

PAPER

Automatic alignment of a musical score to performed music

Yoram Meron¹ and Keikichi Hirose

*Department of Information and Communication Engineering, the University of Tokyo,
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656 Japan*

¹*Present address: Panasonic Speech Technology Laboratory, Santa Barbara, CA, USA
e-mail: (meron, hirose)@gavo.t.u-tokyo.ac.jp*

(Received 8 May 2000, Accepted for publication 14 December 2000)

Abstract: This paper presents several methods for the alignment of a music score (given in MIDI form) to a human performance of the same musical piece. Two distinct cases are considered: first, MIDI-to-MIDI alignment, where rhythm changes, note time shifts, and player errors are handled by a Dynamic-Programming (DP) algorithm. Next, a method for MIDI-to-audio alignment is presented, incorporating spectral data into the DP method. Experiments on a music database showed the effectiveness of the alignment algorithms. For cases where the alignment process gets trapped in a wrong local minimum, an efficient manual bootstrapping method was developed and shown to lead to the selection of the correct alignment.

Keywords: Automatic alignment, Dynamic time warp, MIDI, Performed music, Singing synthesis

PACS number: 43.58.Ta, 43.75.Yy

1. INTRODUCTION

The work described here was done as part of the development of a singing synthesis system, which improved on previous singing synthesis methods by being able to create more natural and expressive singing, while requiring much less manual work in order to make it learn the voice of a new singer. To achieve these goals, we have introduced the use of an automatically trained concatenative singing-voice synthesis system [1].

Our system uses a relatively large database of singing by one singer (30–60 minutes) to “learn” the singer’s voice. By using speech recognition techniques, the system automatically segments the input into short waveform segments, corresponding to phonetic units, and constructs a database of these units, annotated with phonetic, pitch, duration, timbre, and energy information. The system can then synthesize a new song by concatenating the units from the database which best match the features of the desired new song. By using a large unit database, we are able to capture and recreate more of the singer’s variability and expressiveness. At this stage we assume that the pitch and energy contours, as well as the phonemes and their durations, are given for the new song (e.g. copied from another singer). We do *not* deal with the problem of directly converting the score into singing, or with the conversion of singing to score.

Since the phonetic segmentation and the pitch extraction processes are both prone to error, we try to improve their accuracy by using the additional information which is included in the score: the identity and timing of each note (We use the term *note timing* to refer to the onset and release times of a note.).

However, when a musician performs a musical piece, the performed music is different from an exact reproduction of the transcribed notes — differences can be in note timing, duration, amplitude, and for some instruments pitch and timbre. A musician might also make mistakes — e.g., a pianist can miss some keys, strike wrong keys, or the right keys at wrong times. Although for skilled players these errors are rare, and usually hardly perceived by listeners [2], errors are practically unavoidable.

Therefore, we need to obtain a time alignment between the score and the performed music. Using this alignment, we can get an estimate for the timing of each of the pianist’s and singer’s notes, and use it to restrict the pitch extraction and the phonetic segmentation processes. Previous methods for obtaining the time alignment were found to be unsatisfactory for our needs, therefore we introduce a new alignment method, based on a dynamic programming approach, which was shown to produce better results.

In case the audio signal is a mixed recording of singing and piano, a sound separation process is also

needed, in order to remove the piano part before the system can learn the voice of the singer. The estimate of the timing of the piano notes (obtained from the alignment information) is used to initialize the sound separation process [3].

In this paper two kinds of score alignment are considered: alignment of a MIDI score to a MIDI recording of a pianist, and alignment of a MIDI score to an audio recording of a pianist (MIDI — *Musical Instrument Data Interface* — a communication protocol allowing musical instruments to communicate with a computer and with each other, and also allowing a standard way to save musical information in a compact, easily modifiable form [4]). Because of the expressive information in the pianist's playing, the alignment between the score and the performed music is not trivial, for both of these cases.

For training our singing synthesis system, we consider 3 types of training data, which require different types of score alignment:

1. The pianist is recorded using a MIDI sequencer; the singer sings along with the piano (played over headphones) and is recorded separately. In this case we use the time alignment between the score and the sequenced piano (MIDI-to-MIDI) to obtain an estimate for the timing of the singer's notes.
2. Similar to the above, but no MIDI information is available for the piano (only the audio recording). In this case, the score needs to be aligned to the audio signal (MIDI-to-audio alignment).
3. A commercial CD containing a mixed recording of singer and piano, with no available MIDI information. In this case, the MIDI-score-to-audio alignment is needed for initializing the sound separation process, as well as for estimating the timing of the singer's notes.

2. TEST MATERIAL

For evaluating the results of the alignment procedures (both MIDI-to-MIDI and MIDI-to-audio), a music database was constructed.

2.1. Music Database

For this database, all 24 songs from the song cycle "*Winterreise*" and 7 songs from the song cycle "*Schwanesang*" were selected. These songs were written by F. Schubert for piano and voice. A MIDI format transcription of the songs was obtained through internet [5]. In this work we assume this transcription is correct, although this was not verified. The MIDI files contain the piano and the voice parts, on separate channels. A pianist was asked to play the accompaniments to the songs on a MIDI keyboard (Technics P30 keyboard), and was

both audio recorded on DAT and MIDI recorded using a MIDI sequencer program. The DAT recording was then digitized with a sampling rate of 44.1 kHz and 16 bits per sample. Two singers, one male and one female, were asked to sing along with the MIDI-replayed piano, which was played over headphones. Both the pianist and the singers were university music students, pursuing professional musical careers.

2.2. Texture

For MIDI-to-MIDI alignment, texture (We use the term *texture* to refer to the number of simultaneous *notes*, instead of the term polyphony, which refers to the number of simultaneous *voices*.) is not critical. However, for MIDI-to-audio alignment, dense textures can adversely affect the alignment accuracy.

