# The Composing Hand: Musical Creation with Leap Motion and the BigBang Rubette

Daniel Tormoen
School of Music
University of Minnesota
Minneapolis, MN 55455
tormo020@umn.edu

Florian Thalmann
School of Music
University of Minnesota
Minneapolis, MN 55455
thalm007@umn.edu

Guerino Mazzola
School of Music
University of Minnesota
Minneapolis, MN 55455
mazzola@umn.edu

## ABSTRACT

This paper introduces an extension of the Rubato Composer software's BigBang rubette module for gestural composition. The extension enables composers and improvisers to operate BigBang using the Leap Motion controller, which uses two cameras to detect hand motions in three-dimensional space. The low latency and high precision of the device make it a good fit for BigBang's functionality, which is based on immediate visual and auditive feedback. With the new extensions, users can define an infinite variety of musical objects, such as oscillators, pitches, chord progressions, or frequency modulators, in real-time and transform them in order to generate more complex musical structures on any level of abstraction.

## Keywords

hand gesture, meta-composition, improvisation, rubato composer, Leap Motion

## 1. INTRODUCTION

When used for music composition or performance, immediate gestural interfaces using spacial sensors, such as the Kaoss pad, the Kinect, the Wii remote, or Leap Motion are often used to control one or a few parameters or objects at a time, basically imitating slider motion multidimensionally, which is usually called one-to-one mapping [19]. The early musical example applications that use the Leap Motion device are instances of precisely this, for instance GECO [4]. While such use of gestural control can lead to convincing and natural results, there are clear limitations. Musicians think on a more abstract level and expect to control musical structures on a higher level. In recent years, several publications have discussed ways to gesturally control music on a more abstract level, e.g. [13, 3, 11]. All of them show how difficult it is to find an intuitive and direct way to map gestures to higher structures. With a one-to-many relationship, it can be difficult for users to hear and understand the results of their gestures. Compared to other gestural interfaces available, the Leap Motion controller has the potential to reinvent gestural music creation as its low latency, high precision, and numerous accessible parameters meet the high expectations of musicians [19, 20], and allow for a more direct type of interaction with musical structures, even on a higher level.

In this paper we present a solution to the above problem by extending the BigBang rubette module, a gestural composition and performance module for the Rubato Composer software originally controlled with a mouse or multitouch trackpad, in order to support the Leap Motion device. In our method we interpret the usable space above Leap Motion as a three-dimensional coordinate system where each fingertip is either interpreted as a musical object such as a note, oscillator, or modulator, or as an active agent within a transformation. The BigBang rubette's gestural concept, which ensures that every transformation is visualized and sonified gesturally while it is being performed, thereby evokes a sense of immediacy and embodiment that brings new possibilities to computer-assisted composition. After a brief introduction to Rubato Composer and the BigBang rubette module, we discuss how the rubette's basic functionality can be made accessible using Leap Motion. Later on, we introduce further more advanced applications arising with the rubette's possibility of tracing and gesturalizing the compositional process.

## 2. THE SOFTWARE

### 2.1 Rubato Composer, Forms, and Denotators

Rubato Composer is a Java software environment and framework [9] that is based on recent achievements in mathematical music theory. Specifically, it implements the versatile formalism of *forms and denotators* which roughly corresponds to the formalism of classes and objects in object-oriented programming but is realized in a purely mathematical way based on topos theory. *Forms* are generalized mathematical spaces commonly based on the category $Mod^{@}$ of presheaves over modules and defined by combining the logical-structural types **Limit**, **Colimit**, and **Power**, which correspond to limits, colimits, and powersets. These combined structures are ultimately based basic spaces analogous to primitive datatypes, referred to as **Simple**. *Denotators*, in turn, are points in the space of a form. They are the basic data type used for the representation of musical and non-musical objects in Rubato Composer. Rubette modules in the software typically operate on such denotators by applying transformations, so-called *morphisms* within a form or between forms, or evaluating them using *address changes*. For details, refer to [6, 9].

