity rules described in [12, p. 248]. From all the blobs candidates associated to every note, we pick the best combination of consecutive blobs using dynamic programming. The image processing part has the advantage that the blob boundaries will define the note onsets along with the corresponding offsets.

The remainder of this paper is structured as follows. In the first section we describe the note-wise pitch salience computation, followed by the blob selection using image processing methods. Then, we evaluate our algorithm on a dataset of Bach chorales [6] and we discuss the results.

2. METHOD

The proposed method aims to detect the onsets and offsets of the notes from a monaural audio recording, where the score is assumed to be automatically or manually aligned a priori, assuming an error up to 200 ms.

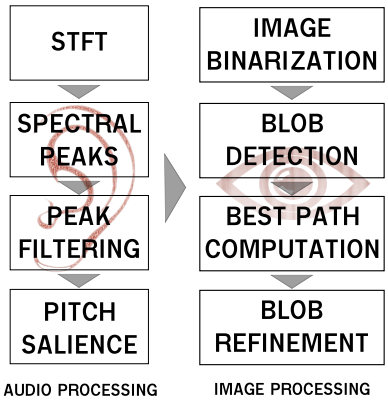

Figure 1. *The two main sections of our method: audio and image processing, and the corresponding steps.*

Figure 2 shows the block diagram of the proposed method. As can be seen, the method is subdivided in two stages. First, in the audio processing stage, a filtered pitch salience matrix is obtained for each of the notes in the score, and for every instrument. Second, in the image processing stage, the pitch salience matrix is regarded as a greyscale image, and blobs are detected in the binarized image. Moreover, we construct a graph with all the blobs and we pick the best combination of blobs by using Dijkstra's algorithm to find the best path in the graph. Finally, we refine the time boundaries for the blobs that overlap, using an adaptive threshold binarization.

2.1 Note-wise pitch salience computation

For each input signal, we first compute the Short time Fourier transform (STFT) and we extract the spectral peaks. Then, we analyze each single note in the score and we select only the spectral peaks in the frames around its approximate time location and the frequency bands associated to its harmonic partials (i.e. multiples of the fundamental frequency). Finally, we compute the pitch salience, using the harmonic summation algorithm described in [10].

To select the time intervals at which we are going to look for the note onsets and offsets, we analyze the pre-aligned score that we want to refine. We start from the assumption that the note onsets are played with an error lower than 200 ms from the actual onset in the score. In other words, we set the search interval to ± 200 ms from the note onset at the score. Additionally, in the case of the offset, we extend the possible duration of a note in the score by 200 ms or until another note in the score appears. In the rest of the paper, this search interval will be referred to as $T_{on}(n)$ and $T_{off}(n)$.

Then, we analyze the spectral peaks within the time interval defined for each note, and we filter them according to the harmonic frequencies of the MIDI note $\hat{F}_n(i)$, where $\hat{F}_n(0)$ is the fundamental frequency of note n. Namely, we take the first 16 of the harmonic partials of this frequency, $\hat{F}_n(i)$ with $i \in [0, ..., 15]$. Taking into account vibratos, we set a 1.4 semitone interval around each of the harmonic partials. Consequently, we select a set of candidate peaks $P_n(k)$ and the associated amplitudes $A_n(k)$ for note n at frame k such that $P_n(k) \in [\hat{F}_n(i) - \hat{L}_n(i), ..., \hat{F}_n(i) + \hat{L}_n(i)]$, where $\hat{L}_n(i)$ is a frequency band equivalent to 0.7 of a semitone.

As a drawback, some of the selected peaks could overlap in time and frequency. To overcome this problem, we distribute the amplitude $A_n(k)$ of the overlapped peaks $P_n(k)$ using a factor $g_i(P_n(k), P_m(k))$, where n and m are the overlapped notes, g_i is a gaussian centered at the corresponding frequency $\hat{F}_n(i)$ of the note n and the harmonic partial i. The standard deviation equals to $\frac{\hat{L}_n(i)}{2}$, thus:

$$g_i(x) = w * 0.8^i * e^{-\left(x-\hat{F}_n(i)\right)^2 \Big/ \frac{\hat{L}_n(i)}{2}^2} \quad (1)$$

Note that the magnitude of the gaussian decreases with the order of the harmonic, i, and is proportional to w, the weight of the rest of the instruments in current audio file, or the coefficient extracted from a pre-existing mixing matrix. For example, if we align using solely a monaural signal in which all four instruments have the same weight, 0.25 for all four instruments, the coefficient will be $w = 0.75$.

