A Collaborative System for Composing Music via Motion Using a Kinect Sensor and Skeletal Data

Christos Garoufis, Athanasia Zlatintsi and Petros Maragos
School of Electrical and Computer Engineering, National Technical University of Athens
cgaroufis@hotmail.gr,nzlat@cs.ntua.gr,maragos@cs.ntua.gr

ABSTRACT

This paper describes MoveSynth, a performance system for two players, who interact with it and collaborate with each other in various ways, including full-body movements, arm postures and continuous gestures, to compose music in real time. The system uses a Kinect sensor, in order to track the performers' positions, as well as their arm and hand movements. In the system's current state, the musical parameters that the performers can influence include the pitch and the volume of the music, the timbre of the sound, as well as the time interval between successive notes. We extensively experimented using various classifiers in order to detect the one that gives the optimal results regarding the task of continuous gesture and arm posture recognition, accomplishing 92.11% for continuous gestures and 99.33% for arm postures, using an 1-NN classifier with a condensed search space in both cases. Additionally, the qualitative results of the usability testing of the final system, which was performed by 9 users, are encouraging and identify possible avenues for further exploration and improvement.

1. INTRODUCTION

The connection between motion and sound has always been of particular interest to humans [1]. However, while responding to sonic input via movements of the body has been practiced since antiquity, the composition of sound from motional input has only recently been explored. The first chronologically tangible result of the above exploration is the theremin [2]. Designed to produce sound without physical contact between the performer and itself, it utilizes two oscillators, the resonant frequency of which is determined by the distance between the performer's hands and the respective antennas. Both the pitch and the volume of the produced sound are directly dependent on the resonant frequencies of the oscillators.

Since then and due to the recent advances in sensors and motion tracking technology, a lot of ground has been covered in the design of systems that compose music using spatial or gestural data [2]. Of particular importance to the above was the launch of Microsoft Kinect [3] in 2010. Microsoft Kinect consists of a variety of sensors, capable of transmitting color, depth and IR data in a maximum rate of 30 frames per second. However, the main reason Kinect is widely used in the design of interactive performance systems is the ability to accurately track a number of skeletons (up to 2 and up to 6 respectively for Kinect v1 and v2), each represented as a number of keypoints (20 and 25 respectively for Kinect v1 and v2) [4], corresponding to various human joints, including a central body joint and the shoulders, elbows, and wrists for each arm.

Herein, we present MoveSynth, an interactive performance system for two players who compose music in real time. More specifically, in Sec. 2, we present the related work in the fields of interactive performance systems and gesture recognition. In Sec. 3, we describe our system in more detail, explaining its architecture in depth, while also describing and using a series of offline experiments to support the final design choices we made. In Sec. 4, we present results regarding the system's quantitative online evaluation as well as a pilot quality of experience (QoE) study, while our conclusions and potential future work and extensions are discussed in Sec. 5.

2. RELATED WORK

2.1 Interactive Performance Systems

The work presented in this paper is about an interactive performance system for two players, which synthesizes music depending on their movements. In the related literature, there are two possible ways of interaction between the system and the performers; the performers either compose the resulting music themselves [5], or tune some parameters of the music - like the volume and the pitch - that the system algorithmically composes [6].

Part of the literature focuses on virtual representations of musical instruments. In these cases, the performers produce music that corresponds to the respective instrument, by performing movements similar to those that would be performed in order to play the actual instrument. These can be tracked using either cameras or other sensors, such as accelerometers and gyroscopes. Examples of the above include enhanced traditional musical instruments in a digitalized form [7], assistance tools for learning how to play an instrument [8], or purely entertainment applications [5, 9]. A large number of these applications use Microsoft Kinect as a tracking sensor. The work in [5], for example, provides a virtual interface for three musical instruments - a guitar, a drum kit and spider king - by tracking the per-

former's movements using Kinect. To increase the accuracy of the instrument's representation, by offsetting the inherent latency of the Kinect, the velocity and acceleration of the performer's hands are taken into account in some cases [10]. For more abstract interactive systems, a popular option is using movements of either some of the performer's body parts, such as their hands or arms, or the performers themselves, as event triggers, and translating them into changes in the pitch, the volume, or the melody of the played music [11].

