Stefan Müller and Guerino Mazzola
Multimedia Lab
Computer Science Department
University of Zürich
Winterthurerstr. 190, 8057 Zürich, Switzerland
s.mueller@computer.org
guerino@mazzola.ch

The Extraction of Expressive Shaping in Performance

In Mazzola and Zahorka (1994a), a general theory of performance transformations $\wp$ from a symbolic score space S to a corresponding physical space P was given. The transformation $\wp$ can be represented by *performance vector fields* that generalize the well-known hierarchies of tempo curves of a performance. This theory has been implemented as a module in the music research software RUBATO® (Mazzola and Zahorka 1994b) and has been successfully applied to the performance of classical compositions. Stange-Elbe (1999) has performed Contrapunctus III of J. S. Bach's *Die Kunst der Fuge* with RUBATO, and the results have been presented at the Diderot Conference on Mathematics and Music at IRCAM (Mazzola 2002c). The original theory dealt with the construction of a performance from a given score and a given performance field.

In this article, we address the question of *inverse performance* (Mazzola 1995) and how expressive shaping can be extracted from a given performance. The problem can be split into two parts: first, the problem of matching symbolic and performance events, and second, the calculation of the performance field after matching has been performed. In previous literature, the matching problem has been extensively discussed (e.g., Dannenberg 1984; Vercoe 1984; Puckette and Lippe 1992; Heijink et al. 2000). The approach to calculation of the performance field and the visualization of the extracted expressive shaping are new and thus receive main focus throughout this article.

EspressoRubette® is a software module that realizes the theory given in this article. The module is able to process a performance (e.g., a MIDI recording) and visualize the calculated vector field in real time. We will discuss the structure of EspressoRubette and give the field interpolation and visualization principles. The article concludes with examples of visualization of performance fields.

Computer Music Journal, 27:1, pp. 47–58, Spring 2003

Expressive Shaping as an Infinitesimal View on Expression

The precise description of a musical performance poses major difficulties. On the one hand, expression is a multi-layered semiotic phenomenon. That is, expression extends from surface parameters to more hidden structures that reveal the score's analytical depth structure, for example. We do not deal with this complex problem here, because on the other hand, the surface expressiveness is indecomposable in general, i.e., it is typically not possible to separate expressive shaping of onsets (agogics) from duration (articulation), loudness (dynamics), or pitch (intonation). For example, the "Chopin rubato" makes it impossible to recover a single tempo curve, because the agogical structure depends on pitch when chords are slightly arpeggiated, and the right-hand melody onsets are locally deformed against the left hand accompaniment. One therefore needs a language that copes with this intrinsic intertwining of parameters.

In traditional musicology, performance theory has never developed an adequate conceptualization beyond fuzzy common language descriptions, although Adorno (1963) promoted an infinitesimal view of performance, termed "micrologic" and based upon the insight that performance deals with the infinite interpolation of shaping parameters. This deficiency is typically reflected in the feuilletonistic music criticism and has to date prevented a truly scientific musicological performance theory. More specifically, inverse performance theory is far beyond musicological concepts, because the reconstruction of system parameters of a given performance would require a precise definition of the performance data and the system setup. Because not even a performance theory based upon score analysis is available, such a system description remains out of reach of traditional musicology. However, in computer-aided performance research, system descriptions, which would enable inverse

performance theory, have been proposed (Sundberg 1991; Todd 1992).

We should remark on the concept of expression, because it is ambiguous in terms of content. If we attempt to analyze expression, this regards not the psychological perception of a performance by humans. This aspect is a legitimate one, but it touches a category that relates the performed music to human categorizations in terms of emotional response. Such a perspective is dealt with, for example, by Honing (1992) and Langner et al. (2000). In contrast, we regard expression as a rhetorically shaped transfer of structural score contents by means of the "deformation" mapping of symbolic data into a physical parameter space. The psychological implications are not the subject of this perspective; it is a purely mathematical description of this mapping, not of the emotional correlates.

Conventions and Definitions

Before starting with mathematical details, let us introduce the conventions used throughout this article. First of all, it is crucial to understand that the theory deals with symbolic musical events (e.g., data retrieved from a MIDI file) rather than acoustic signals. Thus, a score (or composition) C is a set of vectors (also referred to as points, or *events*) in an n-dimensional vector space. The performance $C_{Performed}$ is the corresponding set of the transformed score events.

The dimension of the vector spaces is arbitrary, because the presented theory is generic. Nevertheless, it is sufficient for the reader to assume that the vector spaces are simple and, for example, built of basic instrument parameters, such as onset time, pitch, loudness, and duration. In the symbolic score space S, we will typically use E for onset time, H for pitch, L for loudness, and D for duration. The corresponding lowercase symbols will be used for the performance space P. To denote a vector space of a particular type, we will write, for example, $S = \mathbf{R}[E,H]$ for a two-dimensional vector space over the set of real numbers $\mathbf{R}$, consisting of onset time and pitch.

When referring to a specific point in S, the symbol X will typically be used; the performance vector field will be named $Ts(X)$.

Score-Performance Matching

Much research effort has been invested in score-performance matching techniques. Score following, the real-time matching and tracking of soloists performing a given score, was first published by Dannenberg (1984) and Vercoe (1984). Puckette and Lippe (1992) presented methods used on the IRCAM Signal Processing Workstation (ISPW), whereas Heijink et al. (2000) gave an evaluation of different approaches to score-performance matching.