The factor g_i penalizes frequencies which are in the allowed bands but are further away from the central frequencies. In this way, we eliminate transitions to other notes or energy which can add up noise later on in the blob detection stage.

Finally, for each note n and its associated $P_n(k)$ and $A_n(k)$ where $k \in [T_{on}(n), ...T_{off}(n)]$, we use the pitch salience function described in [10]. The algorithm calculates a salience measure for each pitch candidate, starting at $\hat{F}_n(0) - \hat{L}_n(0)$, based on the presence of its harmonics and sub-harmonics partials, and the corresponding magnitudes. Finally, the salience function for each time window is quantized into cent bins, thus the resulting matrix S_n has the dimensions $(T_{off}(n) - T_{on}(n), Q)$, where Q is the number of frequency bins for the six octaves. In our case, we experimentally choose $Q = 600$ bins.

2.2 Blob selection using image processing

The goals of the image processing stage are to obtain the location of the note onset and offset by binarizing the note-wise pitch salience, and to detect shapes and contours in the binarized image.

Accounting that the image binarization is not a robust process [12], different results are expected as a function of the amount of time overlap between notes, the salience of the pitch and its fundamental frequency. Therefore, as the shape and contour detection heavily relies on this step, we need a robust binarization, which would finally give us the best information for detecting the boundaries of the note.

Previous approaches to improve binarization rely on background subtraction or local binarization [12]. Therefore, we propose a binarization method similar to the local binarization, but adapted to our context: the pitch salience matrix. On the assumption that the bins closer to the fundamental frequency, $\hat{F}_n(0)$, are more salient than the ones at higher frequencies, we split the binarization areas in sub-areas related to the harmonic partials $\hat{F}_n(i)$. Thus, the salience matrix $\mathbf{S}_n$ is binarized gradually and locally, obtaining a binary matrix $\mathbf{B}_n$. Moreover, we consider l as the binarization step, moving gradually from 50 to 600 in steps of 50 bins.

Furthermore, we compute $\mathbf{B}_n$ in l steps, each time only for the columns in the interval $[l-50...l]$.

$$\mathbf{B}_n(i,j) = \begin{cases} 0, \mathbf{S}_n(i,j) < mean(\mathbf{S}_n^l) \\ 1, \mathbf{S}_n(i,j) \geq mean(\mathbf{S}_n^l) \end{cases} \quad (2)$$

where $i \in [T_{on}(n), ..., T_{off}(n)]$, $j \in [l-50...l]$, and $\mathbf{S}_n^l$ is a submatrix of $\mathbf{S}_n$, obtained by extracting the columns of $\mathbf{S}_n$ in the interval [0..l].

As an example, a pitch salience matrix $\mathbf{S}_n$ for a bassoon note is plotted in the Figure 2A. The green rectangles mark the submatrices $\mathbf{S}_n^l$ for various values of l. The resulting binarized image is depicted in Figure 2B.

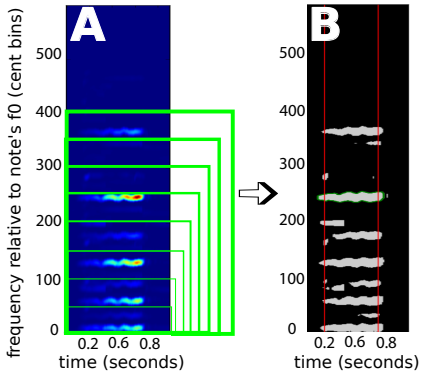

Figure 2. *Binarizing the spectral salience matrix (figure A) and detecting the blobs in the resulting image (figure B). Binarization is done gradually and locally, relative to the green squaresáreas in figure A. The ground truth onset and offset of the note are marked by vertical red lines.*

The next step is detecting boundaries and shapes on the binarized image. We use the connectivity rules described in [12, p. 248] in order to detect regions and the boundaries of these regions, namely the blobs. Thus, we want to label each pixel of the matrix $\mathbf{B}_n$ with a number from 0 to r, where r is the total number of detected blobs.