An intriguing challenge in the design of a digital musical instrument is the creation of a mapping between sounds and motion that is both intuitive and easy to learn [2]. A good example of such a mapping is linking the rhythm of the played music to the speed of the performer's movements, since it satisfies both of these conditions [12]. Other interesting mapping strategies include spatiality-based mappings, where spatially close arm postures correspond to sounds with similar properties [13] and the use of vision-embodied metaphors to musical features [14, 15].

Finally, it is interesting to note that the connection between motion and music can be used for educational purposes, since some musical concepts are easier to explain through linking them to a motion counterpart [12]. An example of such an application is Robinflock [6], designed to teach children the concept of polyphony in music.

2.2 Gesture Recognition from Skeletal Data

Gesture recognition is the research field that focuses on the automatic extraction of meaningful data from the motion of a person's hands, arms, face, head, or body [16]. Its ultimate goal is the classification of these meaningful data into some kind of gesture language or non-verbal commands. Since the launch of Kinect, an emerging subfield is that of classifying gestures using skeletal information.

The most commonly used approach to this problem involves the use of a distance metric between the observed gesture and a number of templates. The metric is applied either on the raw skeletal data [17, 18], or some intermediate representation, such as decompositions of the gestures in simpler movements [19] or the correlation values of the angles between the skeletal joints [20]. In order to take into account the temporal development of the gestures, dynamic time warping is used to calculate the distance metric, and the features are weighted either globally [17] or locally [18]. Other approaches include the use of Hidden Markov Models for modeling the movements of the joints [21], and classifying the gestures - after extracting an intermediate representation - via Support Vector Machines instead of a nearest-neighbor based distance metric [22, 23].

3. MOVESYNTH: A COLLABORATIVE SYSTEM FOR COMPOSING MUSIC VIA MOTION

In this section, we present our system, MoveSynth, and the various subsystems that compose it. It has been designed to operate as an interactive performance system, with two players controlling various musical parameters, such as its volume, pitch and the time interval between successive notes, depending on their full body motion, hand position, arm posture, as well as the execution of a number of continuous gestures.

3.1 Mode Description

MoveSynth includes the following modes, each featuring different ways for composing music and collaborating. Navigating through these modes is achievable by performing specific gestures and movements and, in all of these, the performers' roles are dependent on whether they stand to the right or the left of the camera. In all modes, sine waves of a single frequency, corresponding to specific music notes, are produced [24].

- Mode 1: The pitch of the music changes in discrete time intervals. The performer to the right of the camera plays music, by mapping their arm posture to specific notes. The one on the left is responsible for starting the music, stopping it, defining the time interval length and switching to another mode, via performing a number of gestures, as well as tuning its volume, decreasing it as she/he moves away from the camera.

- Mode 2: The pitch of the music changes quasi continuously. The performer to the right of the camera plays music by controlling both the pitch and the volume via adjusting vertically the position of his/her hands. The performer on the left keeps the same mapping as in mode 1 for all gestures, with the exception of those responsible for controlling the time intervals between successive notes. Since the sound is quasi-continuous in this case, these gestures were re-mapped to control the timbre of the sound, which is altered by the addition of higher-order harmonics to the initial sinusoid.

- Mode 3: Similar to mode 1, the pitch of the music changes in discrete time intervals. The performer on the right plays music exactly as in mode 1. The one on the left controls the volume of the music and the time interval between successive notes, adjusting these parameters by vertically moving their hands. This mode cannot be initiated or stopped on its own, while, by by performing specific full-body movements, the performers can switch back to another mode.

The performers' roles in each of these modes are summarized in Table 1.

