EvoMove: Evolutionary-based living musical companion

Sergio Peignier, Jonas Abernot, Christophe Rigotti, Guillaume Beslon

EvoMove: Evolutionary-based living musical companion

Sergio Peignier[1], Jonas Abernot[1], Christophe Rigotti[1] and Guillaume Beslon[1]
[1] Université de Lyon INSA-Lyon, CNRS, INRIA LIRIS, UMR5205 F-69621, France
name.lastname@inria.fr

Abstract

The EvoMove system is a motion-based musical companion that relies on a commensal computing scheme. The system relies on wireless sensors to detect dancer moves. The sensor information is sent to KymeroClust, an evolutionary algorithm that identifies and maintains a clustering model of the move categories. The system uses this information to play audio samples according to the detected categories. These categories are not predefined, but are built dynamically by clustering the stream of data coming from the motion sensors. The EvoMove system has been tested by different users and subjective promising experiences are reported.

Introduction

One of the objectives of ALife is to create "Living Technologies", i.e. technologies that are able to adapt to their environment in an open-ended way (**?**). However, the concept of "Living Technology" sounds like an oxymoron lying on two antagonistic properties. On the one hand, one of the core universal properties of living beings is their autonomy. Even if some forms of cooperation or altruism can be observed in nature, every biological system is fundamentally selfish. On the other hand, one of the core universal properties of technology is its controllability. These two antagonistic properties immediately conflict when one wants to design living technologies: The aim here is to design a system that satisfies the goals it was built for (technology) and that is autonomous enough to surprise its user (living). How can a technology be simultaneously alive and controllable? We propose here a bio-inspired approach, called "commensal computation" (**?**), to resolve the autonomy vs. controllability conundrum. In biology a commensal interaction is a form of mutualism between two organisms where the association is not detrimental but not obviously beneficial to the partners (**?**). The idea of commensal computation is based on the concept of nutrient processing by microbiota: Gut microbes degrade ingested substances that would otherwise be non-digestible or even harmful to the gut (**?**). Hence, gut microbes pre-process the complex flow of nutrients and transfer the results to their host organism, helping it to regulate its feeding, to extract specific nutrients and to gain resources it can use to survive. Importantly, while doing so, the microbiota changes and evolves according to its environment, i.e., according to what the host eats. The commensal association of the microbiota and the host contains a part of autonomy (the microbes) and a part of control (the host). We propose to organize a living computational system following the manner in which host and microbiota are engaged in a mutualistic association. In commensal computation, some complex data are pre-processed by a virtual microbiome (the "commensals") that transforms them into simpler data that the processing system (the "host") can use. Such an architecture differs from classical preprocessing-processing in that here the pre-processing is performed by a community of virtual bacteria that evolves autonomously (albeit in environmental conditions that depend on the state of the global system). In particular, the only objective of the microbiota is to uptake data, transform them into some "objects" and feed these objects to the main processing system. Importantly, it does so independently from any global objective.

In this paper we present the EvoMove system, a musical companion based on the commensal computation architecture. EvoMove relies on a system of sensors that captures the moves of a dancer thus feeding a population of virtual organisms with complex information. As these virtual organisms evolve, they produce some clustering models of the data, categorizing raw moves into groups of similar ones, later-on called "movements" (a movement being an organized set of moves). Finally, at each time step, the activated movements are sent to the processing layer that uses this information to trigger small sound samples that in turn "feed" the dancer. The EvoMove musical companion has been developed as a demonstrator for the 2013-2016 FP7 EvoEvo project (`www.evoevo.fr`). This project is an interdisciplinary project aiming at developing living technologies by taking inspiration from the evolution of microorganisms (virus and bacteria). EvoMove has been developed to assess the efficiency of the living technologies developed during the EvoEvo project with a challenging real world application.

In the rest of the paper we first present the EvoMove sys-

tem. Then the clustering problem addressed by the move recognition unit is presented more formally, thereon the major principles behind the move recognition system are explained. Finally we present the experimental settings and the main experimental results.