### 2.2 The BigBang Rubette

The BigBang rubette component [14, 15, 8] applies insights from transformational theory [5, 7], music informatics, and cognitive embodiment science by implementing a system of communication between the three musico-ontological levels of embodiment (facts, processes, and gestures) [16]. Traditionally, a composition is seen as a definite *fact*, a static result of the composition process. In BigBang it is rein-

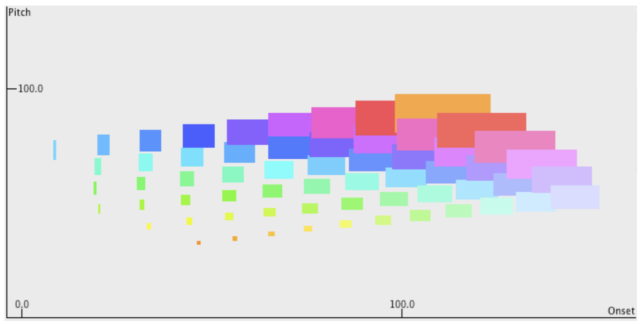


**Figure 1: A factual representation of a composition in BigBang. Each of the rectangles represents a specific object, having a number of freely assignable visual characteristics such as size, position, or color.**

terpreted as a dynamic *process* consisting of an initial stage followed by a series of operations and transformations. This process, in turn, is enabled to be created and visualized on a *gestural* level. The composition can thus typically be represented on any of the three levels. As a number of multidimensional points (denotators) in a coordinate system (according to the form) on the *factual level*, a directed graph of operations and transformations on the *processual level*, and a dynamically moving and evolving system on a *gestural level*. BigBang implements standardized translation procedures that mediate between these representations and arbitrarily translate gestural into processual compositions, processual into factual ones, and vice versa.

More precisely, BigBang enables composers to draw, manipulate, and transform arbitrary objects represented as denotators in an intuitive and gestural way and thereby automatically keeps track of the underlying creative process. It implements a powerful visualization strategy that consists in a generalization of the piano roll view, which can be recombined arbitrarily and which works for any arbitrary data type, as discussed in the next section (Figure 1). In the course of composition, any step of generation, operation, and transformation performed on a gestural input level is recorded on a processual level and visualized in form of a transformational diagram, a directed graph representing the entire compositional process (shown in Figure 2). Furthermore, composers cannot only interact with their music on an immediate gestural level, but also oversee their own compositional process on a more abstract level, and even interact with this process by manipulating the diagram in the spirit of Boulezian analyse créatrice [1]. If they decide to revise earlier compositional decisions, those can directly be altered, removed from the process, or even inserted at another logical location.

## 2.3 Some Examples of Forms

Traditionally, the BigBang rubette was meant to be used to create and manipulate *Score*-based denotators, which roughly correspond to midi-data, extended to include hierarchical objects and timbre. The basic *Score* form is defined as

$$\begin{aligned}&Score : .\mathbf{Power}(Note),\\&Note : .\mathbf{Limit}(Onset, Pitch, Loudness, Duration, Voice)\end{aligned}$$

which means that a score consists of a set of notes, each of them being a point in a five-dimensional space. Each of the dimensions of this space is a **Simple** form, for instance $Pitch : .\mathbf{Simple}(\mathbb{Q})$.

Recently, however, BigBang was generalized to accept

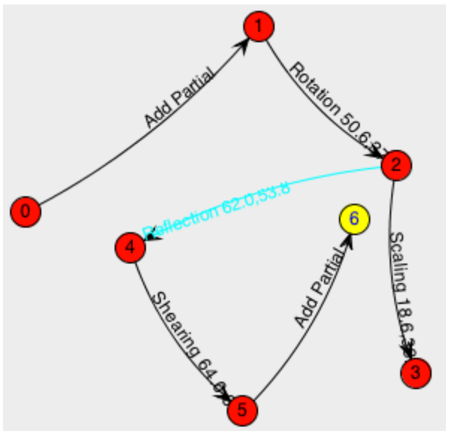


**Figure 2: A graph of a composition process of a** *SoundSpectrum* **including all five geometric transformations (*Translation, Rotation, Scaling, Shearing, Reflection*) as well as the drawing operation (*Add Partial*).**