The literature has differentiated two types of algorithms. For real-time algorithms (primarily used in real-time accompaniment software), good performance had higher priority than matching quality. Offline algorithms, where calculation time is less important, were mostly used for in-depth analysis applications requiring a high level of matching quality. We, however, experienced that with modern processing power, high-quality matching can be performed in real time, particularly when the algorithms are well suited for extensive preprocessing of the given score.

Mathematically, the matching problem is complex and depends upon the desired maximum difference to be allowed between score and performance. For example, if chords remain chords and all notes are played exactly once, the problem is trivial. But normal performance includes more or less strong arpeggiation of chords, omissions of notes, or the playing of additional notes by error or by ambiguous definition of the notes, such as it is common for trills and other ornaments.

We have implemented an algorithm that matches along a "wave front" of notes that are defined by the temporal unfolding of performance. The algorithm thereby matches the constraints given by real-time requirements. Our implementation uses structural properties of a musical score and a corresponding performance: before starting the matching process, the score is structurally reorganized. For example, *pitch lists* are created. A pitch list is a list of all score events with the same pitch, ordered by onset time. Each event in the pitch list references the closest event in the pitch list above and below. The result is a grid-like data structure, which enables fast searching and

evaluation in the neighborhood of an event. Furthermore, dynamic programming techniques help coping with the real-time problem. Multiple possible solutions are created, maintained, and discarded as the matching process is running.

Let us now give a formal description of the basic principles behind the algorithm. Usually, matching is considered bottom-up in that the performance map of the whole piece is constructed from the performance map $X \to \wp(X)$ on the single element X. Instead, we tried to rebuild the element images from maps on sets of specific coverings I and J, respectively, of the composition C and its performance $C_{Performed}$. Typically, one considers the covering of C by hyperplane sections in each parameter (for example onset slices). On $C_{Performed}$, a covering J is defined which is a fuzzier version of I, for example neighborhoods of hyperplane sections (for example, epsilon neighborhoods in the onset dimension). If $\wp$ exists, different constraints can be imposed on the induced map on the coverings: First, $\wp$ induces a map $n_0(\wp)$: $I \to J$ such that $\wp(U) \subseteq n_0(\wp)(U)$ for all covering elements U in I. This yields a map $n(\wp)$: $n(I) \to n(J)$ of the "simplicial nerves" and thus conditions on the map on the covering sets. Second, the sets of these coverings are linearly ordered by $U < V$ if and only if either $U \subset V$ or both $U - V$ and $V - U$ are non-empty and $\min(U - V) < \min(V - U)$. In this order, we require that $U \leq V \Rightarrow n_0(\wp)(U) \leq n_0(\wp)(V)$. Third, if one defines a distance $d(U,V)$ between the covering sets (for example the elastic shape distance from motif theory [see Buteau 1998]), one requires that $d(U, n_0(\wp)(U)) < \varepsilon$ for a given positive distance limit ε. With these constraints, one may define the map $n_0(\wp): I \to J$ and then recover $\wp$ if every point X in C is seen as the intersection of all covering sets of I, which contain the point. This is evidently the case for the hyperplane sections described above.

Performance Field Calculation

The theory of performance fields is derived from the general hypothesis that performance is a smooth (continuously differentiable) isomorphism $\wp$: $S \to P$ on an open neighborhood of the given composition C. This is of course a strong hypothesis, but it is, at least locally on the given composition, a reasonable one. The mathematical model is standard insofar as it describes a transformation by means of given initial values and its derivative, the Jacobian matrix. (Recall from differential geometry that the Jacobian matrix is the matrix of all partial derivatives of the n coordinate functions in an n-dimensional space; it generalizes the usual derivative of a real function.) Its musical meaning is well known in the case of the tempo curve, which is the inverse derivative of the time transformation and yields the latter by integration from a fixed starting time.

In our inverse problem, we are not given $\wp$, but only its restriction $\wp|_C$ to the given composition. Accordingly, we shall not really construct the performance field Ts associated with the unknown map $\wp$, but a discrete performance field, defined on the points of C, which is determined by the restriction $\wp|_C$.

Now, let us consider the score space S, the performance space P, the performance transformation $\wp$: $S \to P$, and the constant vector field $\Delta(x) = \Delta = (1,1,\ldots 1)$ for all $x \in P$. The inverse image of the Δ field is given by

$$Ts(X) = J(\wp)^{-1}(X)\Delta$$

where $J(\wp)$ is the Jacobian matrix at X:

$$J(\wp)(X) = \left(\frac{\partial x_i}{\partial X_j}\right)\Big|_{X_j = E,H,L,D,\ldots}^{x_j = e,h,l,d,\ldots}(X) = \begin{bmatrix} \partial e/\partial E & \partial e/\partial H & \ldots \\ \partial h/\partial E & \partial h/\partial H & \ldots \\ \ldots & \ldots & \ldots \end{bmatrix}(X)$$

To calculate the field vectors in an element X of the given composition C in S, we have to determine $J(\wp)$. Now, assume that we are given a basis matrix U_X of not necessarily orthogonal basis vectors based in X. $J(\wp)$ can be rewritten as

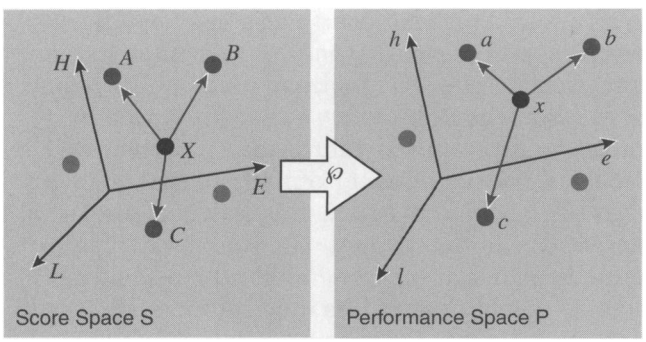
$$J(\wp) = J(\wp)U_X U_X^{-1} = V_X U_X^{-1}$$

where the basis matrix V_X is the image of the basis matrix U_X. Then, using the determinant and the adjoint matrix of V_X, the equation for $Ts(X)$ can be rewritten as

$$Ts(X) = U_X V_X^{-1}\Delta = \det(V_X)^{-1} U_X \operatorname{adj}(V_X)\Delta$$

The last term is used to identify three cases:

Figure 1. Basis transformation from score space S *to performance space* P.



