# *TempoExpress*, a CBR Approach to Musical Tempo Transformations

Maarten Grachten, Josep Lluís Arcos, and Ramon López de Mántaras

IIIA, Artificial Intelligence Research Institute,
CSIC, Spanish Council for Scientific Research,
Campus UAB, 08193 Bellaterra, Catalonia, Spain,
{maarten,arcos,mantaras}@iiia.csic.es,
http://www.iiia.csic.es

**Abstract.** In this paper, we describe a CBR system for applying musically acceptable tempo transformations to monophonic audio recordings of musical performances. Within the tempo transformation process, the expressivity of the performance is adjusted in such a way that the result sounds natural for the new tempo. A case base of previously performed melodies is used to infer the appropriate expressivity. Tempo transformation is one of the audio post-processing tasks manually done in audio-labs. Automatizing this process may, therefore, be of industrial interest.

## 1 Introduction

In this paper we describe a CBR system, *TempoExpress*, that automatically performs musically acceptable tempo transformations. This paper significantly extends previous work [1], that addressed the process of performance annotation, a basic step to construct the cases needed in the CBR system described now.

The problem of changing the tempo of a musical performance is not as trivial as it may seem. When a musician performs a musical piece at different tempos, the performances are not just time-scaled versions of each other, as if the same performance were played back at different speeds. Together with the changes of tempo, variations in musical expression are made [3]. Such variations do not only affect the timing of the notes, but can involve for example the addition or deletion of ornamentations, or the consolidation/fragmentation of notes. Apart from the tempo, other domain specific factors seem to play an important role in the way a melody is performed, such as meter, and phrase structure.

Tempo transformation is one of the audio post-processing tasks manually done in audio-labs. Automatizing this process may, therefore, be of industrial interest.

In section 2, we will present the overall structure of *TempoExpress*. In section 3, we briefly explain the processes involved in case and problem representation. Section 4 describes the crucial problem solving phases of the CBR mechanism, retrieval and reuse. In section 5, some initial results are presented. Conclusions and future work are presented in section 6.

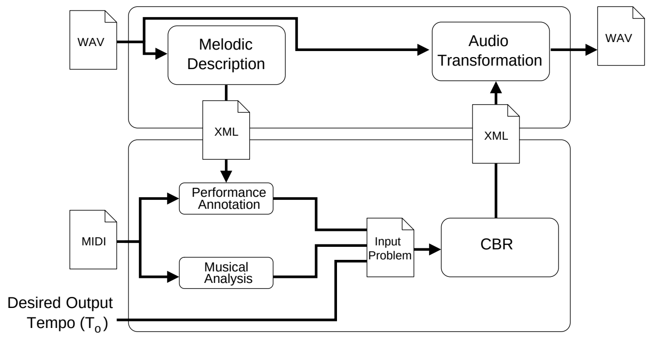

**Fig. 1.** Overview of the basic *TempoExpress* components

## 2 Overview of *TempoExpress*

*TempoExpress* consists of three main parts and two additional parts. The main parts are the melodic description module, the CBR problem solving module, and the audio transformation module. The additional parts are the performance annotation module and the musical analysis module (see figure 1). The melodic description module generates a melodic description of the input recording, that represents information about the performance on a musical level. This information is used together with the score of the performed melody (as a MIDI file), and the desired tempo of the output performance, to construct an input problem. CBR is then applied to obtain a solution for the problem in the form of a melodic description of the new performance. The audio transformation produces an audio file, based on the original audio and the new melodic description.

Since the main information in the input problem and the cases (the melodic material of the score and the annotated performance) is of sequential nature, we apply edit distance techniques in the retrieval step, as a means to assess similarities between the cases and the input problem. In the reuse step we employ constructive adaptation [13], a reuse method for synthetic tasks. This method constructs a solution to a problem by searching the space of partial solutions for a complete solution that satisfies the solution requirements of the problem.

### 2.1 Melodic Description and Audio Transformation

The melodic description and audio transformation are not part of the research reported here. These processes are being implemented within a common research project by members of the Music Technology Group (MTG) of the Pompeu Fabra University, using signal spectral modeling techniques (see [6] for a detailed description). The output of the melodic description process (and input of the audio transformation process), is a description of the audio in *XML* format, that adheres to (and extends) the *MPEG7* standard for multimedia description [5]. This description includes information about the starting and ending of notes, their pitches and amplitudes.