3.2 Feature Extraction

Since we are using Kinect in order to track the performers' movements, we have access to the (x, y, z) coordinates of their skeletal joints with respect to the camera, including those of the shoulder, elbow and wrist of each arm. For the cases where the controlling parameter is just the position of the hand, these are enough with regards to the information that is needed. However, their inability to generalize as features means that, in order to classify both arm postures and

Mode	Performer 1	Performer 2
1	Note Playing via Arm Postures	Start/Stop, Intervals via Hand Gestures, Volume via Distance
2	Note Playing and Volume via Hand Movements	Start/Stop, Timbre
3	Note Playing via Arm Postures	Volume and Intervals via Hand Movement

Table 1. Summary of the roles of the two performers in each of the modes of our system.

continuous gestures, we have to extract from them an intermediate, meaningful representation. Thus, we calculate, for each frame, the direction vectors for each arm using the methodology described below. For the rest of this section, we will consider that the user's arm can be divided in two parts, the upper arm above the elbow and the forearm below it.

For each of the performers, beginning with the (x, y, z) coordinates for each arm joint, we calculate, for each direction, the length of each upper arm and forearm as follows:

$$x_{\text{up}} = x_{\text{shoulder}} - x_{\text{elbow}}, \tag{1}$$

$$x_{\text{low}} = x_{\text{elbow}} - x_{\text{wrist}}, \tag{2}$$

$$y_{\text{up}} = y_{\text{shoulder}} - y_{\text{elbow}}, \tag{3}$$

$$y_{\text{low}} = y_{\text{elbow}} - y_{\text{wrist}}, \tag{4}$$

$$z_{\text{up}} = z_{\text{shoulder}} - z_{\text{elbow}}, \tag{5}$$

$$z_{\text{low}} = z_{\text{elbow}} - z_{\text{wrist}}. \tag{6}$$

And we concatenate these lengths in a single direction vector per arm part:

$$\text{n\_cam}_{\text{up}} = [x_{\text{up}},\ y_{\text{up}},\ z_{\text{up}}], \tag{7}$$

$$\text{n\_cam}_{\text{low}} = [x_{\text{low}},\ y_{\text{low}},\ z_{\text{low}}]. \tag{8}$$

where n_cam_{up} and $\text{n\_cam}_{\text{low}}$ denote these concatenated vectors in the camera's coordinate system.

Afterwards, we normalize them to unit length, to cancel out the inherent variance due to the different natural characteristics of the performers. To do this, we first calculate, for both arms, the total length of each arm part:

$$\text{len}_{\text{up}} = \sqrt{x_{\text{up}}^2 + y_{\text{up}}^2 + z_{\text{up}}^2}, \tag{9}$$

$$\text{len}_{\text{low}} = \sqrt{x_{\text{low}}^2 + y_{\text{low}}^2 + z_{\text{low}}^2}. \tag{10}$$

and then, we divide each element of the respective vectors with these lengths.

Finally, after rotating all four - two per each arm - normalized vectors to align them with the performer's viewing direction, we concatenate them in a single 12-dimensional vector per frame. So, let n_cam be a normalized direction vector for an arbitrary arm part, expressed in the coordinate system of Kinect. To align it with the user's coordinate system, we multiply it with the following rotation matrix R, where θ denotes the angle between the player and the camera:

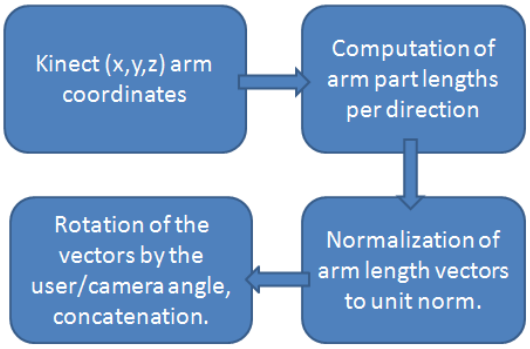

Figure 1. Block diagram for the extraction of the intermediate geometric representation of the arm posture.