EvoMove system

The EvoMove system is a musical companion that generates music on-the-fly according to a performer moves. The system is composed of three elements: an acquisition unit made of Inertial Measurement Units (IMU) body sensors that provide a continuous data stream following the motion of the performer(s). A move recognition unit that receives and analyses on-the-fly the raw data provided by the sensors. This unit is responsible of constructing and updating a clustering model of the IMU data and to identify the categories of new incoming moves. The recognition unit is based on KymeroClust, a new evolutionary clustering algorithm that has been presented in Deliverable 5.2 of the EvoEvo project (`http://www.evoevo.eu`). Finally, the sound generator unit then produces music according to the categories found by the recognition unit. The music generation relies on a tiling over time of audio samples, where each sample is triggered according to the moves detected in the data stream coming from the sensors.

Here, these units are organized according to the commensal architecture schema: The virtual organisms of the KymeroClust algorithm (move recognition unit) constitute the commensal layer. They pre-process the data received by the IMU, enabling the host (the sound generator unit) to interpret the moves and to produce sounds. These sounds influence the reaction and the moves of the dancer, closing an enactive sensori-motor interaction loop where the dancer and the system both recognize each-other (**??**).

System overview

The general architecture of the system is described in figure 1. Hereafter we expose each one of the major components of the EvoMove system with more details.

Data acquisition unit The performer moves are captured via a set of wearable wireless motion sensors built by the HIKOB company (`www.hikob.com`). The sensors are embedded in small boxes tied to the wrists, the ankles, the heads or other parts of the bodies of the dancers (part A figure 1). Each sensor is an Inertial Measurement Unit (IMU) composed of three orthogonal accelerometers, three orthogonal gyroscopes measuring angular speed, and three orthogonal magnetometers. Accelerometers and gyroscope describe the movements and the magnetic field gives a hint about absolute orientation. Each IMU delivers information at 50 Hz. The information is collected by a gateway using a client-server model, thus supporting the deployment of a variable number of sensors, and even to set them on several performers at the same time.

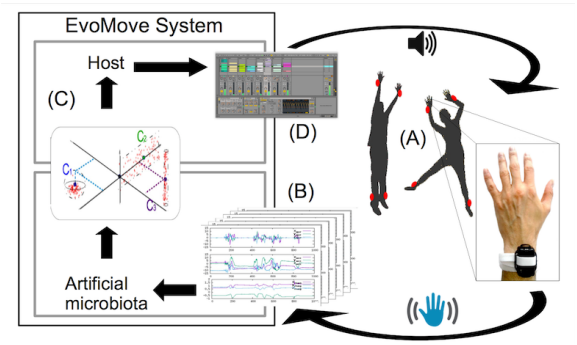

Figure 1: The EvoMove system. (A) the performer(s) moves are captured by multiple sensors, hence producing a high dimensional temporal signal (B). The signal is then processed by the KymeroClust algorithm (the "virtual commensal microbiota"). KymeroClust performs subspace clustering on the input data to exhibit groups of similar moves, and updates on-the-fly the corresponding clustering model. The current move is then associated to one of the clusters (C), and depending on this cluster an audio sample is played by the "host" (D), providing a feed-back to the performer(s).

The information collected by the gateway component is aggregated every time step. This adjustable time step is set to correspond to a beat of the music samples. For instance, if the tempo is set to 60 bpm, the time step will be one-second long. At the end of each time step, the mean and variance of the 9 measures (3 accelerations, 3 angular speeds, 3 magnetic field measures) collected during this time step are computed. This leads to 18 values describing the move that occurred during the time step. These 18 values form a data object in a space having 18 dimensions. This data object is then sent to the move recognition unit.

Move recognition unit The move recognition unit is based on KymeroClust, an evolutionary algorithm for subspace clustering used here to recognize categories of similar moves. This algorithm is based on a compact evolvable genome structure and uses bio-inspired mutational operators. It uses a median-based representation of the categories of moves that makes it resistant to noise and high dimensional datasets (see below). These features allow KymeroClust to exhibit good quality clusters when compared to well-established clustering paradigms on real and synthetic static datasets, while allowing for short runtimes.