and handle any data type modeled as a form. In recent papers we introduced some new examples of how these data types can look like and discussed their capabilities within BigBang [18, 17]. Significant among these in the context of this paper were forms that did not have time-based coordinates, for instance

$$\begin{aligned}&SoundSpectrum : .\mathbf{Power}(Partial),\\&Partial : .\mathbf{Limit}(Loudness, Pitch).\end{aligned}$$

or

$$\begin{aligned}&FMSet : .\mathbf{Power}(FMNode),\\&FMNode : .\mathbf{Limit}(Partial, FMSet),\end{aligned}$$

which allow for the generation of continuously sounding sound spectra and frequency-modulation-based synthesis. The gestural capabilities of Leap Motion is especially valuable in combination with such data types.

## 3. LEAP MOTION AND BASIC BIGBANG

Originally, the BigBang rubette was conceived to be operated with a mouse, where drawing, geometric transformations, and more complex transformations could be executed by clicking and dragging. Later on, it was made compatible with multitouch interfaces [15], which brought significant improvement in gestural intuitiveness and immediacy. The current paper presents a further step towards a more embodied and versatile way of interacting with BigBang.

### 3.1 Why Leap Motion?

There are a few reasons the Leap Motion controller in particular is well-suited for music performance. First, it provides a visual performance. It allows for the manipulation of software instruments and sounds in a way where the audience can feel engaged in the performance. The intuitive response to movement means that the Leap Motion is easy to use for the performer, and the audience can hear the relationship between the movements the performer makes and the sounds produced.

The Leap Motion is also highly accurate and precise for its minimal latency. It has a very high framerate which makes the response to fast movements very smooth. The processing time required for each frame is between 4 to 10 ms, within the upper bound set by [20]. Other comparable sensors such as the Kinect have a much higher latency, around

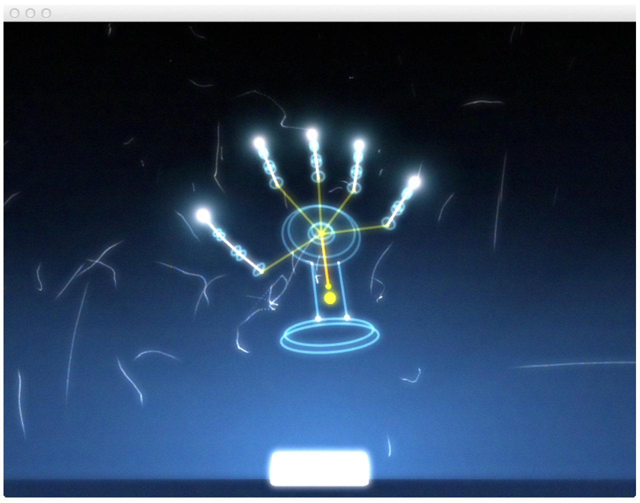


**Figure 3: The default visualization of the data from the Leap Motion from the Leap Motion software. The orientation of the palm, the location of each finger tip, and the direction in which each finger is pointing is all that is used to construct the visualization.**

100 ms. 4 ms is comparable to sitting approximately 1.5 meters from a speaker while 100 ms is the delay experienced 34 meters from a speaker. These are strictly the processing times associated with the sensor's algorithms and do not include the time for software sound production which can further exacerbate a high processing latency. The latency associated with the Kinect's algorithms as well as it's lower framerate and accuracy make it an inferior choice in music performances, e.g. [13].

The Leap Motion achieves this favorable latency at the cost of a much more limited view of its environment (Figure 3). Unlike the Kinect, the Leap Motion does not produce a point cloud of the whole scene within its view. It instead describes specific features. It finds each individual finger of each hand, the direction in which the finger is pointing, the finger tip, and the location and orientation of the palm of each hand. The exact algorithm used is not publicly available, but Leap Motion has publicly verified their method. These highly responsive and precise parameters are especially suitable for live control of continuous computer music parameters. Compared to other earlier devices, for instance described in [10], the Leap Motion yields the gestural motion of 12 three-dimensional vectors as well as 12 three-dimensional points, if two full hands are used. Even though the interdependency of these vectors – the fingers of a hand have physically limited degrees of freedom – seems a disadvantage at first, there are musical constructs that mirror the structure and it can be directly mapped to it, as briefly mentioned in Section 4.2. In this paper, we only use the locational parameters of fingers and hands, but we will soon extend our work to using the direction vectors.