$$R = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix}. \tag{11}$$

After alignment and correction of the x-coordinate of the vectors corresponding to the left arm, we concatenate all partial hand vectors in a single feature vector, n_final, as seen in Eq. 12. We do not correct the x-coordinate of the vectors corresponding to the right arm, since we want the x-axes of the coordinate systems of both hands to extend outwards.

$$\text{n\_final} = \begin{bmatrix} \text{n\_user\_upper\_left} \\ \text{n\_user\_lower\_left} \\ \text{n\_user\_upper\_right} \\ \text{n\_user\_lower\_right} \end{bmatrix}. \tag{12}$$

This process is summarized in Fig. 1.

3.3 Activity Detection and Gesture Recognition

For the real-time recognition and classification of continuous gestures, we have developed a three-stage pipeline. Its stages include an activity detector, a classifier and a control mechanism, that only accepts as valid gestures those that satisfy a similarity threshold to one of the gestures included in our gesture set. The set of gestures used to train the pipeline was a subset of the Microsoft Research Cambridge-12 (MSRC-12) gesture dataset [25], which consists of sequences of human skeletal body part movements, represented as body part locations. Specifically, we used the gestures encoded as G1, G3, G5, G9 and G11, since they only involve arm movements encoded as skeletal data, and are easily separable from each other. Fig. 2 shows the three-stage pipeline of the gesture recognition, while the gestures are partially visualized in Figs. 3-7.

3.3.1 Activity Detector

In order to detect the starting point of a continuous gesture, we built an activity detector. In applications that use videos as input, the optical flow [26] between successive frames is usually computed to determine whether there is any activity or not; in our case, we use skeletal data, and

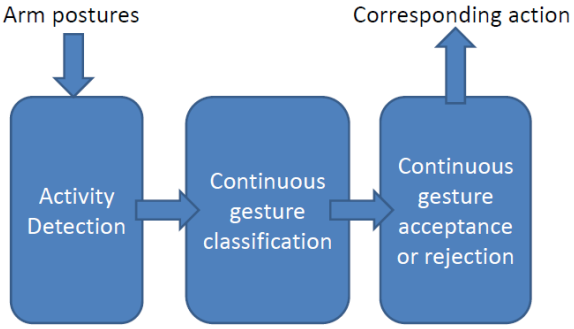

Figure 2. Block diagram of the three-stage pipeline for gesture recognition.

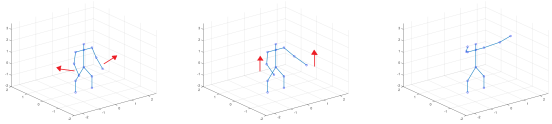

Figure 3. Three snapshots of the execution of the gesture G1

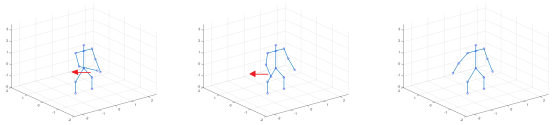

Figure 4. Three snapshots of the execution of the gesture G3

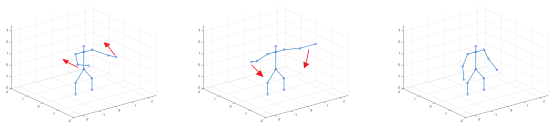

Figure 5. Three snapshots of the execution of the gesture G5

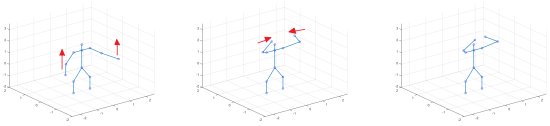

Figure 6. Three snapshots of the execution of the gesture G9

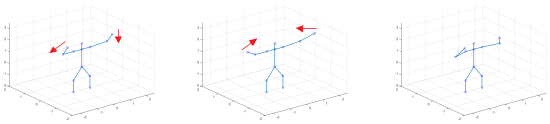

Figure 7. Three snapshots of the execution of the gesture G11

an estimation of the velocity of the arm joints is considered sufficient to this end. Therefore, we need to find a threshold value for the velocity that will, when exceeded, detect the starting point of continuous gestures. In order to accomplish this, we calculated the average per-frame velocity both in every gesture instance in the dataset and in frames between successive gesture instances, where no gesture was performed. The threshold value was chosen so as to split the respective probability distributions, as seen in Fig. 8. After experimentation, the threshold value was