Sound generator unit After the acquisition of a new data point, KymeroClust determines the category of moves (cluster) to which this new point belongs. Then the cluster identifier is passed to the sound generator unit of the EvoMove

system, which sets a trigger to start to play, on the next beat, the audio sample associated to this cluster identifier. The cluster-sample association is randomly determined at the first activation of a cluster. The set of samples itself is built beforehand by a musician, following timber and harmonic design principles, in order to form a coherent musical atmosphere. Finally the *Ableton Live* program is used to generate and play the music samples. From the user perspective, if the tempo is set to 60 bpm, this means that every second a description of the user moves is computed, an audio sample is started, and the system adapts its clustering model.

Medians-based subspace clustering task

In a commensal architecture, the commensal layer is made of a population of virtual bacteria that process the incoming data and evolve while doing so (those bacteria more able to produce good clusters being selected at each time step). In EvoMove, this layer constitutes the move recognition unit that constructs and updates a model of the performer moves. As explained above, the move recognition unit is based on clustering analysis, i.e., it relies on an algorithm that forms groups of moves (clusters) that share similar characteristics.

Clustering based on medians

Different clustering algorithms relying on different similarity measures and paradigms regarding the definition of clusters have been proposed. Among the major categories (**?**), clustering-oriented approaches group objects mainly using distance-based similarities and tend to build center-based hyper-spherical shaped clusters. The very well known K-means algorithm, for instance, uses an Euclidean distance to produce clusters around centroids. Other important techniques known as the K-medians approaches (**?**) are also important in solving optimization tasks. The K-medians problem is a well defined and NP-hard problem that has many different applications (reader is referred to (**?**) for examples of real world applications of K-medians). These techniques use the Manhattan distance to group data objects around medians that are less sensitive to unusual and extreme values and more robust to noise and outliers. Moreover according to (**?**) the use of the Manhattan distance should be preferred for high dimensional data mining applications, since this metric is less impacted by the problem known as the *curse of dimensionality*.

Indeed, dealing with high dimensional data reveals to be a challenging task since usual similarity measures tend to be less meaningful in high dimensional spaces. This complication leads traditional clustering techniques to struggle with high dimensional data. To overcome this problem, new clustering approaches called subspace-clustering approaches have been proposed in the literature. These approaches decompose the clustering task into two different tasks that should be solved simultaneously: detect clusters in the dataset and identify the relevant subspace of each cluster. Various concepts have been investigated in the subspace clustering community as reported for instance in (**??**). In order to take advantage of the benefits of median-based representations and subspace clustering ones, we developed a median-based subspace clustering algorithm (KymeroClust) and used it to model the performer moves. As explained above, in the commensal architecture, the objective function of the commensal layer is different from the one of the host (and from the one of the entire system). Hereafter we describe more formally the objective function that is used by KymeroClust.

Let a set of objects $\mathcal{S} = \{s_1, s_2 \dots\}$ denote a dataset. Each object in $\mathcal{S}$ has a unique identifier and is described in $\mathbb{R}^D$ by D features (the coordinates of the object). Let D denote the number of dimensions (i.e., the dimensionality) of $\mathcal{S}$. Each dimension is represented by a number from 1 to D and the set of all dimensions of the dataset is denoted $\mathcal{D} = \{1, \dots, D\}$.