1. V_X is regular, thus $Ts(X)$ is defined.
2. $\det(V_X) = 0$, $\text{adj}(V_X) \neq 0$: Only the direction of $Ts(X)$ is given, not its length.
3. $\det(V_X) = 0$, $\text{adj}(V_X) = 0$: No information at all is given $\langle / \rangle$.

While we are now able to calculate the field vectors, the question of how to find the appropriate bases is still open (because we just assumed a basis matrix U_X).

Obtaining the Bases

The only way to construct a basis in S is to use the relationships of the given points (i.e., the events of the score) in S. A basis can be constructed by taking n difference vectors with respect to X (n being the dimension of S). This situation is given in Figure 1: The vectors X, A, B, and C define a basis U_X in S, which is transformed to the basis matrix V_X in P, defined by x, a, b, and c. Arbitrary difference vectors could be considered as basis vectors, but owing to the following restrictions, the candidates must be selected carefully. First, only notes in a small Euclidean neighborhood of X should be considered. This principle of locality ensures that the basis consists of notes that are in the local musical context.

The second restriction is of a mathematical nature: we have seen that the transformed basis must be regular for us to be able to calculate the field vector. Because the performance is allowed to have arbitrary deviations from the score, there is no general solution to this problem. What can be done is to decrease the possibility that the transformation of the basis U_X leads to a non-regular basis matrix. This can be accomplished by making U_X as orthogonal as possible.

Thus, the selection of the basis vectors is based on the following two criteria: Locality (i.e., $|\det(U_X)|$ is minimal), and orthogonality (i.e., $|\det(U_X^{Norm})|$ is maximal, where U_X^{Norm} is the matrix of normalized basis vectors).

Note that the two criteria are to some extent mutually exclusive, so they must be weighted and combined. Consequently, a basis-calculation algorithm must select bases by searching for

$$\min\left(w_{Loc} |\det(U_X)| + w_{Orth} \frac{1}{|\det(U_X^{Norm})|} \right)$$

with w_{Loc} and w_{Orth} being positive pre- or user-defined weight values.

Unfortunately, there is still one case that must be dealt with: the case where it is not possible to find a regular basis matrix U_X in a small neighborhood of X. This may occur if all notes have the same loudness, if the basis has to be calculated for an isolated chord, where all onsets are equal, or for repeated notes with the same pitch. The only option left here is to construct orthogonal basis vectors that ensure that the basis remains regular.

The pseudo-code for a basis-calculation algorithm is now given.

```
// calculate a basis for each note X in
// score S
for(each Note X in Score S) {
 // obtain a list of the closest notes
 // with respect to x
   List neighbors = S.getNeighborList
   (X,maxDist);
 // start with an empty list of basis
 // vectors
   List basisVectors = emptyList;
 // add all difference vectors to the
 // list
   for(each Note N in neighbors)
 basisVectors.add(N-X);
 // choose possible basis vector
 // candidates from the list (see text)
   List bases =
 getCandidates(basisVectors);
```

```
// begin with the highest possible cost
  X.basisCost = infinity;
// evaluate all possible permutations
// from the given candidates
  for(each Basis B in bases) {
 // calculate cost based on given
 // weights
   float basisCost = wLoc * abs(B.det())
    + wOrth / abs(B.norm().det());
 // is this a better basis?
   if(basisCost < X.basisCost) {
  // yes: update cost and make it the
  // current basis
    X.basisCost = basisCost;
    X.B = B;
  }
 }
```

The function `getCandidates( )`, whose pseudocode was omitted here, generates a list of bases containing the permutations of the basis candidates, and it also adds constructed basis vectors if necessary. The above algorithm can be optimized by generating the permutations during execution, the best-expected ones first. In that case, the candidate list can be sorted by increasing distance, and the distance is used to stop the loop as soon as it is known that a lower `basisCost` cannot be reached anymore.

Although the method given in this section delivers mathematically correct results, they are in certain cases not satisfactory from a musical point of view. For example, when calculating a basis for an event that is part of a chord, at least one basis vector should be represented by the difference vector of an other event in the chord. Only in this way can we guarantee that the basis for the event is correct in terms of the local musical context. However, the above algorithm cannot guarantee this. If there exist several other events not part of the chord that deliver a lower cost, the chord structure will not be represented in the result. This currently imposes the biggest limitation in our method, and further research is needed to deliver musically correct results in every case. A first step will be structural analysis of the score with respect to basis calculation.