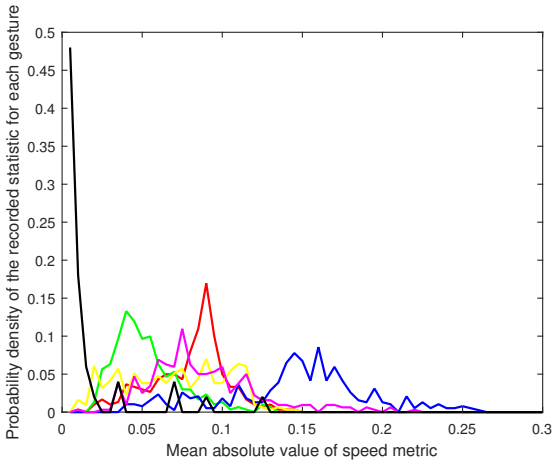

Figure 8. Statistical visualization of the velocity metric during various gesture (colored according to the corresponding gesture) and non-gesture (black) instances. (Please see color version for better visibility)

set to be equal to 0.05.

3.3.2 Continuous Gesture Classification

After any positive activity detection, the skeletal data for the next 31 frames (preprocessed as described in Sec. 3.2) are transmitted to the classifier. The duration of each gesture was taken to be equal to 1 second, and thus, taking into account the 30 fps frame rate of the Kinect sensor, 31 frames are required to cover, time-wise, the gesture duration. In order to choose the optimally performing classifier, we extensively experimented, using 5-fold cross validation, with Hidden Markov Models (HMMs), with either discretized or continuous observation variables, and the k nearest-neighbor algorithm (kNN). For each case, the following hyperparameters were tuned, in order to determine the optimal setup:

- HMMs with discretized observation variables: Number of codewords (up to 80) and number of states (up to 7).
- HMMs with continuous observation variables: Number of Gaussian mixtures per state (up to 5) and number of states (up to 7).
- kNN: Number of voting neighbors (up to 7).

Based on the recognition results of the different classifiers, as shown in Table 2, the nearest-neighbor classifier outperformed the HMM classifier accomplishing an accuracy of 98.6%. In order to further enhance the results, we applied a number of modifications on it. Initially, we experimented with reducing the size of the training set using k-means, in order to reduce the runtime of 1-NN. Consequently, and after concluding that keeping 20 templates per class provides optimal runtime with minimal accuracy loss:

- We applied Principal Component Analysis (PCA) in our data, to reduce the number of features per gesture, experimenting with the number of kept components.

Algorithm	Optimal Setup	Accuracy
Disc. HMMs	7 states, 80 codewords	93.1%
Cont. HMMs	5 states, 5 mixtures	97.1%
Nearest Neighbor	1 voting neighbor	**98.6%**

Table 2. Accuracy percentages (%) for the 3 baseline algorithms we experimented with. "Disc". denotes discretized observation variables, "Cont.", continuous observation variables.

Modification	Optimal Setup	Accuracy
Dataset reduction	20 templates/class	97.2%
PCA	20 features/template	97.3%
DTW	5-width search	97.8%

Table 3. Accuracy percentages (%) for the various modifications of the nearest-neighbor algorithm we experimented with.

- We also used Dynamic Time Warping (DTW), in order to allow for slight discrepancies between the train and test data, with regard to the exact temporal execution of the continuous gestures.