## 3 Case/Problem Representation

In this section, we will explain the various aspects of the construction of cases from available information. To construct a case, a score (in MIDI format) is needed. This score is represented internally as a sequence of note objects, with the basic attributes like pitch, duration and temporal position. This score is analyzed automatically to obtain a more abstract representation of the melody, called I/R representation. This procedure is explained in subsection 3.1. Furthermore, an input performance at a particular tempo is needed. The performance is not stored literally, but rather a *performance annotation* is constructed to describe how the elements from the performance relate to the elements from the score. This procedure is explained in detail in [1], and is briefly reminded in subsection 3.2. The performance annotation is stored as a solution, associated to a particular input description that applies to the performance (in our case, the tempo of the performance). Lastly, the desired output tempo is also included as a part of the problem description, specifying what the solution should be like.

### 3.1 Music Analysis

To prepare cases, as well as the input problem, music analysis is performed on the musical score that was provided. The analysis is used in the problem solving process, for example to segment musical phrases into smaller groups of notes, and to perform retrieval of cases. The musical analysis is based on a model for melodic structure, that is explained below.

**The Implication/Realization Model** Narmour [11,12] has proposed a theory of perception and cognition of melodies, the Implication/Realization model, or I/R model. According to this theory, the perception of a melody continuously causes listeners to generate expectations of how the melody will continue. The sources of those expectations are two-fold: both innate and learned. The innate sources are 'hard-wired' into our brain and peripheral nervous system, according to Narmour, whereas learned factors are due to exposure to music as a cultural phenomenon, and familiarity with musical styles and pieces in particular. The innate expectation mechanism is closely related to the *gestalt theory* for visual perception [9]. Gestalt theory states that perceptual elements are (in the process of perception) grouped together to form a single perceived whole (a 'gestalt'). This grouping follows certain principles (*gestalt principles*). The most important principles are *proximity* (two elements are perceived as a whole when they are perceptually close), *similarity* (two elements are perceived as a whole when they have similar perceptual features, e.g. color or form, in visual perception), and *good continuation* (two elements are perceived as a whole if one is a 'good' or 'natural' continuation of the other). Narmour claims that similar principles hold for the perception of melodic sequences. In his theory, these principles take the form of *implications*: Any two consecutively perceived notes constitute a melodic interval, and if this interval is not conceived as complete, or closed, it is

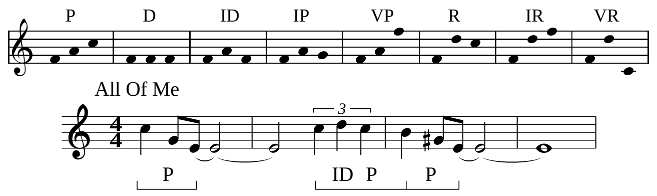

**Fig. 2.** Top: Eight of the basic structures of the I/R model. Bottom: First measures of All of Me, annotated with I/R structures

an *implicative interval*, an interval that implies a subsequent interval with certain characteristics. In other words, some notes are more likely to follow the two heard notes than others. Two main principles concern *registral direction* and *intervallic difference*. The principle of registral direction states that small intervals imply an interval in the same registral direction (a small upward interval implies another upward interval, and analogous for downward intervals), and large intervals imply a change in registral direction (a large upward interval implies a downward interval and analogous for downward intervals). The principle of intervallic difference states that a small (five semitones or less) interval implies a similarly-sized interval (plus or minus 2 semitones), and a large intervals (seven semitones or more) implies a smaller interval. The definitions of 'small', 'large', and 'similarly sized' intervals are specified by the I/R model [11].

Based on these two principles, melodic patterns can be identified that either satisfy or violate the implication as predicted by the principles. Such patterns are called *structures* and labeled to denote characteristics in terms of registral direction and intervallic difference. Eight such structures are shown in figure 2(top). For example, the P structure ('Process') is a small interval followed by another small interval (of similar size), thus satisfying both the registral direction principle and the intervallic difference principle. Similarly the IP ('Intervallic Process') structure satisfies intervallic difference, but violates registral direction.

Additional principles are assumed to hold, one of which concerns *closure*, which states that the implication of an interval is inhibited when a melody changes in direction, or when a small interval is followed by a large interval. Other factors also determine closure, like metrical position (strong metrical positions contribute to closure, rhythm (notes with a long duration contribute to closure), and harmony (resolution of dissonance into consonance contributes to closure). The closure in each of these dimensions add up to the total closure. The occurrence (and degree) of closure at a given point in the melody determines where the structures start and end. For example, on a note where strong closure appears (e.g. closure in meter, harmony and rhythm at the same time), the interval between that note and the next will not be perceived as implicative, and therefore there is no structure describing that interval. When no closure occurs at all, every interval implies a new interval, and since the structures describe two subsequent intervals, this causes a *chaining*, or overlapping of structures.