Real-Time Processing of Expressive Performance

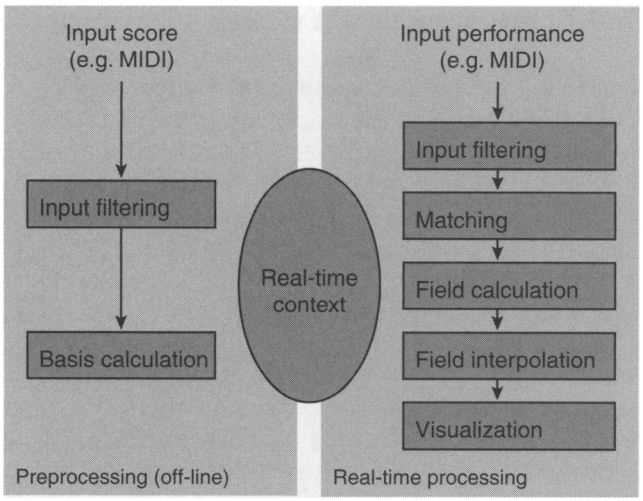
What kind of end-user applications can we expect when implementing the above methods? Because vector fields are well suited for visualization, most applications will make use of that property and will provide a means of visualizing expressive shaping. First of all, the field calculated from a score and a performance can be visualized. The field contains information on shaping of each of the coordinates (*E*, *H*, etc.) in the given parameter space, and thus we are able to display tempo information, articulation, dynamic shaping, and so on. Furthermore, missing or wrongly played notes can be highlighted—an important feature, for example for musicians rehearsing a score. Finally, with multiple performances of the same score available, difference fields can be visualized, thereby providing the possibility to show the differences among performances.

An implementation of the performance field theory should be able to operate in real time, especially for interactive applications, where immediate feedback—either visible or audible—is desired. As we shall see, the complete calculation of the performance fields can be divided into dedicated, communicating modules for specific tasks. Particularly important for performance is the extraction of tasks that can be processed in advance.

Figure 2 gives an overview of the modules and the flow of control (as shown by the vertical arrows) in our implementation. Modules are notified by events when new data for processing is ready. The modules themselves are stateless; they share their information with other modules in the real-time context, a data structure that contains all relevant information for the whole process, thus minimizing the risk of inconsistency. Of course, asynchronous accesses to the context have to be synchronized using locks or a similar synchronization technique.

For increased flexibility and efficiency, all modules accept lists of events, therefore making offline and real-time processing structurally identical. For example, the input filtering modules for the score and for the performance are the same. The former accepts the score as a whole, and the latter processes individual events as they are received in real

Figure 2. Flow of control in performance field calculation.



time. Furthermore, modules can be prioritized and be put to sleep if there is not enough processing power to support all present modules temporarily.

The following subsections give short descriptions for the modules shown in Figure 2. It is important to see that the described architecture allows the definition of additional modules as needed. This mostly depends on application requirements. Also, some applications might not need certain already defined modules, e.g., field interpolation in a computer accompaniment system.

Input Filtering

This module translates incoming note events to the representation defined in the real-time context. It also processes structural information, such as different voices, tempo changes, etc. The input filtering module must be implemented for any external representation (e.g., MIDI or RUBATO's Denotator format, as described in Mazzola [2002b]).

Basis Calculation

The calculation of the bases depends only on the input score, not on the performance and can thus be performed offline. For each event, an appropriate basis must be calculated. Typically, this is a time-consuming process.

Matching

The incoming performance events must be matched to the corresponding score events. As we have seen in the designated section, this is a non-trivial task and has developed into a research field of its own.

Field Calculation

The individual field vectors for each note must be calculated based on the pre-calculated basis and the given match.

Field Interpolation

The field vectors calculated by the former step are typically not aligned on a grid. However, for visualization, a 2D or 3D grid-like field with field vectors defined anywhere in this grid is desired. The interpolation step allows the definition of such a grid and performs the translation from the note field to the interpolated field.

Visualization

Finally, the calculated field is ready for visualization. Here, many user-defined parameters such as scaling, color-specification, ranges, etc., must be taken into account.

Visualization

One of the most straightforward applications of a calculated performance field is its visualization. Vector field visualization has been successfully used in many science and engineering domains, for example, in gas and fluid dynamics. Thus, many different techniques and their corresponding imple-

mentations are available. Common to all those methods is that they should be accurate, fast, and display the field in an intuitive way. (See Cabral and Leedom [1993] for an advanced method that is suited for 2D as well as for 3D visualization.) This section shows how the calculated field vectors must be processed to make them available to such standard visualization methods.

So far, we have dealt with a composition $C \subset S$, being a set of notes, the corresponding performance $C_{Performed} \subset P$, and the associated set F of calculated field vectors. The points in those sets reside in an n-dimensional space, n being the number of symbolic sound parameters. For visualization, n will normally be too large (i.e., larger than three), so as a first step, we must decide which parameters are used for visualization. For instance, we may choose onset E as the horizontal axis and pitch H as the vertical axis in a 2D setup. The remaining sound parameters are omitted. Furthermore, the desired field vector components must be selected, for example onset E in the horizontal direction and duration D in the vertical direction for a tempo-articulation field.

Field Interpolation

Typically, when one deals with vector fields, the field vectors are arranged in a grid of a given resolution. In contrast, our setup implies that the score points reside at arbitrary locations, making it impossible to use standard vector field visualization methods. Thus, a conversion from the calculated field vectors to vectors located on a grid is necessary. This can be accomplished through interpolation.

The most important aspect of interpolation is the fact that we are able to obtain continuous shaping information of the given performance: the interpolated field value at any point tells us how a hypothetical note would be shaped. As we shall see in the second example below, the visualization of that information will help in understanding expressive shaping intuitively.