A subspace clustering $\mathcal{M}$, called hereafter a $model$, is a set of centers such that each center $m_i \in \mathcal{M}$ is associated to a subspace $\mathcal{D}_i$ of $\mathcal{D}$. From the point of view of center-based subspace clustering, a cluster center described in a given subspace can be perceived informally as a summary of the cluster objects. Indeed this set of objects can be represented in a more abstract way simply by the location of the center along the dimensions considered in the cluster subspace. For a dimension d that is not in $\mathcal{D}_i$, the intended meaning is that, along d, the objects of the cluster follow the same distribution as the other objects of the dataset. For an object s, the Absolute Error is then $AE(s, \mathcal{M}) = \min_{m_i \in \mathcal{M}} dist(s, m_i)$, where $dist(s, m_i) = \sum_{d \in \mathcal{D}_i} |s_d - m_{i,d}| + \sum_{d \in \mathcal{D} \setminus \mathcal{D}_i} |s_d|$, with $m_{i,d}$ the coordinate of m_i in dimension d. The Sum of Absolute Errors (SAE) is then defined as $SAE(\mathcal{S}, \mathcal{M}) = \sum_{s \in \mathcal{S}} AE(s, \mathcal{M})$. Each object is associated to the center m_i that minimizes $dist(s, m_i)$. If several centers minimize this expression then the object is non-deterministically associated to one of them. Finally, the size of a model $\mathcal{M}$, noted $Size(\mathcal{M})$, is defined as the sum of the dimensionalities of each subspace associated to the centers in $\mathcal{M}$, and is interpreted as the level of detail captured by the clustering. To perform such a median-based subspace clustering, the core task performed by KymeroClust is to evolve models $\mathcal{M}$ (one per individual) that minimize the SAE and such that $Size(\mathcal{M}) \leq SD_{max}$, where SD_{max} is a parameter denoting the maximum *Sum of Dimensions* used in $\mathcal{M}$ to define all the subspaces. Since the algorithm aims to minimize the $SAE(\mathcal{M}, \mathcal{S})$, this objective function can be seen as the *phenotypic error* of the individual encoding $\mathcal{M}$, in the environmental context of the dataset $\mathcal{S}$, and its fitness can be computed taking the opposite of the *phenotypic error*. In terms of commensal architecture, the success of the commensal layer is thus linked to its ability to efficiently model the incoming data, regardless of the quality

of the music. Note however that if the music is not recognized by the dancer, then he is likely to modify his behavior, thus producing new kinds of moves and large changes in the dataset, leading to directional selection in the commensal layer.

KymeroClust

KymeroClust is based on a phenotypic representation similar to ChameleoClust, an evolutionary subspace clustering algorithm developed previously by the EvoEvo consortium (**?**). As ChameleoClust, KymeroClust takes advantage of an evolvable representation of the genotypes to adapt the numbers of clusters produced and the subspace dimensionalities according to the performer moves. However, since KymeroClust must be able to update its model and to compute the relevant cluster in real time (i.e. while the dancers dance), we used an indirect coding of the genome structure and new bio-inspired mutation operators to increase the algorithm efficiency, both in terms of runtime and adaptation speed. The complete description of KymeroClust and its comparison to the main subspace clustering methods can be found in **?**.

Genotype-Phenotype representation

As in ChameleoClust, in KymeroClust each individual phenotype is a model $\mathcal{M}$, i.e., a set of centers each being defined in its own subspace. To allow for an efficient computation, individuals are encoded by a dual structure that allows to encode at once an implicit representation of the genome and an explicit representation of the phenotype of the individual thus avoiding to compute the phenotype of each individual at each generation. The biological analogs of KymeroClust subspace-clustering paradigm is the following: each center is a phenotypic trait composed of several biological functions – a function being the coordinate of a center along a given dimension. Then, a given gene contributes to one and only one function but a given function can be performed by several genes, possibly carrying different alleles. The genes are implicitly represented and we only store the number of genes involved in a given biological function (i.e. a coordinate) of a given phenotypic trait (i.e. a center) as well as their mean contribution. This level of description is flexible enough and enables to implement a gene-duplication-divergence mechanism while being abstract enough to avoid the use of a genotype to phenotype mapping function to compute phenotypes from genotypes at each generation.

The implicit representation of the genome is based on a weighted model. In KymeroClust each individual is encoded by a pair of matrices $\mathcal{M} = \langle \mathcal{L}, \mathcal{W} \rangle$ that respectively describe the phenotype of $\mathcal{M}$ and the genotypic structure that encodes it. More precisely, $\mathcal{L}$ encodes the values of the biological functions: An element $\mathcal{L}_{i,d}$ represents the coordinate of a center m_i along dimension d (thus defining the phenotypic function). The matrix $\mathcal{W}$ encodes the number of genes encoding a given biological function: An element $\mathcal{W}_{i,d}$ denotes the number of genes encoding the coordinate $\mathcal{L}_{i,d}$.

Given $\mathcal{L}$ and $\mathcal{W}$, the subspace associated to a center m_i is $\mathcal{D}_i = \{d \in \{1, \ldots, D\} | \mathcal{W}_{i,d} > 0\}$. Valid centers are those for which at least one dimension is defined, and their set of index is simply $\Gamma = \{c \in \{1, \ldots, SD_{max}\} | \sum_d \mathcal{W}_{c,d} > 0\}$. To permit to encode a model of size SD_{max} but no more, both matrices have SD_{max} rows and D columns.