## 3.2 Drawing with Leap Motion

The most basic operation in a compositional process is to add musical objects, or in the case of the BigBang rubette, denotators. When using the Leap Motion we treat each finger tip as a denotator and map the $(x, y, z)$ location of each finger using a linear scaling into the coordinate system represented currently displayed by the BigBang rubette. Whenever the fingers move around the corresponding denotators are adjusted, which provides an immediate visual and auditive feedback. From there, we have the option to

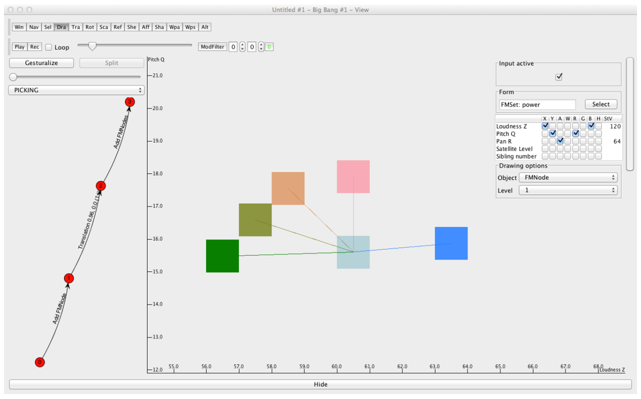


**Figure 4: An *FMSet* denotator consisting of a carrier and five modulators defined by the fingertips of the user.**

capture the currently defined denotators and keep adding new ones using the same method. If we use all three dimensions of the Leap Motion space, capturing is only possible with an external trigger (such as a MIDI trigger). To avoid the use of an external trigger the user can decide to use only two dimensions for drawing (preferably $x \times y$) and the third dimension for capturing, whenever a certain threshold, a plane perpendicular to the z-axis at $z = 0$, is crossed.

Figure 4 shows a situation where the modulators of a carrier in frequency modulation synthesis are defined using Leap Motion. Their arrangement directly corresponds to the fingertips in space, as can be verified visually. Compared to drawing with a mouse or another device, this method has significant advantages. The user can quickly compose complex musical structures while being able to smoothly preview each step until satisfied. Furthermore, the user can also easily edit musical objects added earlier in the process in the same continuous way which has many musical applications, some of them described in Section 4. The high precision of the Leap Motion makes this method is just as accurate as using a mouse or trackpad.

## 3.3 Basic Transformations Using Leap Motion

Since denotators are mathematical objects we can manipulate them using arbitrary morphisms. BigBang provides a set of simple geometric transformations: translation, rotation, scaling, shearing, and reflection. As discussed in [14], any affine transformation can be represented as a series of two-dimensional geometric transformations. In this section we describe how we define a gesture for each of these basic transformations. Each of these gestures imitates input that is possible with a mouse, but later we will discuss methods that go beyond simple 2D input to full 3D input.

To be able to precisely define the transformations we need a method to determine the beginning and end of a gesture. Analogous to the second drawing method described in the previous section, we found that the best solution is to have a vertical plane above the Leap Motion at $z = 0$ that needs to be crossed in order to actively transform the score. On the side of the plane closer to the composer, the composer can use swipe left or swipe right gestures to navigate between different types of transformations without affecting the score.

### 3.3.1 Translation

Translation is simply shifting points along the x and/or y axis. Once the composer's hand crosses the plane above the Leap Motion the frontmost point defines the translation transformation. The x and y values of the denotators move linearly with the x and y location of the finger in the Leap Motion space. The usable space above the Leap Motion is mapped to the size of the space shown in the BigBang rubette. This allows for arbitrary precision because we can zoom in and move around in the BigBang rubette to manipulate a very small area or zoom out for larger translations.