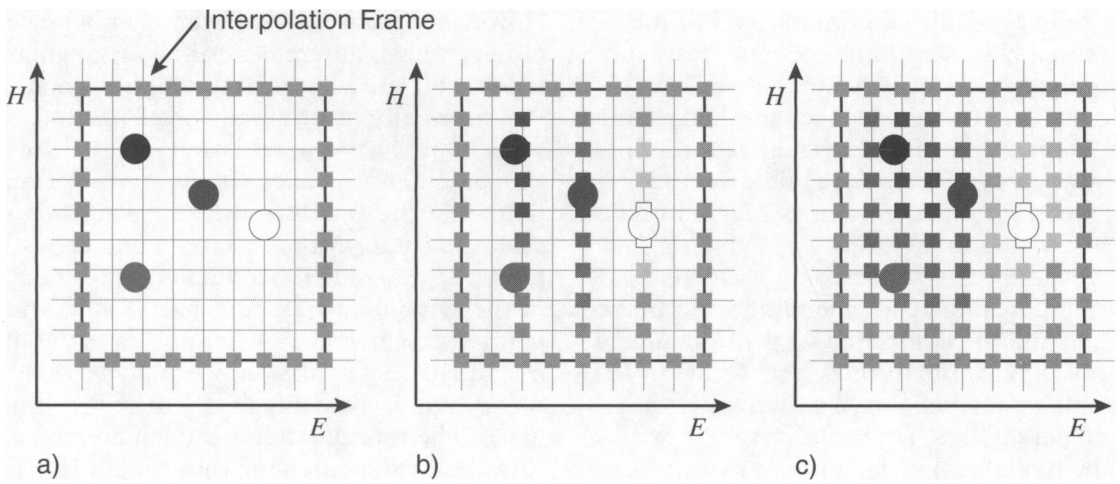
At first sight, a triangularization of the given set could be considered, making it easy to calculate the interpolated grid vectors in the resulting triangles. However, because the different symbolic sound parameters have different meaning, triangularization is not well suited here: interpolation should occur in a musically meaningful way. Therefore, it makes sense to perform interpolation in a defined recursive order. For instance, when interpolating an E-D field, first the D axis of the grid is considered and then the E axis. More precisely, one draws *hyperplanes* $H_1, H_2, \ldots, H_k$ perpendicular to the nth axis in the symbolic parameter space S such that every point of the given composition C sits in one such hyperplane. By recursion, we suppose that the interpolation is available for the first $n - 1$ coordinates. The interpolation value on an arbitrary point X is obtained by drawing the straight line through X and parallel to the nth axis. This line bisects two neighboring hyperplanes at points P and Q. The values in P and Q are then interpolated by a cubic spline with zero slope in P and Q.

Finally, what happens at the boundaries of the given set? Because no field vectors are available, boundary vectors must be defined. When looking at the theory of the former sections, it becomes clear that outside the boundaries, a frame of diagonal vectors must be placed corresponding to an unshaped performance. The dimensions of the surrounding frame must be predefined.

Observe that for an actual implementation, the surrounding frame must be quantized. When dealing with MIDI parameters, we can of course use the quantization of the individual parameters. When dealing with parameter spaces in $\mathbf{R}^n$, however, a resolution must be defined, and the space is effectively sampled.

To conclude this section, we shall give a simple example for the non-discrete 2D E-H case. Figure 3a shows four notes (disks). The value of the gray-scale is mapped to one component of the vector field, for example, the E component. For simplicity, the other field parameters are omitted. Also shown in Figure 3a is the interpolation frame around the four events and the raster that is applied. The interpolation order of the example is H first, then E. For musical content, this is an intuitive choice, because chords should be interpolated simultaneously. Figure 3b illustrates the first step of the interpolation, which takes place along the

Figure 3. Field quantization and interpolation.



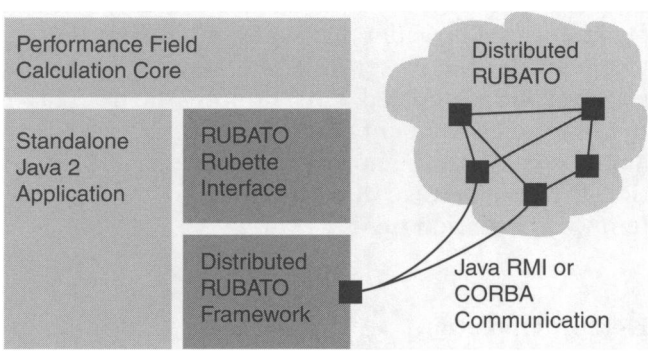
vertical axis. The horizontal position of the interpolation strips is defined by the *E* component of the notes. Finally, in Figure 3c, the remaining cells are interpolated along the horizontal axis.

EspressoRubette

The methods for algorithmic extraction of musical expressive shaping have been implemented in a tool called EspressoRubette. The tool can run as a stand-alone Java application. The Swing and Java2D classes take care of the user interface, and the user can manipulate calculation and visualization parameters through a simple dialog panel.

As an alternative and more flexible approach, this software also implements the Rubette interface and can thus be integrated into the Distributed RUBATO framework. (See Mazzola and Zahorka [1994b] for the original concept of RUBATO and Rubettes.) Since then, RUBATO has been redesigned to operate as a distributed, collaborative environment for music research (see Müller [2003]). Rubettes are basically autonomous components that communicate via Java RMI or with CORBA technology. Crucial to the framework is the PrimavistaRubette, a 3D browser that takes care of all human-machine interaction and visualization of complex, multi-dimensional data structures (see Mazzola [2002a]). Figure 4 gives an overview of the EspressoRubette and its environment. The performance field calculation core implements the concepts given in the earlier sections.

Figure 4. The EspressoRubette components.