Having a pixel (i,j) with $i \in [T_{on}(n), ..., T_{off}(n)]$ and $j \in [0, ..., Q]$, where Q is the number of frequency bins, we need to consider all the neighboring pixels and we have to take into account their connectivity with the current pixel. The 4-way connectivity rules account for the immediate neighbors, as compared to 8-way connectivity which account for all the surrounding pixels. Because we are not interested in modeling transitions between notes, we discard diagonal shapes by using the 4-way connectivity rules. Hence, the connectivity matrix, which determines the neighborhood of the pixel (i,j), can be written as:

$$\mathbf{M} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

For the matrix $\mathbf{M}$, the central pixel with the coordinates (2,2) represents the origin pixel (i,j), and all the other non-zero pixels are the considered positions for the neighbors.

The algorithm, described in [12, p. 251], takes one pixel at a time and visits its non-zero neighbors. Then, we move sequentially from one pixel to its neighbors, setting boundaries for the pixels having neighbors equal to zero. Finally, the shape is enclosed when the algorithm reaches the pixel of origin.

Furthermore, once we have detected a set of blobs b_n for each note n, we need to compute the best combination of the blobs for all notes. Because search intervals for consecutive notes can overlap in time, choosing the best combination of blobs is not as trivial as picking the best blob in terms of area or salience, and the decisions that we take for a current note, should take into account the decisions we take for the previous and the next note. This kind of problem, which chains up a set of decisions can be solved with dynamic programming.

Consequently, we consider the blobs to be the vertices of an oriented graph, in which the edges are assigned a cost depending on the area of the two blobs and the overlapping between them, as seen in Figure 3. Basically, blobs with bigger area and little overlapping will have a lower cost, which makes them ideal candidates when we find the best path in the graph. Additionally, we can have an edge only between blobs of consecutive notes, and we can remove the edges between blobs which overlap more than 50% in time.

Therefore, we compute the area of each blob of the note n by summing up the values in the binarized matrix $\mathbf{B}_n$, enclosed by the corresponding blob contours. Additionally, we exclude the blobs which have the duration less than 100 ms, and the ones starting after the allowed interval for the attack time.

The normalized area of blob i for the note n is $H(b_n^i)$ and is a value inversely proportional with the actual area, because we want the larger blobs to have a lower cost,

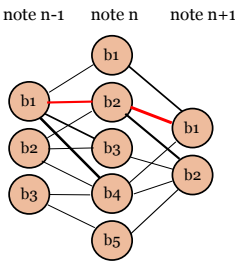

Figure 3. *A sample of the graph between three consecutive notes.* $b_{[1..5]}$ *are the blobs detected for each note. Thicker lines represent lower costs. The red line represents the best path in the graph.*

when picking the best path. In the same manner, we must increase the cost as the overlapping between the blobs increases. Thus, for two adjacent notes n and $n+1$, $O(b_n^i, b_{n+1}^i)$ has cost 1 if there is no overlapping, and an increased value summing up the ratio of the the area of the two overlapping blobs. For instance, if 20% of the area of the first blobs overlaps with 70% of the area of the second blob, $O = 1 + 0.2 + 0.7 = 1.9$.

Thus, the cost for the edges has the expression

$$cost(b_i^n, b_{i+1}^n) = O(b_n^i, b_{n+1}^i) * (H(b_n^i) + H(b_{n+1}^i))$$

In order to find the shortest path between the vertices of the first note in the score and the last one, we use Dijkstra's algorithm described in [5]. The algorithm finds the shortest path for a graph with non-negative edges by assigning a tentative distance to each of the vertices and progressively advancing by visiting the neighboring nodes.

Additionally, after the best path is computed, we can face the situation where two consecutive blobs overlap in time due to the inaccuracy in binarization and the fact that the minimum cost path does not guarantee no overlapping. Because the melody for a particular instrument is considered to be monophonic, we do not allow overlapping between two consecutive notes. Thus, we ought to find a splitting point between the starting point of the blob associated with the next note and the ending point of the blob associated with the current note.