We have designed an algorithm to automate the annotation of melodies with their corresponding I/R analyses. The algorithm implements most of the ‘innate’ processes mentioned before. The learned processes, being less well-defined by the I/R model, are currently not included. Nevertheless, we believe that the resulting analysis have a reasonable degree of validity, since the analyses generated for melodic examples given in [11] were in many cases identical to the analyses proposed by Narmour. An example analysis is shown in figure 2(bottom). This example shows various degrees of structure chaining: the first two structures (P and ID) are not chained, due to strong closure (meter and rhythm); the second pair of structures (ID and P) are strongly chained (sharing two notes, one interval), because closure is inhibited by ‘ongoing’ rhythms (like triplets); the last pair of structures (P and P) are chained by one note, because of weak closure (only in meter).

## 3.2 Performance Annotation

In addition to the score and its musical analysis, the cases in the case base, as well as the problem specification, contain a performance of that score by a musician. The raw format of the performance is an audio file. Using the melodic description mechanism described in section 2.1, we obtain a melodic description of the audio, in XML format. This description contains a sequence of note descriptors, that describe the features like start and end times, pitch, energy of the notes, as they were detected in the audio file. In order to be informative, the sequence of note descriptors is to be mapped to the notes in the score, since this mapping expresses how the score was performed. For example, it allows us to say that a particular note was lengthened or shortened, or played early or late.

But the mapping between score notes and performed notes does not necessarily consist of just 1-to-1 mappings. Especially in jazz performances, which is the area on which we will focus, performers often favor a ‘liberal’ interpretation of the score. This does not only involve changes in expressive features (like lengthening/shortening durations) of the score elements as they are performed, but also omitting or adding notes. Thus, one can normally not assume that the performance contains a corresponding element for every note of the score, neither that every element in the performance corresponds to a note of the score. Taking these performance liberties into account, a description of a musical performance could take the form of a sequence of *performance events*, that represent the phenomena like note deletions or additions that occured in the performance.

From this perspective the edit distance [10] is very useful, since performance events can be mapped in a very natural way to edit operations for sequences of score and performance elements. A performance annotation can then be obtained in the form of a sequence of performance events, by constructing the optimal alignment between a score and a performance, using the edit distance. The set of performance events/edit operations we use is a slight revision of the set proposed by Arcos et al. [1]. It includes:

**Transformation** Representing the reproduction of a score note, possibly with several kinds of transformations, such as change of pitch, duration and temporal position

**Insertion** Representing the occurrence of a performance note that does not correspond to any score note

**Ornamentation** A special case of insertion, where the inserted note (or possibly more than one) has very short duration, and is played as a lead-in to the next note

**Deletion** Representing the occurrence of a score note that does not correspond to any performance note

**Fragmentation** Representing the reproduction of a score note by playing two or more shorter notes (adding up to the same total duration)

**Consolidation** Representing the reproduction of two or more score notes by playing a single longer note (whose duration equals the sum of the score note durations)

We defined the costs of these operations as functions of the note attributes (pitch, duration and onset). However, rather than fixing the relative importance of the attributes (as in [1]), we parametrized the cost functions to be able to control the importance of each of the note attributes in each of the cost functions, and the relative costs of edit operations. This setup enables us to tune the performance annotation algorithm to produce annotations that correspond to intuitive human judgment. We have used a genetic algorithm [8] to tune the parameters of the cost functions, which substantially improved the accuracy of annotation over untuned settings.

## 4 Problem Solving

In this section, we will explain the steps taken to transform the performance presented as input into a performance of the same score at a different tempo. The first step is the retrieval of relevant cases from the case base. In the second step, the retrieved cases are selectively used to obtain a new sequence of performance events. This sequence can then be used to modify the XML description of the performance. Based on this modified description, the original audio file is transformed to obtain the final audio of the performance at the desired tempo.

### 4.1 Retrieval

The goal of the retrieval step is to form a pool of relevant cases, that can possibly be used in the reuse step. This done in the following three steps: firstly, cases that don't have performances at both the input tempo and output tempo are filtered out; secondly, those cases are retrieved from the case base that have phrases that are I/R-similar to the input phrase; lastly, the retrieved phrases are segmented. The three steps are described below.

**Case filtering by tempo** In the first step, the case base is searched for cases that have performances both at the tempo the input performance was played, and the tempo that was specified in the problem description as the desired output tempo. The matching of tempos need not be exact, since we assume that there are no drastic changes in performance due to tempo within small tempo ranges. For example, a performance played at 127 beats per minute (bpm) may serve as an example case if we want to construct a performance at 125 bpm.