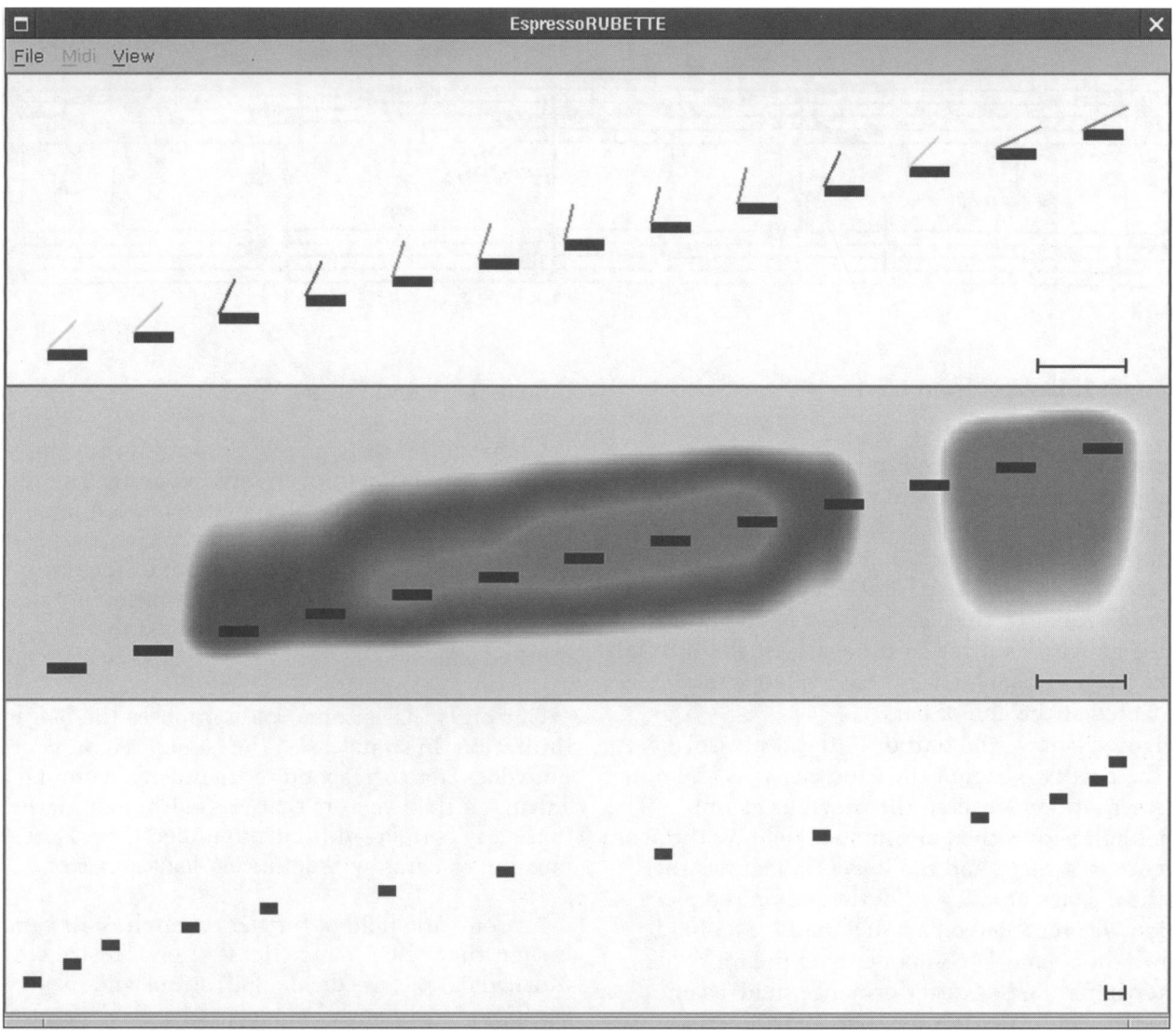


Example 1: Tempo Field of a Chromatic Scale

Let us now give examples of calculated performance fields. Figure 5 shows a chromatic scale and its performance. In this case, representation is close to that of a piano roll: the horizontal axis represents onset time (in units of MIDI ticks), and the vertical axis represents pitch. The width of events corresponds to their duration. Note that the EspressoRubette allows arbitrary redefinition of those mappings.

The top section shows the score containing a chromatic scale: thirteen note events in increasing pitch order, all with the same duration. A hypo-

Figure 5. Tempo field of a performed scale. The horizontal axis shows onset time, and the vertical axis represents pitch.

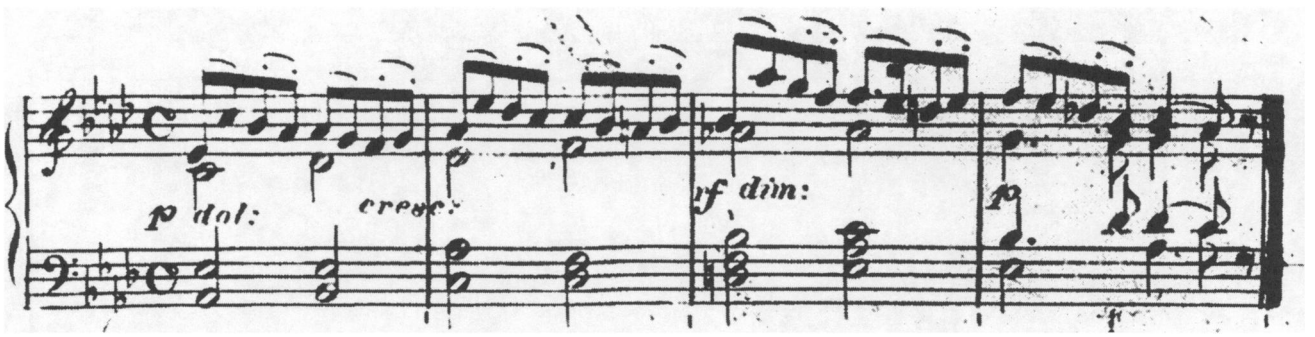


thetical performance with exaggerated rhythmic freedom of the thirteen events is shown in the bottom section. Observe that the horizontal axis in this section is visualized with a different scale, as indicated by the markers on the right. The first three events are played at the same speed as the original MIDI score. Then, the performance slows, and towards the end speeds up again. The last two notes are played faster than the MIDI score.