### 3.3.2 Rotation, Scaling, and Shearing

Rotation, Scaling and Shearing all require two fingers to perform the transformation. We found the best method is to use the two front-most fingers. The locations of the two finger points and the center of the two fingers are necessary for each of these transformations. For each of these transformations the center point can change in the Leap Motion coordinates, but it stays fixed within the BigBang rubette score. This makes the gestures easier to perform and more precise. Each of these operations requires that center of the operation is at the origin so we have to translate the center to the origin and then translate back as seen in equation 1 where $T$ is the translation from the origin to the center of the transformation, $d$ is a denotator, and $S$ is the rotation, scale, or skew matrix.

$$T^{-1}STd = d' \tag{1}$$

For the rotation gesture we use the change in the angle between the two fingers to determine how much to rotate around the center point in the BigBang rubette. For this transformation we find one of the limitations of a vision-based system for gestures. When one's fingers are rotating, at some point one finger can occlude the other making the view from the Leap Motion appear to have only one finger. To overcome this problem we had to extrapolate the expected data to ensure rotations are still possible even with occlusions. When one finger is lost we fix the center of the rotation in the Leap Motion space rather than allowing it to move. We then guess that the second finger is on the opposite side of the center from the finger we can see. We proceed updating the angle with guesses for the second finger until it is found once again. Using this method we can perform a smooth 360° rotation even though the second finger is not in the view of the Leap Motion for part of the rotation.

We can calculate the scale transformation $S$ by comparing the original offsets of the fingers to the current offset. In equation 2, $x_i$ and $y_i$ are the initial locations of the fingers and $x'_i$ and $y'_i$ are the current locations.

$$S = \begin{pmatrix} \frac{x'_2 - x'_1}{x_2 - x_1} & 0 & 0 \\ 0 & \frac{y'_2 - y'_1}{y_2 - y_1} & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2}$$

Shearing is calculated by comparing comparing the original location of a finger relative to the current location. In equation 3 $x$ and $y$ are the original location of a finger relative to the center and $x'$ and $y'$ are the current location.

$$H = \begin{pmatrix} 1 & x' - x & 0 \\ y' - y & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3}$$

## 3.4 Arbitrary Affine Transformations

Arbitrary affine transformations are very difficult to input using a mouse. If we do not include translation we need two pairs of points to define the transformation in two dimensions, two points before the transformation and the corresponding two points after the transformation. Solving a system of equations that includes translations requires the use of a homogeneous coordinate system and three pairs of points. In equation 4 we show the transformation $T$ we need to solve for. $(x_i, y_i)$ represents each of the three starting points and $(x'_i, y'_i)$ are the corresponding ending points. We need to solve for the transformation matrix T.

$$\begin{pmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} x'_1 & x'_2 & x'_3 \\ y'_1 & y'_2 & y'_3 \\ 1 & 1 & 1 \end{pmatrix} \tag{4}$$

Assuming we have non-singular matrices, we can solve for $T$ by inverting $X$ as in equation 5.

$$T = X'X^{-1} \tag{5}$$

There are some potential problems with collinear points using equation 5 which we will discuss in section 3.4.2. There are also a few problems with applying this method of finding arbitrary affine transformations directly to the Leap Motion. First, we could potentially use any number of fingers. For the device to be easy to use as a composition and performance tool we need to handle any number of fingers intuitively and make transitions seamless. In order to do this we found that a method of chaining transformations together is most effective as shown in equation 6. At each step we receive a list of all fingers that the Leap Motion was able to find in the scene. At each step we compare this list to the previous frame to determine which fingers were present in the previous frame, or in other words are a "match". A transformation matrix that describes the change between the two frames is found and multiplied to the cumulative transformation matrix for the whole gesture. $T_c$ is the cumulative transformation for the first $n$ steps.

$$T_n T_{t-1} \ldots T_2 T_1 = T_c \tag{6}$$

Depending on how many matches are found the way the next step in the transformation is found changes. When only one finger is found the functionality is the same as the translation gesture in section 3.3.1. When two matches are found the functionality is similar to the rotation transformation described in section 3.3.2 except the center of the transformation in the BigBang rubette is not fixed. The denotators can be translated linearly with the movement of the center of the two found fingers. When there are three matches we can solve for an arbitrary affine transformation directly as shown in equation 4.