The alignment of MIDI to audio is especially difficult when simultaneous notes are harmonically related (have overlapping harmonics). The average density of texture of the piano part in "*Winterreise*", calculated from the MIDI recording of the pianist, is shown in Fig. 1. The light bars in the figure show the average texture density, considering the use of the sustain pedal. The dark bars show the texture density, calculated for the same recordings by disregarding the sustain pedal.

2.3. Jittered MIDI

For the evaluation of the accuracy of a given alignment (either MIDI-to-MIDI or MIDI-to-audio), we look at the difference between the actual time a note occurred and the time at which the alignment estimates it occurred. When note insertions and deletions are present, the evaluation of the resulting alignment becomes more difficult. For example, if the pianist hit a key 4 successive times instead of 3, it is unclear between which notes the time

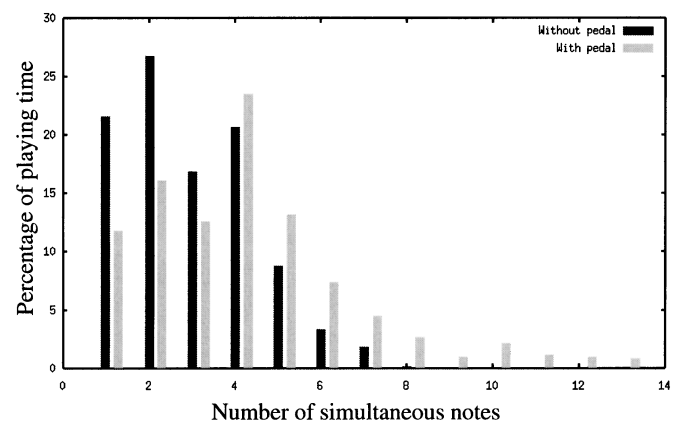


Fig. 1 Average texture density of the piano part in "*Winterreise*" as played by a pianist, given as percent of play time, calculated *without* (dark bars) and *with* (light bars) consideration of the sustain pedal.

difference should be measured.

For this reason another set of test files was constructed, which guarantees a known one-to-one match between the notes of two files. These files are referred to here as *jittered MIDI* files, and are created by adding “noise” to the original MIDI files. A computer program was written which takes as input a MIDI file, modulates the local tempo of the MIDI file, and adds a small time shift to each individual note. The program applies a multiplicative factor to the local tempo. This factor randomly changes in a smooth way between the values of 0.66 and 1.5. Note timings are changed by small random shifts of up to ± 0.03 s, which is more than expected for a pianist playing simultaneous notes [6].

3. MIDI-TO-MIDI ALIGNMENT

The alignment is achieved using a dynamic programming method, a variation of the dynamic time warping (DTW) algorithm [7]. This algorithm was originally developed for the alignment of speech sequences, which are subject to nonlinear duration changes. In the original DTW algorithm, the spectrum of two speech sequences is compared. For the MIDI-to-MIDI alignment, we adapt the DTW algorithm to perform a comparison between two MIDI files. A brief explanation of the DTW algorithm is given below.

The two MIDI files are divided into an equal number (N) of time frames (fixed frame duration for each file). Each frame is represented by the set of piano notes which are held down at the time point corresponding to the frame. In this explanation it is assumed that the pianist-performed MIDI file lies along the x axis ($P_{1..N}$) and the MIDI file corresponding to the score lies along the y axis ($S_{1..N}$) of a two-dimensional grid. In the following explanation, each grid point (i, j) corresponds to a pair of MIDI frames: P_i and S_j . The DTW finds an alignment function $F_{\text{align}} : S \rightarrow P$ which aligns the score-MIDI and pianist-MIDI frames.

The algorithm starts by calculating a local weight function $L(i, j)$, a measure showing the similarity between P_i and S_j . The local weight is defined as the ratio between the number of identical notes in the two frames and the number of different notes in those two frames. For example: if one frame has MIDI notes $\{60, 64, 72\}$ and the other has the notes $\{52, 64, 72\}$, then there are a total of 4 different notes $\{52, 60, 64, 72\}$ and 2 matching notes $\{64, 72\}$, giving a match weight of $2/4 = 0.5$.

The algorithm then goes over the grid points, from left to right, and from bottom up, and calculates the accumulated weight function $Ac(i, j)$. For each point (i, j) , it looks at all the possible transitions to (i, j) . The transitions are restricted to the ones shown in Fig. 2. This con-

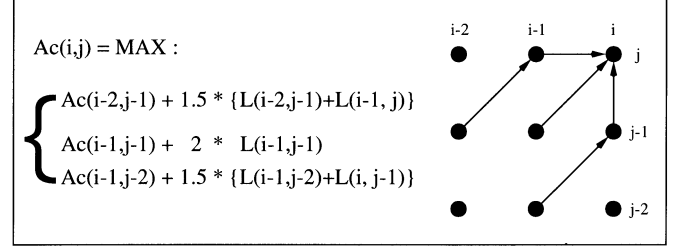


Fig. 2 Allowed transition types in the DTW algorithm, and their associated accumulated weight $Ac(i, j)$

straint on the transitions effectively limits the allowed local tempo change to be between the values of 0.5 and 2.0. The transition producing the maximum value of $Ac(i, j)$ (as shown in Fig. 2) is selected, and a backtracking table $B(i, j)$ stores the selected transition type (one of the transitions in Fig. 2). After the calculation of $Ac(i, j)$ is completed for the whole grid, the alignment path with the highest weight is obtained by backtracking from (N, N) to $(1, 1)$ according to $B(i, j)$. For full details of the DTW see [7].