Variation operators

The exploration of the space of possible clustering models is ensured here by two bio-inspired mutational operators: gene duplications-divergence and gene deletions. Given the structure of the individuals, these operators revealed to be key elements of the algorithm since they allow a joint exploration of the space of solutions and of the space of possible genome structures.

It is well known that duplicated genes have an important role in evolution (**?**) as they constitute raw material that can evolve through divergence, thus facilitating the acquisition of new functions and allowing individuals to quickly adapt to new environments. However, most of the duplicated genes are deleted (reverted) just after being created, especially if the gene duplication leads to deleterious consequences related to dosage effects (**?**). Indeed, new functions emerge only when one copy mutates (divergence). Hence the appearance of new functions via gene duplication-divergence is slow, indirect and fragile (one copy can be deleted anytime). Therefore we decided to group gene duplication and gene divergence in a single operation. Moreover, in order to avoid deleterious dosage effects, multiple genes coding for the same biological function have a non-additive effect on it. Indeed gene multiplicity only has an impact on the robustness (backup genes) and the evolvability (larger mutational target) of the biological function.

Duplication-divergence operator When a duplication occurs, a gene is randomly chosen on the genome (i.e. a non-null row in the matrix $\mathcal{W}$). This gene is duplicated and immediately mutated. The mutation has two possible outcomes: Functional divergence or allelic divergence. With a low probability functional divergence gives birth to a new phenotypic trait hence creating a new center in the phenotype. Otherwise the functional divergence leads to the acquisition of a new biological function that still contributes to the same phenotypic trait as the original gene (the center acquires a new coordinate along an unexplored dimension). If the duplicated gene undergoes allelic divergence, the value of the corresponding biological function is modified (the coordinate is changed). In both cases, the new function is randomly chosen in the domain of the possible functions. This choice is supported by the fact that new proteins created through gene duplication and subsequent divergence

may still be involved in reactions related to the role of the original protein (**?**).

The duplication-divergence operator modifies both the genome structure ($\mathcal{W}$) by increasing the number of genes associated to the biological function involved and by modifying the given biological function (the center location in $\mathcal{L}$) thus changing the phenotypic representation. Importantly, in KymeroClust, gene mutations (i.e. divergence) are not uniformly chosen. Indeed, when a bacteria faces a new environmental challenge, its evolutionary fate is partly predictable and partly not. Indeed, we KNOW that the bacteria will adapt by integrating information from its environment but we DO NOT KNOW how it will do it at the genetic level (**?**). We model the unpredictable part by choosing randomly the genes that are duplicated (or deleted – see below). On the opposite, the predictable part lies in the divergence process: In case of a divergence (either functional or allelic), the set of values that can be taken by the new biological functions (i.e. center coordinates in $\mathcal{L}$) are randomly chosen among the coordinates of the available data objects. Then the exploration space is restrained and it is easier for the individuals to acquire new promising functions, leading the algorithm toward better candidate solutions.

Deletion operator Most of the time the deletion operator modifies only the genome representation by decreasing the number of genes associated to the function. This operator only modifies the phenotype when no more genes encode the function after the deletion, in this case the function is suppressed from the phenotypic trait (i.e. the location of the center along the given dimension is set to zero in $\mathcal{L}$).

Genome size and model size The algorithm uses the data objects themselves to generate the children of a model $\mathcal{M}$ through mutations: A child of a model $\mathcal{M}$ is a model that can be obtained from $\mathcal{M}$ by inserting or removing a dimension in a subspace and setting a center coordinate to a value equal to one of the object coordinates (gene deletion and gene duplication/divergence operators). However, this may result in the indefinite increase of the genome size and possibly to very bad clustering performance (e.g. one cluster per object in the dataset). That is why we set a maximum genome size SD_{max} (the genome size being defined as the total number of genes in the genome: $\sum_{i,j} \mathcal{W}_{i,j}$). It is worth to say that SD_{max} constrains the *maximal* size of the subspace model encoded in the genome. However, the algorithm has still an important degree of freedom to adapt the size of the subspace clustering model it encodes. This flexibility is ensured by duplicate genes since more than one gene can encode the same biological function and thus the size of the resulting model can be smaller than the genome size itself. In addition KymeroClust can also encode centers that correspond to empty clusters, in this case associated genes can be seen as non-functional genes. This enhances the ability of the algorithm to adapt the number of clusters it produces.