### 3.4.1 Overdetermined Transformations

When there are more than three matches the system of equations is overdetermined and may not have an exact solution. Instead we have to come up with a good guess. The simplest way to get a good estimate of a transformation is to use a brute-force method to minimize the error. Assuming we have a guess of the transformation $T$, we can evaluate the error $e$ as shown in equation 7.

$$\sum \|Tp_i - p'_i\| \quad = \quad e \tag{7}$$

In most cases this method is fast enough to be used in real time, but the time complexity as the number of matched fingers is $\Theta(n^3)$ as this is a simple $n$ choose $k$ problem with $k$ fixed at 3. There are methods to ensure that even when many matches are found that performance is still very good. RANSAC (Random Sample Consensus) [2] randomly picks a set of three matched points that are assumed to be inliers. The affine transformation is then calculated using these matches. All other points are tested using the error method shown in equation 7 to see if the transformation is a good estimate of the rest of the data. Then the process is repeated with a new random sample. The number of iterations is set based on the performance requirements of the system or set by a certain error threshold. After all iterations have completed the best estimated transformation is used.

Another possible extension to the RANSAC method is to use least-median of squares linear regression [12] to find a local minima for affine transformation using the error function in equation 7. The globally optimal solution may not actually match up exactly with any set of 3 matches. By choosing the best set of 3 matches and then using the least-median of squares to find a local minimum the solution should be closer to the global minimum. Although this method is not guaranteed to converge on a globally optimal solution, it will produce a smoother chain of transformations when there are many matches.

#### 3.4.2 Handling Collinear points

Solving equation 5 requires that the set of points we are using are non-collinear. As the points approach being collinear, solving for the affine transformation can become numerically unstable, or make being precise difficult for the composer. There are a few solutions to this problem. First we can modify our approach of chaining transformations together so that instead of updating the starting points of the match at each step, we keep starting points for as long as all of the same fingers are matched. Each transformation can describe multiple frames from the Leap Motion. This is can solve the problem in some cases, but in many cases occlusions can cause fingers to be lost, forcing a new transformation to occur in the chain. Another solution is when 3 points are close to collinear we remove one match and only use the remaining matches. When there are more than 3 matches, we simply skip the iteration of RANSAC that is a set of near-collinear points. This method has the potential drawback that it may not always produce a smooth motion when the composer's fingers are near collinear.

### 3.5 Extending to 3D Coordinate Systems

So far the methods we have described for creating different transformations are all based on a 2D coordinate system for denotators. Each of these methods can be extended to 3D with a few small changes. First, crossing a plane above the Leap Motion to start manipulating the score is no longer an effective on/off trigger. This trigger limits how we can manipulate the score along the z-axis. The best alternative is to use a trigger external to the Leap Motion such as a MIDI foot pedal so we can maintain precise gestures and make the best use of the active space the Leap Motion covers. This allows us to start and end a gesture in any area that the Leap Motion covers, is easy to use, and is simple to implement.

Solving for 3D affine transformations also requires some changes. First we now need 4 matches to solve for an arbitrary 3D affine transformation rather than 3. Each transformation matrix is a $4 \times 4$ matrix so we can still use a homogeneous coordinate system. Each of the cases must change to accommodate. For 1 match the translation is now in 3D rather than 2D. For 2 matches the translation and rotation is now in 3D. For 3 matches we do not have enough parameters to solve for an arbitrary 3D affine transformation. Instead we generate a fourth point that will allow us to solve for a 3D affine transformation directly. This point is calculated by finding the vector that is perpendicular to the plane defined by the 3 matched points. Equations 8 and 9 show how to generate a fourth point. By using this equation on both the starting and corresponding end points a transformation can be found with only 3 non-collinear matches.

$$v = (p_2 - p_1) \times (p_3 - p_1) \quad (8)$$

$$p_4 = p_1 + \frac{v}{\|v\|} \quad (9)$$

In 9 the normalization step can be left out to allow the points to be scaled along the basis vector $v$ proportionally to the average of the scaling along the basis $p_2 - p_1$ and $p_3 - p_1$.

When there are 4 matches the affine transformation can be solved for directly. Finally when there are more than 4 matches a brute force or RANSAC method can be used.