3.1. Note-to-note Matching

The found alignment function provides a match between the the score and the pianist MIDI frames, from which we find a match between the score and the pianist notes. For a given score-note \bar{n} , we look at the score-MIDI frame S_j where its onset occurred. We expect to find a matching pianist-note onset around the frame P_i , where $i = P_{F_{\text{align}}(j)}$. A 3 frame window around the frame P_j is searched. If the onset of a note of the same key as \bar{n} is found in the window, a match between the two notes is marked (see Fig. 3). If several matching notes are

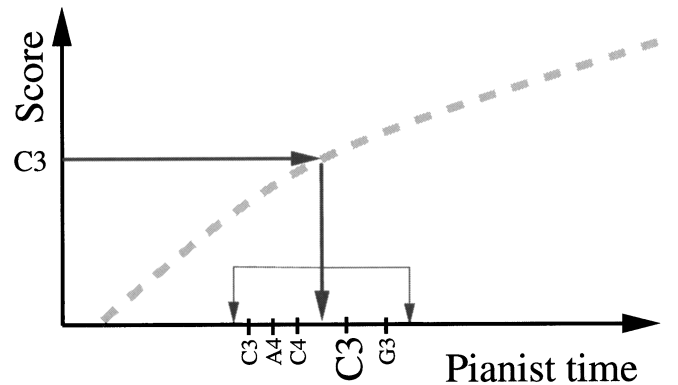


Fig. 3 Example of note matching- the alignment function (dashed line) in a short interval around a score note to be matched (C3) is shown. The score note is projected through the alignment function onto the pianist MIDI time scale. A short window around the projected time is searched for occurrences of the same note. The closest occurrence is selected (large font C3).

found inside the window, then the pianist-note closest to the aligned onset time of \bar{n} is selected. If several score-notes are matched to the same pianist-note, then only the closest score-note is marked as matched, and the rest as unmatched.

The found note-to-note match is used to create a refined time alignment function \hat{F}_{align} : for a given time point in the score-MIDI, we assume that in a short interval around it, the alignment can be approximated by a linear function. To estimate this linear function, we take 5 matched score-notes before the given time point, and 5 matched score-notes after it. In case these 10 notes span more than 3 MIDI frames, we set \hat{F}_{align} to be equal to F_{align} . Otherwise, linear regression is used to estimate a linear alignment function from the 10 pairs of selected score-notes and their matching pianist-notes' onset times.

The MIDI-to-MIDI alignment is needed in order to obtain an estimate of the timing of the voice, which was singing along with the pianist-MIDI. In order to do that, the notes of the voice part in the score-MIDI have to be projected onto the pianist-MIDI time scale, using the time alignment function. This can be seen in Fig. 4: for each voice note in the score-MIDI, we construct \hat{F}_{align} and use it to find the time point in the pianist-MIDI where the singer is expected to sing the note (not taking into account deviations in timing introduced by the singer for

expressive purposes).

The same method can be used to "correct" the pianist-MIDI file: to create a new MIDI file with one note for each score-note. If the score-note is matched, we use its matched note's information. If not, we predict the note's timing using \hat{F}_{align} . In this work we are not concerned with predicting the amplitude for these notes; therefore, simple averaging of the amplitudes of close notes is used.

3.2. MIDI-to-MIDI Alignment Test

The MIDI-MIDI alignment algorithm, described in the previous section, was run on 10 songs from the music database, to obtain an alignment between 10 pairs of score-MIDI and pianist-MIDI files.

On average, more than 95.5% of the score-notes were matched with pianist-notes. Running the alignment between the score-MIDI and jittered-MIDI files gave a correct match for more than 99.5% of the score-notes.

Two qualitative, informal listening experiments were conducted by 3 listeners without piano training. The listeners were familiar with the musical pieces. In the first experiment, the voice part from the score-MIDI was aligned with the pianist-MIDI and then MIDI-synthesized as a musical instrument (organ) along with the pianist-played piano part. No obvious errors in the timing of the voice part were perceived.

In the second experiment, the corrected pianist-MIDI file (as explained in the previous section) was created for a short piece, MIDI-synthesized, and played back to the same listeners. Less than 1% of the notes (3 out of 352) were judged to be clearly played at the wrong time. Amplitude errors, which are not our present concern, were more frequent.

4. MIDI-TO-AUDIO ALIGNMENT

Next, the alignment of MIDI to audio recordings is considered. We start by reviewing two previously published systems, describe the performance of one of them on our music database, and suggest ways to improve the alignment results

4.1. Existing Systems

In [8] a system was developed for determining the tempo of music performed by percussion instruments. This system assumes that only one instrument plays at a time. It relies on the well-defined time localization of percussion attack sound, and suffers accuracy degradation when bleed from other instruments is present.

In [9], an experimental system was built which extracts expressive information from a performance (recording) of a piano score (without voice). This system first analyzes the score to determine which notes are expected to