Evaluation and selection

In practice the SAE is not evaluated on the entire dataset, KymeroClust relies on a sliding window $\tilde{\mathcal{S}}$ that only keeps the latest data objects (i.e. the latest moves of the performer) and assess the SAE on it.

Each time step a new point from the data stream is received by KymeroClust, the oldest object from the sliding sample is replaced by the new point. From an evolutionary point of view, this corresponds to an environmental modification. The effect of the new environment on the population can be assessed by computing the SAE (of the best individual) in the new environment. This can be done very efficiently: the AE associated to the oldest point is deducted from the SAE while the new point is assigned to its closest clusters and the AE corresponding to this point is added to the SAE. Now, either the new SAE can be larger or smaller than the former one. This corresponds respectively to challenging and non-challenging environmental variations. In case of non-challenging variation, the selection pressure on the population is supposed to be stabilizing and no mutation is fixed (in practice no new generation is computed). In case of a challenging environment variation, i.e. if the SAE of the best individual increases, the selection pressure is supposed to be directional and a new generation is computed following a $(1, \lambda)$ Evolutionary Strategy selection scheme: The model $\mathcal{M}$ with the lowest phenotypic error (SAE) is selected to generate λ children. Then the offspring undergoes mutation and the SAE of the new candidate models are evaluated. Finally only the best individual, among the λ children and the parental model $\mathcal{M}$, is chosen to populate the next generation.

Initialization KymeroClust is organized in two different phases, an initial phase called *construction phase* operates during the first generations and a *stationary phase* that operates then until the end of the performance.

The algorithm takes as initial individual $\mathcal{M}_\emptyset$, the empty model (a model containing no center), for which the definition of AE is extended as $AE(s, \mathcal{M}_\emptyset) = \sum_{d \in \mathcal{D}} |s_d|$. During the *construction phase*, whenever a new individual is produced by replication, one duplication-divergence operation is applied, but no deletion is performed. Consequently the genomes of the individuals are progressively built from scratch along generations by successive duplications-divergences until genomes reach the genome size SD_{max}. Once the *construction phase* ends, the algorithm enters in a *stationary phase* and remains in this phase until the end of the run. During this phase, each time a replication occurs in directional selection, the gene deletion operator is applied before the gene duplication-divergence operator to keep the genome size constant (note that, as explained above, this

does not mean that the model size is constant during the stationary phase).

Parameter setting At the beginning of each EvoMove session, the user chooses the set of audio samples that the system will use during the performance, the number of sensors, a well as the main parameters of the KymeroClust algorithm. This parameter setting was the same for all experiments reported here. The maximal genome size was set to $SD_{max} = 100$. The model could for instance build a maximum of 10 clusters with subspaces having 10 dimensions as well as 100 clusters in mono-dimensional subspaces. The number of objects in the sliding sample used to assess the fitness of the individuals was set to $|\tilde{\mathcal{S}}| = 100$. Finally, in order to largely explore the space of candidate solutions, the number of children produced at each generation was set to $\lambda = 100$.

Experiments

During the prototyping and tests of the EvoMove system, we had the occasion to work with different people to go deeper in our understanding of the behavior of the system and to what extend it could be used and enhanced in order to help dancers in their creation/performance process. In particular we had the occasion to work with professional dancers from the Anou Skan dance company (`http://www.anouskan.fr`, later-on called the "Anou Skan experiment") and from the Désoblique company (later-on called the "Désoblique experiment"). Two videos of the Anou Skan experiment can be found at `https://www.youtube.com/channel/UCoyfXJx_izpQZi6hD8w5M3A`. In addition the EvoMove system has also been used in public performances Meute (Pack in English) in February and March 2017 (by Claire Lurin - Désoblique danse company, and performers: Claire Lurin, Jean Boulvert, Maxence D'Hauthuille and Jonas Abernot).