## 4. ADVANCED USES OF LEAP MOTION AND BIGBANG

Using the Leap Motion to define, move around, and transform arbitrary denotator objects in BigBang adds highly intuitive and direct possibilities for musical improvisation and composition. However, these are just the basic functions of the BigBang rubette. When combined with the more complex operations available in BigBang, there are unique possibilities.

### 4.1 Dynamic Motives

#### 4.1.1 Transformations

The transformation graph generated by the BigBang rubette (described in Section 2.2) not only keeps track of all operations and transformations in order of execution, but also allows users to edit and adjust earlier transformations while observing later states of the composition. Previously, the add operation which adds denotators to the composition could merely be edited by drawing additional objects with the mouse. The gestural immediacy of Leap Motion now allows users to edit an earlier set of denotators by redefining them gesturally, basically holding a denotator with each of their fingers, while all subsequent transformations and operations are applied to these redefined objects. This allows us to for instance gesturally define a preliminary set of denotators, then produce several copies by translating, rotating and scaling them, and finally go back and edit the objects added in the first step. This regenerates all copies based on the currently defined input set.

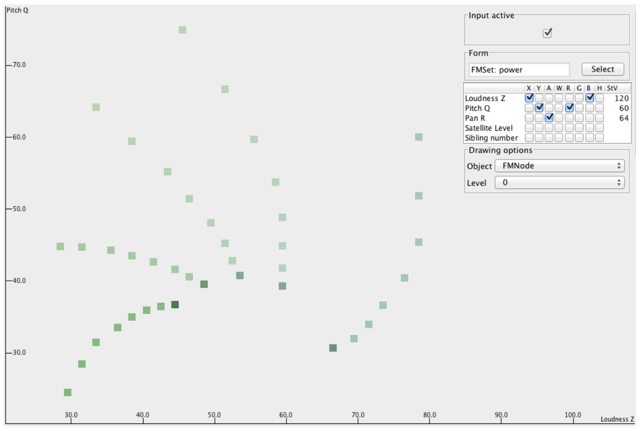
#### 4.1.2 Wallpapers

Even more powerful is the use of the wallpaper function (see [14, 9]), which systematically creates patterns of copies of an original motif, by applying a number of transformations repeatedly to it. When the step where the wallpaper motif is drawn is reedited with Leap Motion, the motif can virtually be grabbed by the user and moved around upon which the entire wallpaper moves accordingly. Figure 5 shows an example of such a wallpaper, where the motif has a recognizable hand shape defined by the user.

#### 4.1.3 Gesturalized Processes

A third possibility is to use the BigBang rubette's gesturalizing function which animates the entire evolution of the compositional process gesturally (see Section 2.2). Even during the process of gesturalization, the input set of denotators can be varied, which enables the user for instance to define a process of a gradually evolving complex sound structure, the germinal motif of which is itself changing.

Note that any of these examples in this section can be created using any user-defined form, which means that the de-



Input active ✓

Form: FMSet: power

| | X | Y | A | W | R | G | B | H | StV |
|---|---|---|---|---|---|---|---|---|---|
| Loudness Z | ✓ | | | | | | ✓ | | 120 |
| Pitch Q | | ✓ | | | ✓ | | | | 60 |
| Pan R | | | ✓ | | | | | | 64 |
| Satellite Level | | | | | | | | | |
| Sibling number | | | | | | | | | |

Drawing options

Object: FMNode

Level: 0

**Figure 5: A wallpaper with a motif defined by the fingers of a hand.**

notators defined gesturally with Leap Motion can represent any objects, for instance oscillators, modulators, pitches in a chord, or sound events in a loop.

## 4.2 Dynamic Sound Synthesis

Instead of defining motifs in a composition or improvisation as suggested in the previous section, the denotators defined in real time with Leap Motion can also be interpreted as designed sounds, directly mapped to the keys of a keyboard controller. While the user is playing the keyboard, the positions of the fingers over the Leap Motion can be directly mapped to carrier oscillators or frequency modulators (as shown in Figure 4) and each hand movement changes their parameters. Such a hierarchical modulator structure is especially interesting for Leap Motion control, since the palms of the hands can define the carriers while the fingers define the hierarchically dependent modulators. In a similar way, the user can create sounds and transform them gesturally in any of the geometrical transformation modes. This way, instead of changing simple parameters in a linear way as with commonly available synthesizer interfaces, multiple parameters can be changed in a complex way, such as for instance manipulating both frequency and amplitude of hundreds of oscillators around a defined sound center.