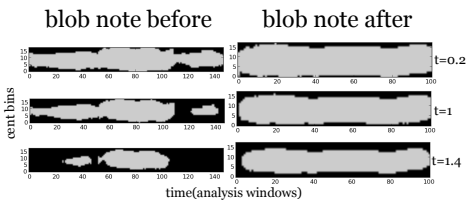

Figure 4. *Blob refinement using adaptive threshold binarization of two consecutive overlapping blobs in the best path. The minimum overlapping is achieved for threshold* $t = 1.4$

Having two consecutive blobs from the best path, b_n and b_{n+1}, we take the image patches surrounding their boundaries and we adaptively increase the threshold of binarization until the minimum overlapping is achieved. Consequently, we consider the submatrices $\hat{\mathbf{S}}_n$ and $\hat{\mathbf{S}}_{n+1}$ of the corresponding pitch salience matrices $\mathbf{S}_n$ and $\mathbf{S}_{n+1}$, and for a variable threshold $t = [0.2..2]$, we compute the binary matrices $\hat{\mathbf{B}}_n^t$ and $\hat{\mathbf{B}}_{n+1}^t$.

$$\hat{\mathbf{B}}_n^t(i,j) = \begin{cases} 0, \hat{\mathbf{S}}_n(i,j) < t * mean(\hat{\mathbf{S}}_n) \\ 1, \hat{\mathbf{S}}_n(i,j) \geq t * mean(\hat{\mathbf{S}}_n) \end{cases} \quad (3)$$

As seen in Figure 4, the higher the threshold t, the less pixels are be assigned to value 1 in the binary matrices, thus we increase the threshold gradually until no overlapping is achieved.

Finally, the note onset and offset are extracted from the leftmost and the rightmost pixels of the refined blobs in the best path.

3. EVALUATION

3.1 Experimental setup

The dataset used to evaluate our proposal consists of 10 human played J.S. Bach four-part chorales, and is commonly known as Bach10 . The audio files are sampled from real music performances recorded at 44.1 kHz that are 30 seconds in length per file. Each piece is performed by a quartet of instruments: violin, clarinet, tenor saxophone and bassoon. Each musician's part was recorded in isolation. Individual lines were then mixed to create 10 performances with four-part polyphony. More information about this dataset can be found in [6].

We observe that the dataset has a few particularities. First, every recording presents fermatas, where the final duration of the note is left at the discretion of the performer or the conductor, making it more difficult to detect the onset and offsets of the notes. Second, the chorales have a peculiar homophonic texture. Third, the annotated note onsets and offsets in the ground truth can have more or less notes than the actual score. We discovered that this mismatch comes from repeating notes, which in the original score are represented by a single larger note. This step also makes the detection of the note offsets more difficult.

In order to perform alignment at the note level, we generate a misaligned score by introducing onset and offset time deviations for all the notes and all the instruments in the ground-truth score. The deviations are randomly and uniformly distributed in the intervals $[-200, ..., -100]$ and $[100, ..., 200]$ ms. Moreover, we aim at refining the alignment of the algorithm proposed by [3]. Thus, we correct the onset times and we detect the offsets around the beginning of the next note. For both of these tasks we consider the interval $[-200, ..., 200]$ ms.

Furthermore, the STFT is computed using a Blackman-Harris 92dB window with a size of 128 ms and, a hop size of 6 ms. Additionally, we zero-pad the window by three times its length. Moreover, frequencies and magnitudes of the spectral peaks are extracted with the algorithm described in [8], which uses parabolic interpolation to accurately detect positive slopes in the spectrum computed at the previous step.

3.2 Results

We aim at correctly aligning the onsets and offsets of the misaligned score described in Section 3.1 and we add up 200 ms before and after the note boundaries in order to search for the exact starting and ending point of the note. Thus, our algorithm can have up to 400 ms in error for the onsets, and a larger error for the offset, because we are not constraining the duration of the note to any interval.

For each piece, aligned rate (AR) or precision is defined as the proportion of correctly aligned notes in the score and ranges from 0 to 1. A note is said to be correctly aligned if its onset does not deviate more than a threshold from the reference alignment. To test the reliability of our method, we tried different threshold values ranging form 15 to 140 ms. Other measures as the average offset (i.e. average absolute-valued time offset between a reported note onset by the score follower and its real onset in the reference file) and the std offset (i.e. standard deviation of sign-valued time offset) are also considered.

As illustrated in figure 5, the proposed system is able to accurately align more than the 30% of the onsets with a detection threshold lower than 15 ms. Furthermore, more than 80% of the onsets are accurately detected with a threshold of 60 ms. Because the search time interval for the note allows for error larger than 200 ms, the AR for the onset does not reach 100% in $t = 200ms$, as less than 2% of the onsets have larger errors.

Furthermore observe that we less accurate in detecting the offsets, particularly when we do not know the approximate note offset and we estimate it around the onset of the next note, as when we take as input the alignment of the algorithm proposed by [3]. The drop in performance of the offset detection can also be explained by the fact that the energy of a note can decay below a threshold, thus excluding it when binarization is performed.