The best results of this experimentation are provided in Table 3. It is worth noting that, as a result of these experiments, we used a nearest-neighbor-based classifier in the final application, using 20 templates per class. Despite the fact that both PCA and DTW outperformed their baseline, neither was used in the final application since the accuracy gain was not large enough to overcome the real-time constraints.

3.3.3 Continuous Gesture Acceptance/Rejection

Finally, in order to determine whether a performed and classified continuous gesture actually belongs to our gesture set, we developed a gesture-rejecting system that works as follows: Given the distance, as returned by the nearest-neighbor algorithm, between a performed gesture and its closest template from any class, this metric is compared to a class-specific threshold. If the threshold is surpassed, then the gesture is assigned to a "None" category. Otherwise, the performed gesture is accepted. In order to determine the optimal values for the thresholds, we followed a procedure similar to the one for the activity detector. Thus, after dividing the gesture instances in train and test data, and compressing the train data to 20 templates per class using k-means, we calculated the distance between every test instance and each class, choosing, for each class, the minimum out of all its respective template distances. Afterwards, we used them to generate the probability distribution of the instance-template distances for each class, for both the correct class and non-correct classes. Finally, we selected as threshold value the one that optimally splits these distributions. An indicative visualization of these distributions is shown in Fig. 9.

3.4 Arm Posture Classification

The frequency range of our system was decided to be equal to an octave, ranging from C5 to C6. As a result, for all our

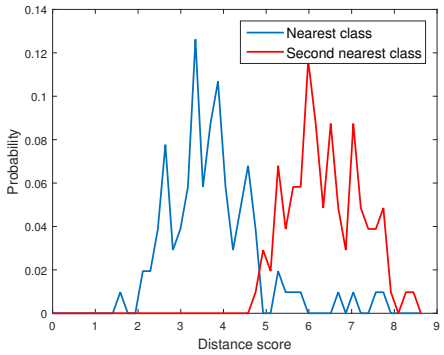

Figure 9. Statistical visualization of the distances of an instance to a) all templates in its closest class, b) all templates in its second closest class. In this case, the distance threshold that better splits these distributions is approximately 5. (Please see color version for better visibility)

operation modes apart from mode 2, we needed to define an arm posture that would produce each note, leading to a total of 12 different arm postures. The classification of the arm postures is done by using a simple template matching algorithm, with one template per class. Again, the arm postures are represented by the 12-dimensional vector described in Sec. 3.2.

After choosing the arm postures that will correspond to the various musical notes, it was necessary to find a mapping between these postures and the notes. As a constraint, we desired that successive, pitch-wise, notes correspond to arm postures that are spatially close enough to be immediately accessible from each other. We faced this as a discrete optimization problem, and therefore used an evolutionary algorithm in order to find the optimal mapping. The algorithm works as follows:

At the first iteration, 50 random mappings between arm postures and musical notes are produced. Then, for the next 50 iterations:

- The mappings are evaluated through the cost function in Eq. 13, which penalizes large sonic differences in close spatially arm postures:

$$C = \sum_{i=1}^{12} \sum_{j=i+1}^{12} \frac{\text{d\_sonic}(i,j)}{\text{d\_spatial}(i,j)}, \qquad (13)$$

  where i, j are arm posture indices, the sonic distance is expressed in number of semitones and the spatial distance is the Euclidean distance between the two arm postures.

- After the evaluation, we create the next generation of mappings using:
  - The 4 best mappings of the previous generation unchanged.
  - For each of them, 9 permutations of the mapping, each produced by swapping two arm postures of its "parent" mapping.
  - Finally, 10 new, random mappings.

Pose Description (Left Arm, Right Arm)	Note
(Down, Down)	C5
(Down, Front)	C5#
(Down, Stretched)	D5
(Stretched, Stretched)	D5#
(Stretched, Down)	E5
(Front, Stretched)	F5
(Diagonal Stretch, Diagonal Stretch)	F5#
(Front, Front)	G5
(Up, Front)	G5#
(Diagonal Up, Diagonal Up)	A5
(Up, Up)	A5#
(Up, Down)	B5

Table 4. The mapping between arm postures and musical notes.