In this section we first describe the settings of the EvoMove system used in all these experiments, then we will illustrate the EvoMove performance by commenting the online videos and discuss the dancers and the audience perception of the system. We will then conclude by discussing the performances of the EvoMove system and providing insights regarding the conception of a personal living companion.

Sensor setting Beyond the preprocessing choices (computing means and variances over one time step), and the KymeroClust settings, the interaction between EvoMove and the performer(s) can be configured. The first critical choice is of course the number and position of the sensors on the bodies of the performers. In the Anou Skan experiment, we used two sensors attached to the wrists of one dancer. During the Désoblique experiment Claire Lurin used up to 4 sensors. One was attached to her right wrist, one to her left ankle, one in her hair and one on the abdomen. In the public Meute performance, three dancers were equipped with sensors.

Importantly, the number of KymeroClust instances used to process the collected data can be different from the number of sensors. One can use a single instance collecting all the data or one instance per sensor or whatever combination (e.g. one instance for the two arms and one instance for the two legs). In the Anou Skan experiment accessible online, the two sensors were collected by two different instances of KymeroClust running at the same time, each of them using its own set of audio samples. However, this setup is only one of the numerous possible ones with this system. As experienced during the public performance, several users could wear sensors, several sensor measures could be combined to describe moves (e.g., using the difference between two measures or their product) and data captured by different sensors could be sent to the same KymeroClust instance. In some of our experiments, sensors have been added, removed or even passed from one performer to another one on the fly, without requiring any reconfiguration of the system. This versatility is one of the benefit of the commensal architecture since the commensal layer adapts dynamically to the new conditions, continuously delivering coherent output to the host layer.

Audio setting The second critical choice is the audio setting. Apart from the choice of the audio samples themselves and from the music tempo (set to 60 bpm in all our experiments), one can choose when the sample will be played. In all our experiments, a given sample was triggered at each time step. More precisely, at the end of the time step, the data collected during this time step is associated to a cluster identifier that is mapped to an audio sample to be played on the next beat. Importantly, the audio samples used in the experiments were from one time-step up to four time-steps long (most of them spreading over four time steps). This gives a feeling of "trails", moves initiating sounds that will fully stop later. When a long sample was triggered at consecutive time steps, it was restarted at its beginning for each time step and was only entirely played at its ultimate activation.

User perception and system behavior Interesting questions that have motivated the EvoMove demonstrator involve the interaction between the users and the EvoMove "living" personal companion. Does the user feels that an interaction with the system is established? Does it seem to the user that the system behaves randomly or that he is able to influence the system in some directions?

During the trial sessions that were organized with different users to analyze the system, we also asked them about their feeling about the system and their interaction with it. These tests were performed with people having different

backgrounds and different approaches of the system, ranging from people involved in the development of the project, to professional dancers and also musicians. Hereafter we provide three examples of the mental representations and feelings of three users regarding their interactions with the system. These descriptions were collected after the first trial of the system by the users.

- When using EvoMove, one of the developers of the system, had a geometrical representation of the system, feeling himself placing points in a multidimensional space, so as to push and pull candidate cluster centers around in this space. This feeling enables him to create clusters but also to adapt them to his will.
- A musician described himself as feeling involved in a taming relationship with the system, trying to repeat gestures so as for the machine to memorize it, trying to insist on some distinctions between gestures, just as if the user was teaching a trick to an animal.
- A dancer perceived the system as a monitoring tool for her moves, that was noticing small move differences by playing a different sound, as if a specialist (a dance teacher) was checking the correctness of her moves.

From the discussions we had with the users, it appeared that they had very different inner representations of what the system was doing and they also had different feelings regarding their interaction with the system. In particular they imagined different applications of the system, ranging from its use as a dance improvisation companion to its use as an electronic instrument that could be mastered to control what sound will be played by the system. These observations are rather encouraging, since they illustrate the wide scope of applications of the EvoMove system and also suggest that the users perceive that the system interacts with them.

Interestingly, some emergent behaviors of the systems that were not predicted have been observed during the working sessions. Let us illustrate this on the video EvoMove AnouSkan 1 (`https://www.youtube.com/watch?v=p_eJFiQfW1E&t=141s`).