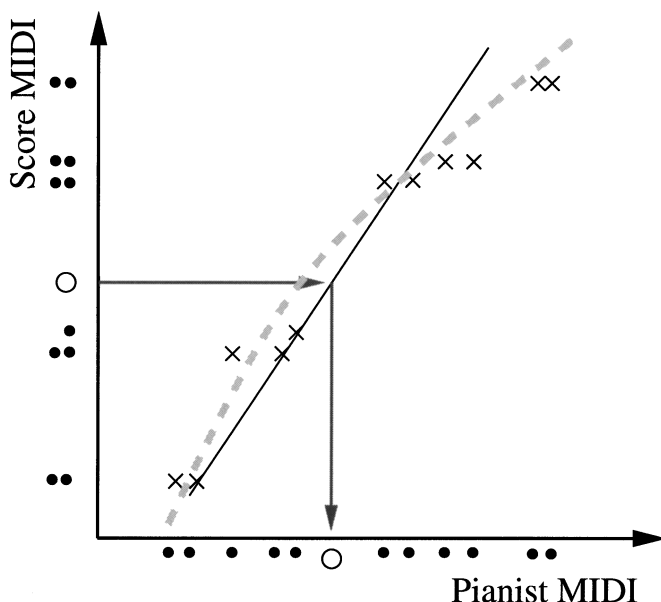


Fig. 4 Estimating the time of a singer's note: The DTW alignment (dashed curve) is used to find a match between score and pianist notes (dark circles). The matched notes (crosses) are used to estimate a local alignment around the time of the singer's note (unfilled circle on the score axis), by using linear regression (diagonal thin line). The local alignment is used to project the singer's note onto the pianist-MIDI time scale.

be struck together and which notes are expected to overlap. The system then enters a loop in which it detects note onset and release times, from which it can update the local estimate of the tempo, which is used to improve the time prediction for the next iteration. The detection of note onset and release is performed by looking at the energy envelopes (and their derivatives) at the output of filters on the audio signal, around the predicted time. The selection of the filters depends on the context information derived from the score, using sharper filters when necessary to avoid close harmonics of simultaneous notes.

A test of the system [9] reports successful alignment for short musical pieces (100–150 notes per piece), with an error standard deviation of about 20 ms for the detection of note onsets.

4.2. Application to Our Music Database

The applicability of the method described in [9] was tested on our music database. In our database, the pieces are between 450 to 3,000 notes long, often with a dense texture, and with considerable tempo changes. Another problem is the existence of a vocal part in addition to the piano part, which makes the tracking of the piano more difficult. It is not possible to predict in advance the exact pitch of the voice at each time point, especially when vibrato is present (see Fig. 5, for an example of the expected and actual singer pitch values).

The system described in [9] was re-implemented and run on our music database. Running the system using score-MIDI files and audio files created from their corresponding pianist-MIDI files showed that this alignment method was not able to keep track of the music for most of the pieces. This is due to the fact that the system calculates an expectation time for the next note, based on the previous note and the local tempo. If several successive notes are not found close to their expected times, the system loses track of the music and has no mechanism to recover from that point on.

5. MIDI-TO-AUDIO ALIGNMENT USING DTW

To solve this problem, a two-step approach is taken; first, a rough alignment for the whole piece (using a DTW algorithm) is obtained, and then a second pass of alignment refinement is run.

The DTW implementation is similar to that used for the MIDI-to-MIDI alignment, except for the calculation of the local weight function. The pianist audio file is divided into equal duration audio frames $A_{1..N}$, each corresponding to a short segment of raw waveform. The score-MIDI file is divided into equal duration MIDI frames $M_{1..N}$, each of which is represented as the set of piano notes held down at the time corresponding to the frame.

The only difference in the algorithm is the local weight function $L(i, j)$, which has to compare an audio frame A_i to a MIDI frame M_j . First, we construct $harm(j)$: a set of all the harmonics of all the notes in M_j . Duplicate harmonics are removed from $harm(j)$ (i.e., if two harmonics have the same frequency f_0 , then $harm(j)$ will contain f_0 only once).

For the harmonic h in $harm(j)$, we define the energy associated with h for the grid point (i, j) to be:

$$E_j^i(h) = \left(\sum_{k=1}^{len} a_i(k) \cos(k * Fr_h) \right)^2 + \left(\sum_{k=1}^{len} a_i(k) \sin(k * Fr_h) \right)^2,$$

where a_i is the audio segment, consisting of len samples, corresponding to audio frame A_i , and Fr_h is the frequency of the harmonic h . Finally, we define

$$L(i, j) = \sum_{h=1}^{N_{harm}} E_j^i(h),$$

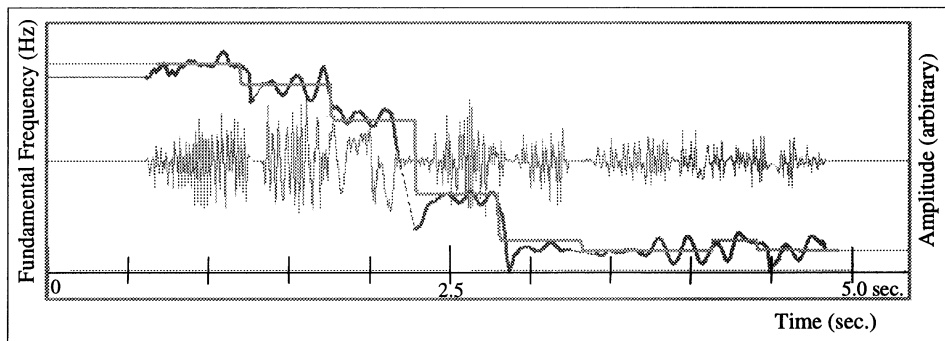


Fig. 5 Score notes of the voice part (light stepwise line) imposed over the actually extracted singer's fundamental frequency (dark wavy line). The audio signal is also shown.