Figure 6 shows boxplots of the average offset and the std error for each instrument, and for the note onset and offset, for the misaligned dataset. The lower and upper lines of each box show 25th and 75th percentiles of the sample. The line in the middle of each box is the average offset. The lines extending above and below each box show the extent of the rest of the samples, excluding outliers. Outliers are defined as points over 1.5 times the interquartile range from the sample median and are shown as crosses.

We observe that performance is lower for violin compared to the other instrument. This can be explained by the fact that for this dataset the violin has noisier or soft attacks, which do not yield a high enough value in terms of pitch salience, and is lost when binarizing the image.

Moreover, the fact that we are able to detect most of the onsets in the interval 0.06 seconds, which is an acceptable interval for the attack of the instruments aligned, point us on some limitation on using the pitch salience function, which is not able to be accurate enough with noisier attacks, as it happened for the violin.

Furthermore, we want more insight on how the errors are distributed across the time range. Thus, we plot the 2-d histogram of the onset errors, as seen in Figure 7. We

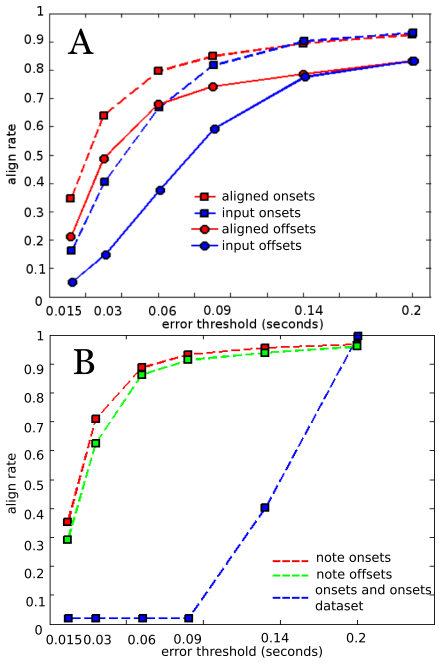

Figure 5. *The proposed system improves the align rate of (A) the system proposed by [3] and of (B) the misaligned dataset, for onset errors, as well as offset errors*

observe that even though the original dataset had large errors, our method was able to detect the note onsets within a small time frame, as most of the errors are in the bin centered at zero.

Moreover, our method is better at fixing the delays in the note onsets. In comparison, we can commit more errors if the onset of the note is thought to be before the actual onset, because the window in which we have to look for it overlaps more with the previous note, hence we have more interference.

Additionally, for every note and every instrument, we compute the percentage of correctly detected frames with respect to ground truth. Our algorithm is able to correctly detect 89% of the frames of the ground truth notes. In comparison, the notes in the misaligned dataset have a degree of 66% correctly detected frames.

Finally, we compute the percentage of frames which are erroneously detected as part of the notes. We observe that solely 0.07% of frames from the notes we refine are outside the boundaries of the ground truth notes, compared to the misaligned dataset, for which 34% of the frames are displaced outside the time boundaries of the notes.

Therefore, our algorithm is more likely to shorten the notes, rather than making erroneous decisions regarding their time frame. This is due to the way we are picking the best sequence of blobs, which penalizes the overlapping, thus picking blobs which have a smaller area but less overlapping with the blobs from neighboring notes.

4. CONCLUSIONS

We proposed a method to refine the alignment of onsets and offsets in orchestral recordings, using audio and im-

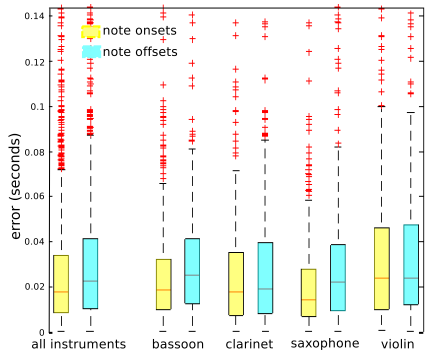

Figure 6. *The average offset and the std offset in terms of 25th and 75th percentile of the proposed system for bassoon, clarinet saxophone, and violin, for note onsets, as well as note offsets*

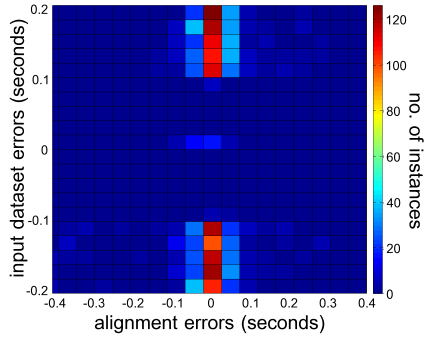

Figure 7. *The histogram of error distribution in the onset alignment*

age processing techniques. We compute a note-wise pitch salience function and we binarize it. Moreover, we detect blobs in the binarized image, and we pick the best blob candidate for each note by finding the best path in the associated graph. Furthermore, as offset detection is regarded as a more difficult problem, the proposed method addresses this issue by detecting image blobs to simultaneously label note onsets and offsets.