**I/R based retrieval** In the second step, the cases selected in step 1 are assessed for melodic similarity to the score specified in the problem description. In this step, the primary goal is to rule out the cases that belong to different styles of music. For example, if the score in the problem description is a ballad, we want to avoid using a bebop theme as an example case. Note that the classification of musical style based on just melodic information (or derived representations) is far from being an established issue. Nevertheless, there is some evidence [7] that the comparison of melodic material at different levels of abstraction yields different degrees of discriminatory power. For example comparing on the most concrete level (comparing individual notes) is a good way to find out which melodies in a set are nearly identical to a particular target melody. But if the set of melodies does not contain a melody nearly identical to the target, the similarity values using this measure are not very informative, since they are highly concentrated in a single value. On the other hand, comparisons based on more abstract descriptions of the melody (e.g. melodic contour, or I/R analyses), tend to produce a distribution of similarity values that is spread out through the spectrum more equally. Thus, these measures tell us in a more informative way *how* similar two melodies are (with respect to the other melodies in the set), even if they are considerably different. As a consequence, a melodic similarity measure based on an abstract representation of the melody seems a more promising approach to separate different musical styles.

We use the I/R analysis of the melodies to assess similarities. The measure used is an edit distance. The edit distance measures the minimal cost of transforming one sequence of objects into another, given a set of edit operations (like insertion, deletion, and replacement), and associated costs. We have defined edit operations and their corresponding costs for sequences of I/R structures (see [7] for more details). The case base is ranked according to similarity with the target melody, and the subset of cases with similarity values above a certain threshold are selected. The resulting set of cases will contain phrases that are roughly similar to the input score.

**Segmentation** In this step, the melodies that were retrieved in the second step are segmented. The motivation for this twofold. Firstly, using complete melodic phrases as the working unit for adaptation is inconvenient, since a successful adaptation will then require that the case base contains phrases that are nearly identical as a whole to the input phrase. Searching for similar phrase *segments* will increase the probability of finding a good match. Secondly, the segmentation

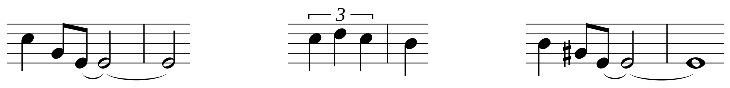

**Fig. 3.** Segmentation of the first phrase of 'All of Me', according to I/R structures. The segments correspond to single I/R structures, or sequences of structures if they are strongly chained (see subsection 3.1)

is motivated by the intuition that the way a particular note is performed does not only depend of the attributes of the note in isolation, but also on the musical context of the note. Therefore, rather than trying to reuse solutions in a note-by-note fashion, it seems more reasonable to perform the reuse segment by segment. This implies that the performance of a retrieved note is only reused for a note of the input phrase if their musical contexts are similar.

Melodic segmentation has been addressed in a number of studies (e.g. [16][2]), with the aim of detecting smaller musical structures (like motifs) within a phrase. Many of them take a data driven approach, using information like note interonset intervals (IOI) and metrical positions to determine the segment boundaries. Our method of segmentation is based on the I/R representation of the melodies. This may seem quite different from the approach mentioned above, but in essence it is similar. The melodies are split at every point where the overlap of two I/R structures is less than two notes (see subsection 3.1). This overlap is determined by the level of closure, which is on its turn determined by factors like metrical posisiton and IOI. The resulting segments usually correspond to the musical motifs that constitute the musical phrase, and are used as the units for the stepwise construction of the output performance. As an example, figure 3 displays the segmentation of the first phrase of 'All of Me' (the complete phrase is shown in figure 2).

### 4.2 Reuse

In the reuse step a performance of the input score is constructed at the desired tempo, based on the input performance and the set of retrieved phrase segments. This step is realized using constructive adaptation [13], a technique for reuse that constructs a solution by a best-first search through the space of partial solutions. In this subsection, we will first explain briefly how the reuse step can in general be realized as best-first search, and then we will explain how we implemented the functions necessary to make the search-algorithm operational in the context of performance transformation.