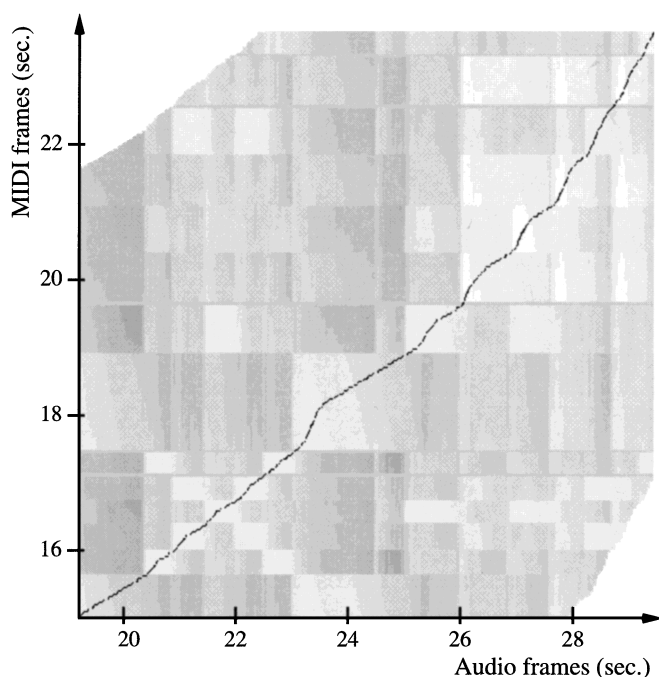


Fig. 6 Example DTW result — a part of the DTW grid. The dark line shows the found alignment. The local weight of the grid points is indicated by different shades of gray — lighter shades show higher local weight. The DTW algorithm tries to find an alignment function which goes through grid points with a high local weight.

where N_{harm} is the number of harmonics in $harm(j)$. Several energy normalization schemes are considered in section 5.2. below.

Figure 6 shows the alignment function and the values of the local weight function for a short part of an example piece.

5.1. Overall Tuning

Before the DTW can be run, the tuning of the piano has to be found. Using an incorrect tuning would reduce the accuracy of the local weight function $L(i, j)$, as it would sum the energy over wrong frequencies.

In [9], a manual tuning method is described in which the tuning is obtained by locating an occurrence of an isolated piano note and applying to it a set of filters with slightly different center frequencies. The frequency maximizing the energy at the output of the filter is chosen as the correct tuning. The rest of the notes are tuned relative to this isolated note, as it is assumed that the piano's notes are in tune with each other.

In this work, we have developed an automatic tuning method, which does not require locating an isolated note (which may not exist). For the first non-silent MIDI frame, we construct a set of harmonics $tune_harm$ in a similar way to the construction of $harm(j)$, but the frequencies of the harmonics in $tune_harm$ are calculated by

multiplying a reference frequency $Freq_{ref}$ by a tuning factor. In other words, all the harmonics of all the notes are assumed to be well tuned relative to $Freq_{ref}$.

Formally, we assume $\hat{F}r_h = Ratio_h * Freq_{ref}$, where F_rh is the frequency of the harmonic component h , and $Ratio_h$ is the theoretical ratio between the reference frequency and h . For example, if the reference frequency corresponds to a middle C, and h corresponds to the second harmonic of a C one octave above middle C, $Ratio_h$ is exactly 4. (A more exact value of $Ratio_h$ should include the effect of piano inharmonicity [10].)

For a given reference frequency, we then define for each harmonic h in $tune_harm$:

$$\hat{E}_h(Freq_{ref}) = \left(\sum_{k=1}^{len} a(k) \cos(k * \hat{F}r_h) \right)^2 + \left(\sum_{k=1}^{len} a(k) \sin(k * \hat{F}r_h) \right)^2,$$

where $a(k)$ is the audio segment corresponding to the first non silent MIDI frame, and $\hat{F}r_h$ is obtained from $Freq_{ref}$ as explained above.

A loop is run over finely spaced values of $Freq_{ref}$, and the value which maximizes the energy sum $\sum_{h=1}^{N_{harm}} \hat{E}_h(Freq_{ref})$ is selected as the tuning frequency.

5.2. Algorithm Parameter Tuning

A heuristic fine tuning of the algorithm parameters was done in order to improve the performance of the MIDI-to-audio alignment process.

Frame size:

The frame shift and frame length were set to 0.015 s and 0.045 s respectively, achieving a favorable tradeoff between efficiency and accuracy.

Release lag:

A parameter controls the duration for which a harmonic is still considered after a note is released. In other words, in how many frames after the MIDI frame containing the note's release, the energies of the note's harmonics are still included in the calculation of the local weight.

Best results were obtained for no release lag. The energies of a note's harmonics are not considered beyond the MIDI frame where it was released. Increasing the lag induced premature detection of some notes' onset.

Note amplitude decay:

To improve the alignment accuracy, an amplitude parameter was used — high in the beginning of the note, and decreasing as time progresses. This amplitude parameter is used to multiply the harmonics' energy values, so that a better match (higher weight) is obtained between the first MIDI frame of a note and the first audio frame of the note.

The decay function finally selected was an exponential decay of 85% per second. This corresponds to fast note decay, placing more weight on the correct alignment of note onsets (In this experiment, the amplitude decay was considered independent of the note's pitch. Taking into account such a dependency may improve alignment results.).

Frequency penalties:

An option was tested where $L(i, j)$ is calculated as the sum of energies of the harmonics in $harm(j)$, minus the sum of the energies of the harmonics which appear in $harm(j-1)$ and $harm(j+1)$, but do *not* appear in $harm(j)$.

As an explanation, consider the example where the onset of a note appearing in a MIDI frame M_j actually occurred at audio frame A_i . Using the frequency penalty method, not only does $L(i, j)$ get a high value, but the value of $L(i, j-1)$ gets lowered, as energies in unwanted frequencies are detected. This can help find the correct alignment of the note.

In practice, due to the existence of singing in the audio signal, the voice can accidentally increase this negative weight by crossing one of the unwanted frequencies, increasing the risk of wrong alignment. Results were not conclusive regarding the frequency penalty method, thus it was rejected.