## 4.3 Playing Instruments with Leap Motion

A final example for an application of Leap Motion in conjunction with BigBang uses the new real-time midi out function of BigBang. Objects generated with BigBang may also be played back on instruments with a midi interface, such as a midi grand piano. The most straightforward application of this is to move the hands in a space roughly corresponding to the real instrument space, such as pitch as an x-axis and loudness a y-axis with a piano. However, the improviser may also choose to act on an abstract level not directly related to the instrument. One finger may stand for an entire chord or pattern to be played and moving it around may change the pattern. This way, the improviser gains higher-level control over the instrument, while keeping the gestural intuitiveness commonly associated with instrument-playing.

## 5. REFERENCES

[1] P. Boulez. *Jalons*. Bourgeois, Paris, 1989.

[2] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[3] J. Françoise, N. Schnell, and F. Bevilacqua. A multimodal probabilistic model for gesture–based control of sound synthesis. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 705–708. ACM, 2013.

[4] Geco. multi-dimensional midi expression through hand gestures. http://www.uwyn.com/geco.

[5] D. Lewin. *Generalized Musical Intervals and Transformations*. Oxford University Press, New York, NY, 1987/2007.

[6] G. Mazzola. *The Topos of Music. Geometric Logic of Concept, Theory, and Performance*. Birkhäuser, Basel, 2002.

[7] G. Mazzola and M. Andreatta. From a categorical point of view: K-nets as limit denotators. *Perspectives of New Music*, 44(2), 2006.

[8] G. Mazzola and F. Thalmann. Musical composition and gestural diagrams. In C. Agon et al., editors, *Mathematics and Computation in Music - MCM 2011*, Heidelberg, 2011. Springer.

[9] G. Milmeister. *The Rubato Composer Music Software: Component-Based Implementation of a Functorial Concept Architecture*. Springer, Berlin/Heidelberg, 2009.

[10] E. R. Miranda and M. M. Wanderley. *New digital musical instruments: control and interaction beyond the keyboard*, volume 21 of *Computer music and digital audio series*. A-R Editions, Middleton, 2006.

[11] T. J. Mitchell. Soundgrasp: A gestural interface for the performance of live music. In *Proc. NIME*, volume 2011, 2011.

[12] P. J. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79(388):pp. 871–880, 1984.

[13] S. Sentürk, S. W. Lee, A. Sastry, A. Daruwalla, and G. Weinberg. Crossole: A gestural interface for composition, improvisation and performance using kinect. In *Proc. NIME*, volume 2012, 2012.

[14] F. Thalmann and G. Mazzola. The bigbang rubette: Gestural music composition with rubato composer. In *Proceedings of the International Computer Music Conference*, Belfast, 2008. International Computer Music Association.

[15] F. Thalmann and G. Mazzola. Affine musical transformations using multi-touch gestures. *Ninad*, 24:58–69, 2010.

[16] F. Thalmann and G. Mazzola. Poietical music scores: Facts, processes, and gestures. In *Proceedings of the Second International Symposium on Music and Sonic Art*, Baden-Baden, 2011. MuSA.

[17] F. Thalmann and G. Mazzola. Using the creative process for sound design based on generic sound forms. In *MUME 2013 proceedings*, Boston, 2013. AAAI Press.

[18] F. Thalmann and G. Mazzola. Visualization and transformation in a general musical and music-theoretical spaces. In *Proceedings of the Music Encoding Conference 2013*, Mainz, 2013. MEI.

[19] M. M. Wanderley. Gestural control of music. In *International Workshop Human Supervision and Control in Engineering and Music*, pages 632–644, 2001.

[20] D. Wessel and M. Wright. Problems and prospects for intimate musical control of computers. *Computer Music Journal*, 26(3):11–22, 2002.