In constructive adaptation, partial solutions of the problem are represented as states. Furthermore, a function `HG` must be defined for generating a set of successor states for a given state. The state space that emerges from this function and the state that represents the empty solution (generated by a function `Initial-State`), is then searched for a complete solution that satisfies certain constraints (through a function `Goal-Test`). The resulting state is transformed to a real solution by a function `SAC`. The order of expansion of states is controlled by a function `HO` that orders the states in a best-first manner. The search process is expressed in pseudo code below.

```
Initialize OS = (list (Initial-State Pi))
Function CA(OS)
      Case (null OS) then No-Solution
      Case (Goal-Test (first OS)) then (SAC (first OS))
      Case else
            Let SS = (HG (first OS))
            Let OS = (HO (append SS (rest OS)))
                  (CA OS)
```

**Fig. 4.** The search process of constructive adaptation expressed in pseudo code. Functions `HG` and `HO` are Hypotheses Generation and Hypotheses Ordering. Variables *OS* and *SS* are the lists of Open States and Successor States. The function `SAC` maps the solution state into the configuration of the solution. The function `Initial-State` maps the input problem description *Pi* into a state. From Plaza and Arcos [13]

We explain our implementations of the functions `Initial-State`, `HG`, `HO`, `Goal-Test`, and `SAC` below.

**Initial-State** The function Initial-State returns a state that is used as the starting point for the search. It takes the input problem description (the score, analysis, input-performance, and desired output tempo) as an argument. In our case, the state contains a sequence of score segments, and a slot for storing the corresponding performance segments (none of which is filled in the initial state, obviously). Furthermore, there is a slot that stores the quality of the partially constructed performance, as a number. We will explain the derivation of this number in the next subsection. Figure 5 shows the initial state for a short musical fragment (containing two segments).

**Hypothesis-Generation (HG)** The Hypothesis-Generation function takes a state as an argument and tries to find a sequence performance events for one of the unprocessed score segments in the state. We will illustrate this procedure step by step, using the first segment of the initial state in figure 5 as an example. The steps are presented graphically in figure 7 (at the last page of this paper).

The *first step* is to find the segment in the pool of retrieved melodic segments that is most similar to the input score segment. The similarity is assessed by calculating the edit distance between the segments (the edit distance now operates on notes rather than on I/R structures, to have a finer grained similarity assessment). A mapping between the input score segment and the best matching retrieved segment is made.

In the *second step*, the performance annotation events (see subsection 3.2 and [1]) corresponding to the relevant tempos are extracted from the retrieved segment case and the input problem specification (both the input tempo $T_i$ and

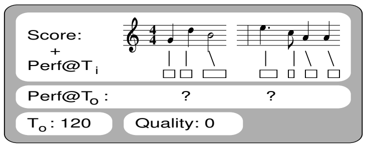

**Fig. 5.** Example of an initial state in Constructive Adaptation. $T_i$ is the tempo of the input performance; $T_o$ is the desired output tempo

the output tempo $T_o$ for the retrieved segment case, and just $T_i$ from the input problem specification).

The *third step* consists in relating the annotation events of the retrieved segment to the notes of the input segment, according to the mapping between the input segment and the retrieved segment, that was constructed in the first step. For the notes in the input segment that were mapped to one or more notes in the retrieved segment, we now obtain the tempo transformation from $T_i$ to $T_o$ that was realized for the corresponding notes in the retrieved segment. It is also possible that some notes of the input segment could not be matched to any notes of the retrieved segment. For such notes, the retrieved segment can not be used to obtain annotation events for the output performance. Currently, these gaps are filled up by directly transforming the annotation events of the input performance (at tempo $T_i$) to fit the output tempo $T_o$ (by scaling the duration of the events to fit the tempo). In the future, more sophisticated heuristics may be used.

In the *fourth step*, the annotation events for the performance of the input score at tempo $T_o$ are generated. This is done in a note by note fashion, using rules that specify which annotation events can be inferred for the output performance of the input score at $T_o$, based on annotation events of the input performance, and the annotation events of the retrieved performances (at $T_i$ and $T_o$). To illustrate this, let us explain the inference of the Fragmentation event for the last note of the input score segment (B)in figure 7. This note was matched to the last two notes (A, A) of the retrieved segment. These two notes were played at tempo $T_i$ as a single long note (denoted by the Consolidation event), and played separately at tempo $T_o$. The note of the input segment was also played as a single note at $T_i$ (denoted by a Transformation event rather than a Consolidation event, since it corresponds to only one note in the score). To imitate the effect of the tempo transformation of the retrieved segment (one note at tempo $T_i$ and two notes at tempo $T_o$), the note in the input segment is played as two shorter notes at tempo $T_o$, which is denoted by a Fragmentation event (F).

In this way, adaptation rules were defined, that describe how the tempo transformation of retrieved elements can be translated to the current case. In figure 7, two such rules are shown. If the antecedent part matches the constellation of annotation events, the tempo transformation in the consequent part can be applied. It can occur that the set of rules contains no applicable rule for a particular constellation, in particular when the performances at $T_i$ of the retrieved note and the input note are too different. For example, if the score note is played as a Transformation event, but the retrieved note is deleted in the performance at $T_i$, then the performances are too different to make an obvious translation. In this case, the annotation events from the input performance are transformed in the same way as in the case where no corresponding note from the retrieved segment could be found (see the third step of this subsection).