Energy normalization:

As explained above, $L(i, j)$ is calculated by summing up the energies of the harmonics in $harm(j)$ for the audio frame A_i . Using this kind of calculation causes the local weight to be higher in places where $harm(j)$ contains a larger number of harmonics. Thus, the selected alignment tends to stay longer in grid points corresponding to MIDI frames with denser texture (although the frequency penalty method can reduce this tendency). A partial solution is to normalize the energy sum by dividing it by a factor proportional to the number of summed harmonics' energies.

In a similar way, the alignment path is biased to stay longer in grid points corresponding to audio frames where notes were played with higher amplitudes. A partial solution is to calculate the total energy of the frame, and to normalize the sum of energies of the harmonics in $harm(j)$ by the total energy. However, in the presence of singing this normalization scheme can wrongly lower the value of the local weight.

Best results were obtained when dividing the energy sum by the number of summed harmonics, and when not using total frame energy normalization.

Silence treatment:

For silent MIDI frames (assuming we do not use the frequency penalty method) the value of the local weight

is necessarily 0, as $harm(j)$ does not contain any harmonics. There is no increase in the value of $L(i, j)$ for grid points corresponding to silent MIDI-frames and low energy audio-frames. Since a local weight of 0 generally indicates a total mismatch between A_i and M_j , the alignment will tend to avoid grid points corresponding to silent MIDI-frames.

To try and improve this procedure, $L(i, j)$ is given an artificially high value at grid points where the total energy in A_i is below a threshold, and where M_j is a silent MIDI frame.

Search beam:

Because of memory and time considerations, it is not desirable to calculate $L(i, j)$ and $Ac(i, j)$ for the whole grid. Instead, a limit on the maximum beam width of the DTW is enforced; after the DTW fills the i -th column of $Ac(i, \cdot)$, the grid point (i, \hat{j}) with the highest value in this column is found. Progressing to the $i+1$ -th column, the DTW computes $L(\cdot, \cdot)$ and $Ac(\cdot, \cdot)$ only for grid points $(i+1, k)$ such that $\hat{j} - Beam \leq k \leq \hat{j} + Beam$, where *Beam* is the maximum allowed beam width. This process is repeated for each column until the last.

A beam width equivalent to 3 seconds (1.5 s before and after the best point in each column) was used. The clipping of the search area can lead to a drastic acceleration of the execution time, but may not immediately find the correct alignment. To solve this problem, successive iterations of the DTW are run until the alignment function converges (usually 3–4 iterations).

Energy calculation:

In order to accelerate the DTW program, an alternative energy calculation method was used. First, a set *HARM* of all the harmonics of *all* the notes in the piece is constructed (similar to the construction of $harm(j)$). Close harmonics whose frequency difference is less than 2 Hz are unified, resulting in around 300 harmonics.

Before running the DTW, an FFT is run for each audio frame, applying a Hamming window and padding it to a length of 8,192 samples. For each frame, the energy values calculated by the FFT for each of the harmonics in *HARM* are stored in an auxiliary file. When the DTW is run, instead of repeatedly calculating the energy values for each grid point, the previously calculated energy values are used. This greatly reduces the amount of calculation, especially when iterating the alignment (as described above).

6. MANUAL PATH INITIALIZATION

In one of the 10 tested files, the DTW was unable to find the correct alignment, due to large tempo changes in the playing (see Fig. 7). To solve this problem, a manual procedure for the initialization of the alignment was

created.

An audio-graphic interface was used to allow easy input of a manual initialization for the alignment function. The program shows the user the text of the song, divided into separate syllables. The computer replays the audio (piano and singing). At each moment, one of the displayed syllables is highlighted. When hearing the highlighted syllable sung, the user hits a key. The time the key was hit is recorded. Then, another syllable is highlighted, and the process is repeated until the end of the song.

Due to the simplicity of this procedure, the user does not have to be musically literate, and does not need any special skills. The task is not very demanding — it is enough to mark only one or two syllables per song line, and the system is very tolerant of inaccurate time marking of the syllables.

A quick test is run on input recordings in order to know if a manual initialization is necessary — for each file, a few words are selected. Each selected word is displayed, and the part where the alignment indicates it should appear in the recording is played back. If the display and the playback disagree with each other, manual initialization is needed.

After the initial alignment is created, successive iterations of the DTW are run, until the alignment function converges. The dark crosses in Fig. 7 show the manually

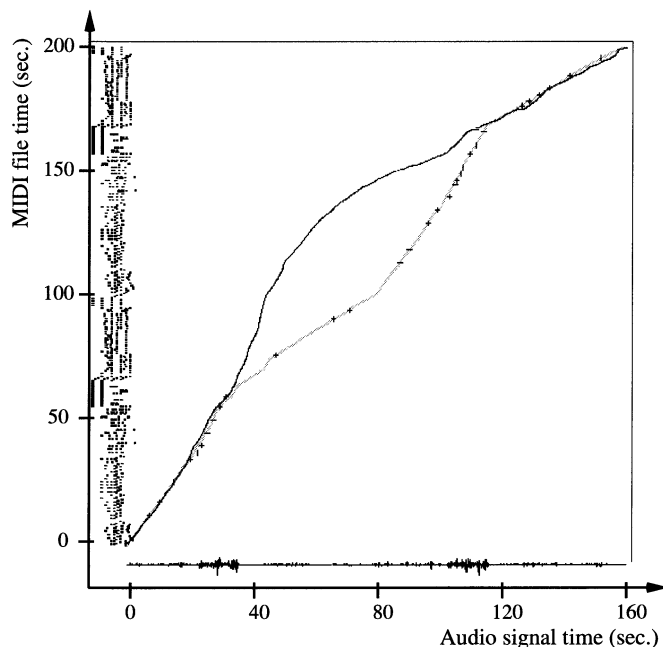


Fig. 7 DTW example where manual initialization is necessary. x axis: audio, y axis: MIDI. Dark line shows failed alignment (no manual initialization). The dark crosses show the manual initialization, and the light line shows the correct alignment, obtained by the DTW using the manual initialization.

created initial alignment. The result of running the DTW using this initialization is shown by the light line.