The evaluation shows that our method is able to refine the alignment in a misaligned dataset, having detected more than 80% of the onsets with an error of 60 ms. Moreover, we analyzed the performance across all four instruments, and we discovered that the accuracy drops for a violin, as being higher for the other instruments. Thus, as a future step, we need to analyze what limitation has the algorithm regarding certain instrument classes. Additionally, the proposed method should be tested with another dataset, with more complex polyphonies and tempo variations.

Furthermore, our method can be improved by using timbre models when filtering the spectral peaks and decreasing their magnitude. Additionally, choosing the best sequence of blobs can be improved by using a better cost function for the Dijkstra's algorithm. In addition, one could use image processing with other data obtained by audio processing means, as the spectrogram or come with a more robust approach than the pitch salience which does not capture noisy note attacks or noisy spectrum.

Finally, the note refinement can be used to improve the performance of score informed source separation, in the situation where the score is not well aligned with the audio.

5. ACKNOWLEDGEMENTS

This work was supported by the European Commission, FP7 (Seventh Framework Programme), STREP project, ICT-2011.8.2 ICT for access to cultural resources, grant agreement No 601166. Phenicx Project

6. REFERENCES

[1] J.-J. Aucouturier and M. Sandler. Finding repeating patterns in acoustic musical signals. *VIRTUAL, SYNTHETIC, AND ENTERTAINMENT AUDIO*, pages 412–421, 2002.

[2] J.J. Bosch, K. Kondo, R. Marxer, and J. Janer. Score-informed and timbre independent lead instrument separation in real-world scenarios. In *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, pages 2417–2421, Aug 2012.

[3] J.J. Carabias-Orti, P. Vera-Candeas, F.J. Rodriguez-Serrano, and F.J. Canadas-Quesada. A RealTime Audio to Score Alignment System using Spectral Factorization and Online Time Warping. *IEEE Transactions on Multimedia(submitted)*, 2014.

[4] A. Cont. A coupled duration-focused architecture for real-time music-to-score alignment. *Pattern Anal. Mach. Intell. IEEE ...*, 32:974–987, 2010.

[5] E. W. Dijkstra. A note on two problems in connexion with graphs. *NUMERISCHE MATHEMATIK*, 1(1):269–271, 1959.

[6] Z. Duan and B. Pardo. Soundprism: An online system for score-informed source separation of music audio. *Selected Topics in Signal Processing, IEEE ...*, pages 1–12, 2011.

[7] S. Ewert and M. Muller. Using score-informed constraints for NMF-based source separation. *Acoustics, Speech and Signal Processing (...*, 2012.

[8] J. O. Smith Iii and X. Serra. Parshl: An analysis/synthesis program for non-harmonic sounds based on a sinusoidal representation . 1987.

[9] C. Joder and B. Schuller. Off-line refinement of audio-to-score alignment by observation template adaptation. *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 206–210, 2013.

[10] A. Klapuri. Multiple fundamental frequency estimation by summing harmonic amplitudes. In *in ISMIR*, pages 216–221, 2006.

[11] B. Niedermayer. *Accurate Audio-to-Score Alignment Data Acquisition in the Context of Computational Musicology*. PhD thesis, Johannes Kepler Universität, 2012.

[12] M. Nixon. *Feature Extraction and Image Processing*. Elsevier Science, 2002.

[13] S. Senturk, A. Holzapfel, and X. Serra. Linking Scores and Audio Recordings in Makam Music of Turkey. *Journal of New Music Research*, pages 35–53, 2014.

[14] T.M. Wang, P.Y. Tsai, and A.W.Y. Su. Score-informed pitch-wise alignment using score-driven non-negative matrix factorization. In *Audio, Language and Image Processing (ICALIP), 2012 International Conference on*, pages 206–211, July 2012.