The mismatch between the input segment and the retrieved segment and the inability to find a matching adaptation rule obstructs the use of case knowledge to solve the problem and forces *TempoExpress* to resort to default mechanisms. This will affect the quality of the solution. To reflect this, the value of the quality slot of the state (see figure 5) is calculated as the number of input score notes for which annotation events could be inferred from retrieved cases, divided by the total number of notes processed so far (that is, the sum of all notes in the processed input segments, including the current input segment).

**Hypothesis-Ordering (HO)** The Hypothesis-Ordering function takes a list of states (each one with its partial solution) and orders them so that the states with the most promising partial solutions come first. For this ordering, the quality value of the states is used. In our current implementation, the quality value is only determined by one factor, roughly the availability of appropriate cases. Another factor that should ideally influence the quality of the states is the 'coherence' of the solution. For example, if the notes at the end of one segment were anticipated in time (as a possible effect of a Transformation event), then anticipation of the first notes of the next segment will not have the typical effect of surprise, since the listener will experience the performance as being shifted forward in time, instead of hearing a note earlier than expected. We are currently incorporating the detection and evaluation of such phenomena into the Hypothesis-Ordering function, so that this functionality will soon be available.

**Goal-Test** The Goal-Test function is called on the best state of an ordered list of states to test if the solution of that state is complete and satisfies the constraints imposed upon the desired solution. The completeness of the solution is tested by checking if all segments of the input score have a corresponding segment in the performance annotation for the output tempo. The constraints on the solution are imposed by requiring a minimal quality value of the state. In our case, where the quality value represents the ratio of notes for which annotation events were obtained using retrieved cases (a value between 0 and 1), the quality value is required to be superior or equal to 0.8.

**State-to-Solution (SAC)** The State-to-Solution function takes the state that passed the goal-test and returns a solution to the input problem. This step consists in building a complete performance annotation from the annotation events for the score segments (basically concatenation of the events). The new performance annotation is used to adapt the XML description of the original audio file, by changing attribute values, and possibly deleting and inserting new note descriptors. Finally, the audio transformation module (which is under development) generates a new audio file, based on the new XML description.

### 4.3 Retain

When the solution that was generated is satisfying to the listener, and when the quality of the solution is high (that is, default adaptation operations have been scarcely used, or not at all), it is retained as a case that includes the input score, the input performance, and the newly generated performance.

## 5 Results

Although the *TempoExpress* is operational, there are some components that need improvement. In particular, the case base is still of limited size (it contains ten different phrases from three different songs, played at approximately ten different tempos). Nevertheless, some good results were obtained for some melodies. We have performed a tempo transformation of a phrase from *Once I Loved* (A.C. Jobim). The original performance of the phrase was at a tempo of 55 beats per minute (bpm), and using the CBR system, the performance was transformed to a tempo of 100 bpm. For comparison, the tempo transformation was also realized using uniform time stretching of the original sound file (i.e. the durations of all notes in the original performance are lengthened by a single scaling factor, while leaving the pitches of the notes unchanged). Figure 6 shows the audio signals of the original sound, and the two transformations. Notable differences between the two transformations occur in the notes 3 to 9 (the numbered vertical lines in the views indicate the start of the notes). Note that in the CBR transformation, the eighth note is missing, due to a consolidation. Furthermore, those notes have considerable variations of duration in the CBR transformation, whereas they are more regularly played in the uniformly time stretched version (as in the original), making the latter sound somewhat mechanical at the faster tempo. Slight changes in the dynamics can also be observed, e.g in note 1 and 12. The sound files from the example are publicly available in mp3 format, through the world-wide web[1].