6.1. Note to Syllable Match

Before the manual alignment described above can take place, it is necessary to know exactly which note matches which syllable, so that the displayed text will match the played audio. Note that this kind of alignment needs to be done only once for each musical piece. Once done, it can be used for the alignment of repeated performances of the same piece.

In principle, this match between notes and syllables is an integral part of the score, but for most available computer readable scores it is given as a separate text file.

For most notes, there is a one-to-one match between notes and syllables. In some cases, one syllable can match several consecutive notes. Rarely, one note can match several consecutive syllables (more often this would be transcribed as a sequence of short repetitions of the same note).

Since manual typing of notes and their corresponding syllables is time consuming, a graphical interface was created (see Fig. 8). In the top window in this figure, the waveform of an audio recording of the song is shown (and can be played back). In the bottom window, the notes of a short segment of the voice part are displayed. The alignment between the notes and the audio is used to ensure that the displayed waveform part matches the displayed notes. The user listens to the recorded voice segment and can then mark syllables which span more than one note (e.g. “wan” in Fig. 8), or notes which match several syllables.

7. ALIGNMENT REFINEMENT

The DTW gives a rough estimate of the note timing, using only a limited time resolution, and assuming that notes which are transcribed as simultaneous were indeed

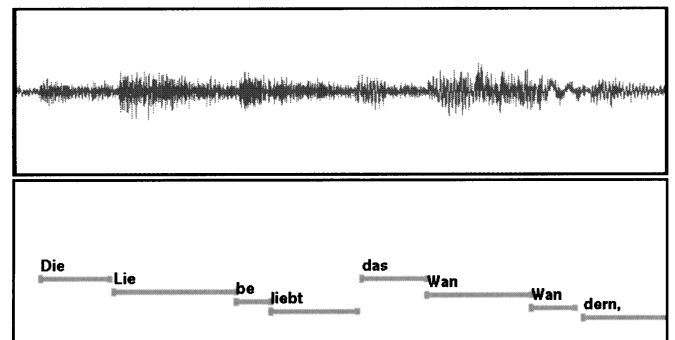


Fig. 8 Graphic tool for marking note to syllable match. The syllable “wan” is matched to two consecutive notes.

played simultaneously. A second pass tries to refine this estimate.

Starting from the output of the DTW, for each individual note, a refinement of its onset detection is looked for, within a window around the DTW's estimate. This is done by trying to detect peaks in the sum of energies of the note's harmonics. Since harmonics of simultaneous notes might interfere with the calculation of this energy sum, a higher weight is given in the summation to harmonics which have no close interfering harmonics.

For each note i , we look at all of its harmonics H_h^i , $h = 1..N_{harm}^i$. For each harmonic, we find the closest harmonic (Cl_h^i) of any of the notes which are supposed to be played simultaneously with note i according to the score.

A reliability weight $Weight_h^i$ is attached to H_h^i according to the frequency difference between H_h^i and Cl_h^i — higher weight for greater frequency difference. We then look at slightly shifted audio segments around the time the DTW predicted that note i was played in the audio file. For each shifted segment a_l , we calculate the weighted energy sum as $E_w(l) = \sum_{h=1}^{N_{harm}^i} Weight_h^i * e_h(l)$, where $e_h(l) = (\sum_{k=1}^{len} a_l(k) \cos(k * fr_h))^2 + (\sum_{k=1}^{len} a_l(k) \sin(k * fr_h))^2$.

If a prominent peak is detected, it is taken as the note's onset. Otherwise, the original estimate is retained.

8. RESULTS

8.1. Jittered MIDI Files Test

A first test of the alignment accuracy is done using jittered MIDI files: A score-MIDI file, $M.mid$, is chosen. A 'jittered' version of this file, $\hat{M}.mid$, is created by changing its tempo and note timing (as described in section 2.3.). $\hat{M}.mid$ is used to create $\hat{M}.pcm$ — a raw audio file recorded from the output of a Roland SC-88 sound synthesis module, set to *piano 1w* patch. The MIDI-to-audio DTW program is run, aligning $\hat{M}.pcm$ to $M.mid$.

An evaluation program evaluates the alignment which was found by the DTW; for each note \hat{i} in $\hat{M}.mid$, we look at the difference between the onset time of \hat{i} and the time to which the alignment function projects the onset of i (the note in $M.mid$ corresponding to \hat{i}).

For evaluating the system results, the significant measure is the error standard deviation, rather than the average error. Running the above evaluation on 10 MIDI files, gave average alignment error standard deviation of 0.034 s and 0.023 s for note onset, for the first and second alignment passes, respectively.

A closer look at the error distribution revealed that most cases of large error occurred at places where the same note, or note combination, was played repeatedly; for example the same chord hits several times in succes-

Table 1 Alignment accuracy — error standard deviation of note onset time detection. Results calculated for 10 pianist-MIDI files, for the first and second alignment passes.

Alignment pass	Err. std. all notes	Err. std. no repetitions
First alignment	0.043 s	0.014 s
Refinement	0.025 s	0.013 s

sion. Excluding these events (which account for up to 30% of the notes in some of the pieces) from the error calculation, the error standard deviation dropped to 0.012 s.

8.2. Pianist-MIDI Test Files

The two alignment passes were run on the same 10 files, this time performed by the pianist. Table 1 shows the alignment accuracy for the first and second alignment passes. The results show that the alignment error was reduced by the second (refinement) pass. The error reduction was almost entirely due to improvements in the detection of onsets of repetitive notes.