The vectors shown in the top section reflect this situation: the first, second, and eleventh vectors are diagonal vectors, stating that the notes are played at the given tempo. The angle of the other vectors depends on the local tempo played at a given note. Because we are dealing with a one-dimensional tempo field, only the angle of the field vectors is relevant in this case, not their length.

The middle section shows the corresponding interpolated field at a resolution of 400 × 200 cells. Here, the slope of each vector has been mapped to a shade of gray, and its length is related to the brightness of a cell.

Figure 6. Exercise from Carl Czerny's Vollständige theoretisch-praktische Pianoforte Schule, *Op. 500.*



Example 2: Excerpt from Czerny's Piano School

The second performance is a real-world example, namely a performance field for an exercise from Carl Czerny, *Vollständige theoretisch-praktische Pianoforte Schule*, Op. 500. (Refer to Figure 6 for the score of the exercise.) The exercise contains a "Chopin rubato," that is, the right hand plays the melody slightly shifted in time against the firm left hand chords in such a way that synchronization is recovered at the end of bars.

Figure 7 shows the output of the EspressoRubette for the first two bars of the exercise; axes and note representations are as in the previous example. The upper half shows the performance field on the score space $S = \mathbf{R}[E,H]$, and the lower half shows the physical space $P = \mathbf{R}[e,h]$ of the performed piece, which was recorded on a MIDI piano. The field shows the *E*- and *D*-components of the four-dimensional *E-H-L-D*–performance field, as encoded by grayscale. We see that there is a right hand rubato effect in the middle of each bar, significantly stronger in the second bar (right half of figure). The rubato is responsible for the rather complex vector field: the situation cannot be represented anymore by a single tempo curve. The cyclic shading effects are due to a multiple covering of the grayscale in order to make small slope differences of the performance field more visible. (The scaling of those deviations can be tuned by the user.) Full-size color images and MIDI files of above examples are available on-line at www.ifi.unizh.ch/staff/mueller/expression.

Conclusions and Future Work

We have presented a novel approach for the algorithmic extraction of expressive shaping. The results—calculated and interpolated performance fields—contain explicit expressive information and are available for visualization or for other performance analysis tools. The algorithms are not restricted to specific sound parameters, and the method can thus be used for extensive analysis of expressive shaping.

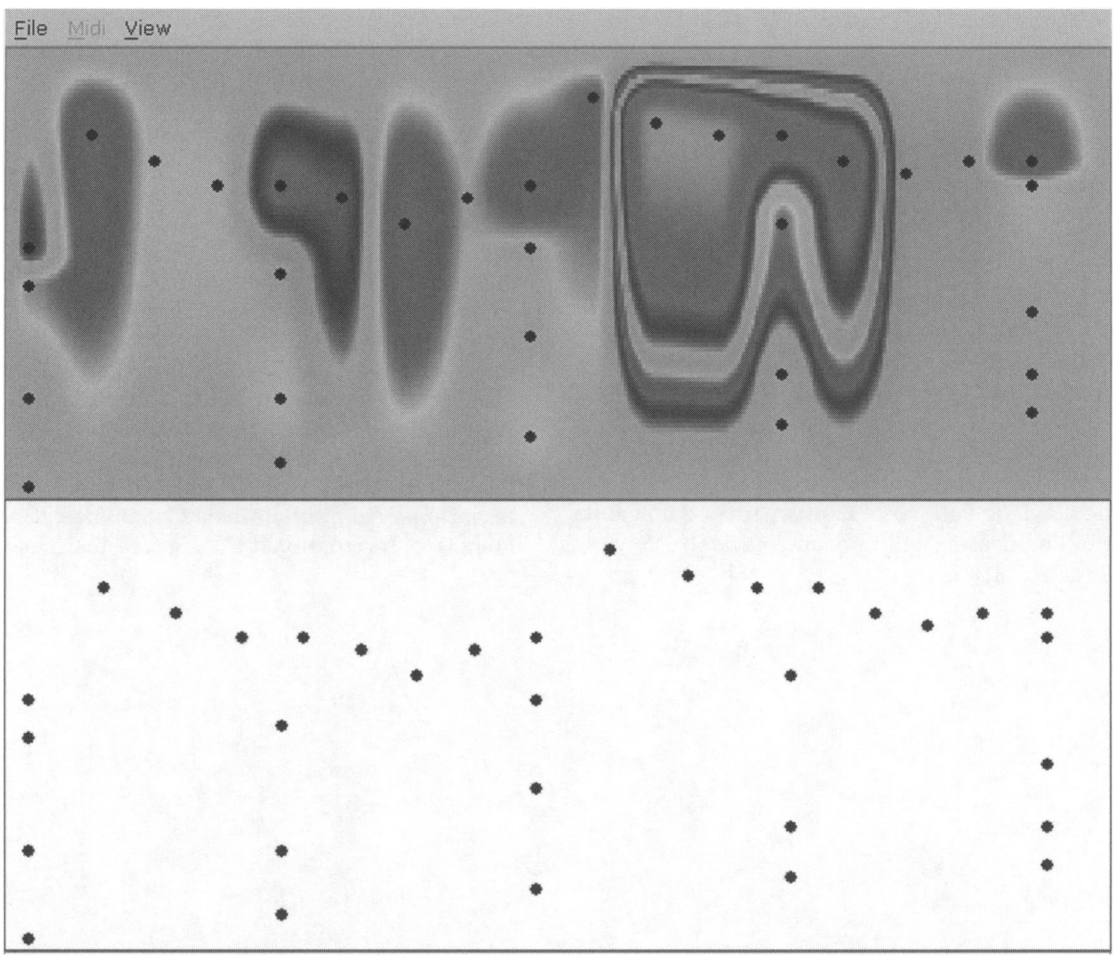
Currently, basis calculation imposes the biggest limitation. In some cases, the calculated basis of a note does not correspond to its musical context, resulting in field vectors that are—although mathematically correct—difficult to understand. Here, ongoing research will definitely lead to better results.