- Biological organisms are immersed in environments that constantly change. Hence they must adapt constantly to new environments. However they can do so through the acquisition of new traits or, more basically, by adapting existing traits. In KymeroClust, when a new move is introduced by the user, the system can create a new center (phenotypic trait), but, if the new move is close to a one previously observed, KymeroClust can simply drift an existing center to account for the modification. An example of this behavior can be observed in the video from timestamp $1'28''$ to $1'54''$. During this period, the dancer slightly transforms her moves at each repetition, but the sound remains the same along these thirty seconds. The variations of the dancer moves are followed by the system through the gradual changes of the associated center.
- Some phenotypic traits are specific to a given environment, therefore if an individual is never confronted to this environment, the genes involved in the trait are not used (biologically, the genes are not activated). As a consequence there is no selective pressure to remove these genes and they can be conserved for many generations. In the EvoMove this allows moves that have been played once to be recognized later on, even though they were not activated for a period longer than the observation window that is 100 seconds long (a sample of 100 objects received at a rate of one object per second). This can be observed in the video EvoMove AnouSkan 1: three occurrences of a circular walk (timestamps $2'00''$, $'4'10''$ and $6'00''$) are accompanied by the same sounds.

Conclusion

Most of the state-of-the-art move recognition systems used in music control, rely on many specific sensors and preprocessing steps, combined with supervised learning algorithms. On the contrary, thanks to its commensal architecture, the EvoMove system uses a few generic sensors, that can be placed anywhere on the body and changed at anytime: The music produced by the host (sounds generator) depends on a model dynamically updated by the virtual commensal organisms (KymeroClust), which themselves depend on the moves the user executes in reaction to the music heard. Hence, EvoMove creates a feedback loop including the human user. Moreover, since the timing of the loop iteration is rather short (around one second), the EvoMove enables a direct, enactive, interaction between the system and the user. Interestingly, here the user does not have a full control over the system, but he can influence it, and contrary to most software where the human acts *on* a system, here the user is acting *in* the system. Indeed, we hypothesize that the efficiency of EvoMove could be partially related to the integration of the human user in the feedback loop. Indeed, since the dancer receives a direct feedback from the system regarding his previous action, he can adapt (even unconsciously) his actions in order to be understood by the system. This behavior depends on the representation the user has of the system, and therefore this mental image could help the dancer to maintain and enrich the interaction with the system.

EvoMove opens a lots of possibilities. An interesting perspective would be to use the cluster centers to directly control the sound generation as in (**?**). Another possibility would be to let the samples evolve as in (**?**), rather than relying on predefined samples. One could also use the same architecture for other purposes like rehabilitation (**?**) or to control other systems: A version in which the host produce drone commands rather that music samples is being developed. The objective here is to test whether the drone can

act as a dance (or game) companion. However, the most exciting perspective is to explore the artistic landscape opened by this "living" musical companion. Indeed, while several examples of systems (**?**) or performances (Variations V by Merce Cunningham and John Cage, or Virus/Antivirus by Cie Lanabel), also use motion sensors, the novelty of EvoMove is to let open the mapping between the motion space and the sound space. Indeed, the correspondence between sounds and movements is not chosen in advance, it is built on-the-fly depending on the moves made and repeated by the performers. So, the performers have to find their own way through a musical landscape generated by them and for them.

To conclude, it is worth noticing the reaction of the audience when the EvoMove system has been used in the public Meute performances. The artists chose not to inform of the EvoMove use, thus the audience was *naive*. After the performances we collected reactions, and actually we were not able to find someone having noticed something special (even a person knowing that the system was used admitted that he forgot it during the show). Whether this is good or bad news is a matter of discussion...

Acknowledgements

This research has been supported by EU-FET grant EvoEvo (ICT-610427). The authors would like to thank all the partners of the project for fruitful discussions. They also would like to thank the Anou Skan and the Désoblique Dance Companies. Many thanks to Leo Lefebvre and Anthony Rossi who contributed to the development of some modules used in the system. Other support: Christophe Rigotti is a member of LabEx IMU (ANR-10-LABX-0088).