## 6 Conclusions and Future Work

In this paper, we have described *TempoExpress*, an application for applying musically acceptable tempo transformations to monophonic audio recordings of musical performances. *TempoExpress* has a rich description of the musical expressivity of the performances, that includes not only timing deviations of performed score notes, but also represents more rigorous kinds of expressivity such as note ornamentation, and consolidation. Within the tempo transformation process, the expressivity of the performance is adjusted in such a way that the

[1] `http://www.iiia.csic.es/~maarten/cbr/tempo-transformation`

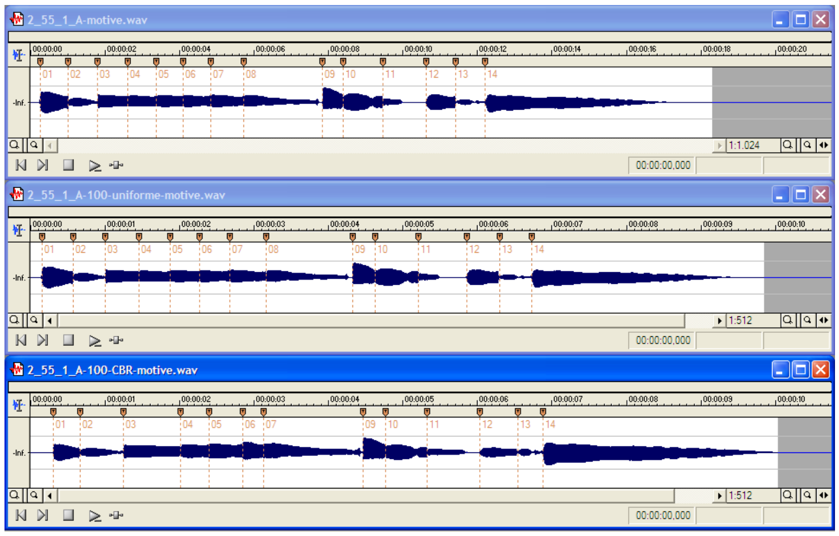

**Fig. 6.** Audio signals of a part of the first phrase of *Once I Loved*. The upper view shows original sound file (55 bpm), the middle view shows a tempo transformation by uniform time stretching, and the lower view shows a tempo transformation using the CBR system. The vertical lines indicate the positions of the note onsets

result sounds natural for the new tempo. A case base of previously performed melodies is used to infer the appropriate expressivity.

Future work includes elaborating the reuse step, to put more musical constraints on the way in which partial solutions can be combined. Also, we intend to add more cases to the case base, to broaden the range of problems that can be satisfyingly solved by the system. Finally, a more thorough evaluation of the results is necessary. This could be done for example by quantitatively comparing transformed performances to performances at the final tempo by a musician, or by a blinded evaluation of performances by a panel.

## 6.1 Related Work

In the field of expressive music performance generation, Widmer [17] has taken a data mining approach to discover rules that match expressive phenomena to musical patterns. Friberg et al. [4] have proposed a set of performance rules that was constructed with the help of musical experts. Serra et al. [14] have applied a CBR approach to expressive music performance. They designed *SaxEx*, a system for adding expressiveness to inexpressive performances of melodies. Some design choices in *TempoExpress* were adapted from this application. Suzuki has recently presented *Kagurame*, a CBR system for the expressive performance of a musical score [15]. All of the above approaches either generate expressive performances only based on a score, or apply a transformation to an inexpressive performance (*SaxEx*). Thus, as opposed to *TempoExpress*, they don't consider any expressive information as input to the system.

**Acknowledgments** This research has been partially supported by the Spanish Ministry of Science and Technology under the project TIC 2003-07776-C2-02 "CBR-ProMusic: Content-based Music Processing using CBR" and EU-FEDER funds. The authors acknowledge the Music Technology Group of the Pompeu Fabra University for providing the melodic description and audio transformation modules.