A promising field of further research is also the insight that performance fields are not restricted to musical data. In medical applications and in computer-aided anthropology, the growth information of human bones and organs can be extracted in a similar manner, in which case we may talk about "Nature's Performance." In this analogy, the role of the score is the genotype, while the role of the performance is the phenotype, which in biological terms is called the "expression" of the genotype.

Acknowledgments

We thank Peter Stucki for continuously supporting our research projects. This research was in part supported by SNSF grant 21–59 417.99.

Figure 7. Performance field of a "Chopin rubato" of the right hand in two bars in Czerny's exercise. Onset time is mapped to the horizontal axis, and pitch is mapped to the vertical axis.



References

Adorno, T. W. 1963. *Der getreue Korrepetitor.* Reprint, Frankfurt: Gesammelte Schriften, Band 15.

Buteau, C. 1998. *Motivic Topologies and their Signification in Musical Motivic Analysis.* Masters Thesis, Université Laval, Québec.

Cabral, B., and L. Leedom. 1993. "Imaging Vector Fields Using Line Integral Convolution." *Computer Graphics 27*:263–272.

Dannenberg, R. B. 1984. "An On-Line Algorithm for Real-Time Accompaniment." *Proceedings of the 1984 International Computer Music Conference.* San Francisco: International Computer Music Association, pp. 193–198.

Heijink, H., et al. 2000. "Make Me a Match: An Evaluation of Different Approaches to Score-Performance Matching." *Computer Music Journal* 24(1):43–56.

Honing, H. 1992. "Expresso, a Strong and Small Editor for Expression." *Proceedings of the 1992 International Computer Music Conference.* San Francisco: International Computer Music Association, pp. 215–218.

Langner, J., et al. 2000. "Realtime Analysis of Dynamic Shaping." In C. Woods, et al. *Proceedings of the Sixth International Conference on Music Perception and Cognition.* Keele, Staffordshire, UK: Department of Psychology.

Mazzola, G. 1995. "Inverse Performance Theory." *Proceedings of the 1995 International Computer Music Conference.* San Francisco: International Computer Music Association, pp. 533–540.

Mazzola, G. 2002a. *The Topos of Music.* Basel: Birkhäuser-Verlag.

Mazzola, G. 2002b. "Semiotic Aspects of Musicology: Semiotics of Music." In P. Posner, K. Robering, and T. A. Sebeok, eds. *Semiotics: A Handbook on the Sign-*

Theoretic Foundations of Nature and Culture. Berlin: Walter de Gruyter.
Mazzola, G. 2002c. "The Topos Geometry of Musical Logic." In G. Assayag, H. H. Feichtinger, and J. Rodrigues, eds. *Mathematics and Music.* Berlin et al.: Springer-Verlag.
Mazzola, G., and O. Zahorka. 1994a. "Tempo Curves Revisited: Hierarchies of Performance Fields." *Computer Music Journal* 18(1):40–52.
Mazzola, G., and O. Zahorka. 1994b. "The RUBATO Performance Workstation on NEXTSTEP." *Proceedings of the 1994 International Computer Music Conference.* San Francisco: International Computer Music Association, pp. 102–108.
Müller, S. 2003. "Computer-aided Musical Performance with the *Distributed Rubato* Environment. In G. Johannsen and G. de Poli, eds. "Human Supervision and Control in Engineering and Music." *Journal of New Music Research* 31(1).
Puckette, M., and C. Lippe. 1992. "Score Following in Practice." Proceedings of the 1992 International Computer Music Conference. San Francisco: International Computer Music Association, pp. 182–185.
Stange-Elbe, J. 1999. *Analyse- und Interpretationsaspekte zu J. S. Bachs 'Kunst der Fuge' mit Werkzeugen der objektorientierten Informationstechnologie.* Habilitation, Germany: University of Osnabrück.
Sundberg, J. 1991. "Music Performance Research. An Overview." In J. Sundberg, L. Nord, and R. Carlson, eds. *Music Language, Speech and Brain.* London: MacMillan Press.
Todd, N. P. M. 1992. "The Dynamics of Dynamics: A Model of Musical Expression." *Journal of the Acoustical Society of America* 91(6):3540–3550.
Vercoe, B. 1984. "The Synthetic Performer in the Context of Live Performance." *Proceedings of the 1984 International Computer Music Conference.* San Francisco: International Computer Music Association, pp. 275–278.