8.3. Commercial CD Recording

Running the alignment program on recordings from a commercial CD [11] showed that it can keep track of the music. Visual inspection showed that the alignment errors were of the same order of magnitude (although slightly larger) as for the MIDI-synthesized audio files. A quantitative evaluation of the error calls for marking of note onset times in the audio file (which is a time-consuming manual task), thus only a short part of one file was used.

Onset times of 50 notes were manually marked (with an estimated manual marking accuracy of 5–10 ms) in a part including voice and piano with a low texture density. Comparison to the alignment results gave an alignment error standard deviation of 0.052 s at the end of the second alignment pass.

9. CONCLUSION

Two alignment methods, MIDI-to-MIDI alignment and MIDI-to-audio alignment, were presented in this paper. A DTW algorithm was adapted to perform the alignment, and was shown to improve the robustness of the alignment, in comparison to previously published systems.

For MIDI-to-audio alignment, summation of energy at expected notes' harmonics is used by the DTW to find the alignment function. Alignment of score-MIDI to recordings of a piano synthesizer showed that the DTW is able to keep track of long musical pieces.

A limited test of the system on a commercial musical recording showed that the method may be applied to

“real world” recordings using an acoustic piano in a reverberant environment, and with the presence of a singer, although with reduced accuracy.

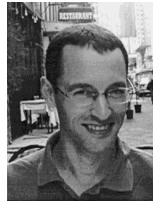
For the needs of our automatically trainable singing synthesis system (i.e., the restriction of automatic phonetic segmentation and pitch estimation processes, and the initialization of a signal separation process), the accuracy of the alignment is acceptable.

An approach to refining the alignment, which seems promising, is to use a method of note shape modeling, e.g. like the one described in [3], to improve the identification of note locations.

Another possible refinement is to try and track the singer's pitch in the audio recording (e.g. using a method along the lines of [12]) and use that to compensate for the interference of the singer's energy with the piano harmonics.

REFERENCES

- [1] Y. Meron, “High quality singing synthesis using the selection-based synthesis scheme”, *Ph.D. thesis, The University of Tokyo* (1999), <http://www.gavo.t.u-tokyo.ac.jp/~meron/HomePage.html>
- [2] B. H. Repp, “The art of inaccuracy: Why pianists' errors are difficult to hear”, *Music Percept.* **14**, 161–184 (1996).
- [3] Y. Meron, K. Hirose, “Separation of singing and piano sounds”, *Proc. Int. Conf. on Spoken Language Processing III*, 1059–1062 (1998).
- [4] MIDI manufacturers association web site, <http://www.midi.org/>
- [5] M. Briessinck, Winterreise web site, <http://www.gopera.com/winterreise/>
- [6] B. H. Repp, “Patterns of note onset asynchronies in expressive piano performance”, *J. Acoust. Soc. Am.* **100**, 3917–3932 (1996).
- [7] L. Rabiner and B. H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall signal processing series (Prentice Hall, Englewood Cliffs, 1993).
- [8] J. Bilmes, “Timing is the essence: Perceptual and computational techniques for representing, learning and reproducing expressive timing in percussive rhythm”, *Master's thesis, MIT Media Lab.* (1993).
- [9] E. D. Scheirer, “Using musical knowledge to extract expressive performance information from audio recording”, in *Readings in Computational Auditory Scene Analysis*, D. Rosenthal and H. Okuno, Eds. (Lawrence Erlbaum, Mahwah, NJ, 1998).
- [10] A. Galembo and A. Askenfelt, “Signal representation and estimation of spectral parameters by inharmonic comb filters with application to the piano”, *IEEE Trans. Speech Audio Process.* **7**(2), 197–203 (1999).
- [11] D. Fischer-Dieskau (baritone) and G. Moore (piano), “Winterreise”, Deutsche Grammophon, 429-970-2 (1972).
- [12] Ph. Depalle, G. Garcia and X. Rodet, “Tracking of partials for additive sound synthesis using hidden Markov models”, *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing I*, 225–228 (1993).



Yoram Meron Was born on January 1966 in Haifa, Israel. He received a B.Sc. in physics, mathematics and computer science from the Hebrew University in Jerusalem, Israel in 1987. After 4 years work on speech-recognition, he was granted a scholarship from the Japanese Ministry of Education (Monbusho), under which he had received the M.Sc. degree (applications of speech technology to foreign language teaching), and Ph.D. degree (speech synthesis) from the University of Tokyo, in the Department of Information and Communication Engineering, in 1996 and 1999 respectively. From 1999 to 2000 he held a post-doctoral position in speech synthesis at the Science University of Tokyo. In 2000 he has joined the Panasonic speech technology lab (PSTL) at Santa Barbara, California. Dr. Meron is a member of the International Speech Communication Association and the Acoustic Society of Japan.



Keikichi Hirose received the B.E. degree in electrical engineering in 1972, and the M.E. and Ph.D. degrees in electronic engineering in 1974 and 1977, respectively, from the University of Tokyo, Tokyo, Japan. In 1977, he joined the University of Tokyo as a Lecturer in the Department of Electrical Engineering. He has been a Professor at the Graduate School of Engineering, Department of Information and Communication Engineering, University of Tokyo, since April 1994. In April 1999, he received a dual appointment as Professor in the University's Graduate School of Frontier Sciences. From March 1987 until January 1988, he was a Visiting Scientist of the Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge. His research interests include speech signal processing. Dr. Hirose is a member of the Institute of Electrical and Electronics Engineers, the Acoustical Society of America, the International Speech Communication Association, the Institute of Electronics, Information and Communication Engineers, the Information Processing Society of Japan and other professional organizations.