## References

1. J. Ll. Arcos, M. Grachten, and R. López de Mántaras. Extracting performer's behaviors to annotate cases in a CBR system for musical tempo transformations. In *Proceedings of the Fifth International Conference on Case-Based Reasoning (ICCBR-03)*, 2003.
2. E. Cambouropoulos. The local boundary detection model (lbdm) and its application in the study of expressive timing. In *Proceedings of the International Computer Music Conference (ICMC'2001)*, Havana, Cuba, 2001.
3. P. Desain and H. Honing. Tempo curves considered harmful. In "Time in contemporary musical thought" J. D. Kramer (ed.), Contemporary Music Review. 7(2), 1993.
4. A. Friberg. Generative rules for music performance: A formal description of a rule system. *Computer Music Journal*, 15 (2):56–71, 1991.
5. E. Gómez, F. Gouyon, P. Herrera, and X. Amatriain. Using and enhancing the current mpeg-7 standard for a music content processing tool. In *Proceedings of Audio Engineering Society, 114th Convention*, Amsterdam, The Netherlands, 2003.
6. E. Gómez, A. Klapuri, and B. Meudic. Melody description and extraction in the context of music content processing. *Journal of New Music Research*, 32(1), 2003.
7. M. Grachten, J. Ll. Arcos, and R. López de Mántaras. A comparison of different approaches to melodic similarity, 2002. Second International Conference on Music and Artificial Intelligence (ICMAI).
8. M. Grachten, J. Ll. Arcos, and R. López de Mántaras. Evolutionary optimization of music performance annotation. In *CMMR 2004*, Lecture Notes in Computer Science. Springer, 2004. To appear.
9. K. Koffka. *Principles of Gestalt Psychology.* Routledge & Kegan Paul, London, 1935.
10. V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707–710, 1966.
11. E. Narmour. *The Analysis and cognition of basic melodic structures : the implication-realization model.* University of Chicago Press, 1990.
12. E. Narmour. *The Analysis and cognition of melodic complexity: the implication-realization model.* University of Chicago Press, 1992.
13. E. Plaza and J. Ll. Arcos. Constructive adaptation. In Susan Craw and Alun Preece, editors, *Advances in Case-Based Reasoning*, number 2416 in Lecture Notes in Artificial Intelligence, pages 306–320. Springer-Verlag, 2002.
14. X. Serra, R. Lopez de Mantaras, and J. Ll. Arcos. Saxex : a case-based reasoning system for generating expressive musical performances. In *Proceedings of the International Computer Music Conference 1997*, pages 329–336, 1997.
15. T. Suzuki. The second phase development of case based performance rendering system "Kagurame". In *Working Notes of the IJCAI-03 Rencon Workshop*, pages 23–31, 2003.
16. D. Temperley. *The Cognition of Basic Musical Structures.* MIT Press, Cambridge, Mass., 2001.
17. G. Widmer. Machine discoveries: A few simple, robust local expression principles. *Journal of New Music Research*, 31(1):37–50, 2002.

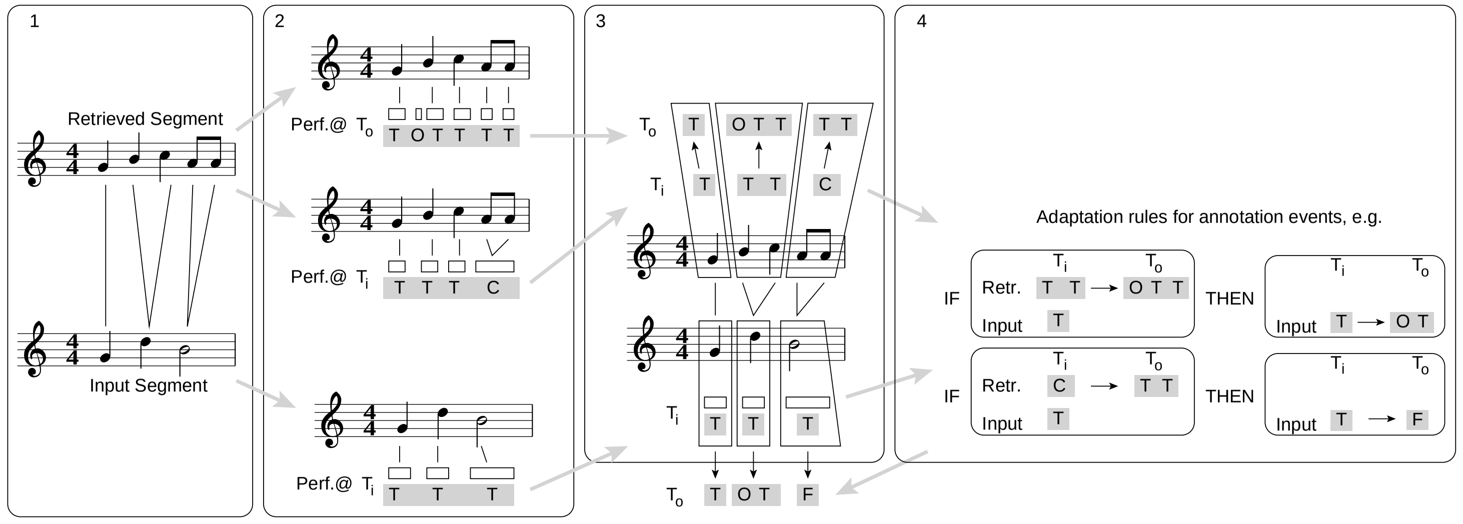

**Fig. 7.** The process of hypothesis generation. In step 1, a mapping is made between the input score segment and the most similar segment from the pool of retrieved segments. In step 2, the performance annotations for the tempos $T_i$ and $T_o$ are collected. In step 3, the performance annotation events are grouped according to the mapping between the input score and retrieved score. In step 4, the annotation events are processed through a set of rules to obtain the annotation events for a performance at tempo $T_o$ of the input score segment