After all 50 iterations, we choose the overall best scoring mapping as the final one. That mapping between arm postures and notes is presented in Table 4.

4. REAL-TIME EVALUATION AND USABILITY TESTING

Our system was subjected to both quantitative and qualitative evaluation. The purpose of the first set of experimental evaluation was to validate its real-time functionality, while the second set was geared towards measuring the quality of the user experience (QoE).

4.1 Quantitative Evaluation

4.1.1 Arm Posture and Gesture Classification

The first quantitative experiment was carried out to evaluate the performance of the arm posture classifier described in Sec. 3.4, as well as the continuous gesture classifier presented in Sec. 3.3, in real time. Both these cases share a similar experimental protocol. In particular, all possible arm posture and continuous gesture instances were recreated in front of a Kinect sensor. In the case of arm postures, each of them was executed 5 times, with the execution duration being equal to 5 seconds. Since the postures were evaluated in a per-second basis, a total of 300 instances were classified. It is worth noting that, out of these, only 2 were misclassified, resulting in an accuracy of 99.33%.

In the case of the continuous gestures, each of the five gestures was executed 8 times. The results of this classification experiment are presented in Table 5, in the form of a confusion matrix. We note that, out of the 40 performed instances, only 38 were successfully recorded, due to failure of the tracking system, and out of these, 3 were misclassified, resulting in an accuracy of 92.11%.

4.1.2 Arm Posture - Note Mapping Evaluation

Finally, it is desired that any two sonically close arm postures are directly accessible from each other. The testing of this condition was not carried out in real time to ensure independence between the testing results and any other possible malfunctions. To this end, we used Matlab in order to simulate all possible spatial transitions through arm postures that correspond to a sonic distance of one, two, and three semitones. Between each such pair of arm postures, we interpolated 50 intermediate postures and classified them. Since we want direct accessibility between sonically close arm postures, we recorded as misclassifications the cases, where the intermediate posture was classified as a posture different to its starting and its final. We present the results of this experiment in Table 6. The results are satisfying, indicating that the performers should be able to directly transition between sonically close arm postures in the majority of the cases.

	G1	G3	G5	G9	G11
G1	8	-	-	-	-
G3	-	6	-	-	-
G5	1	-	7	-	-
G9	-	-	-	8	-
G11	-	-	2	-	6

Table 5. Confusion matrix - where the rows correspond to the ground truth and the columns to the classifier's output - for the real-time continuous gesture classification experiment. A temporary failure of the tracking system resulted in only 6 takes of the gesture G3 being recorded.

Dist.	# of Frames	# of "Mistakes"	Acc.
1	551	17	96.91 %
2	501	35	93.01 %
3	451	46	89.80 %
Total	1503	98	**93.48** %

Table 6. Number of "misclassifications" of intermediate arm postures, respective to the sonic distance (in semitones) between the initial and final postures.

4.2 Qualitative Evaluation

A variety of methods have been proposed in order to measure the quality of the interaction between the system and the performers in interactive performance systems in a quantifiable way. In many cases, as in [12] and [27], some subscales of the Intrinsic Motivation Inventory (IMI) [28], which was created as a means of assessing and evaluating the experience of users in participatory laboratory experiments, are used. Other approaches, such as the one described in [14], are based on a transparency metric for both the performers and the audience, used as a measure of the expressivity of the performance [29].

4.2.1 Methodology

In our case, our system's interactive and sonic environment was evaluated by testing the system on a number of performers. A total of 9 users took part on this preliminary qualitative study, the majority of whom were undergraduate or PhD students at the National Technical University of Athens. Three of them had knowledge of playing a musical instrument, while the majority had some prior experience in using gesturally controlled systems.

First, a brief explanation of how the system works was provided to the users. Then, each of them tried out the sys-

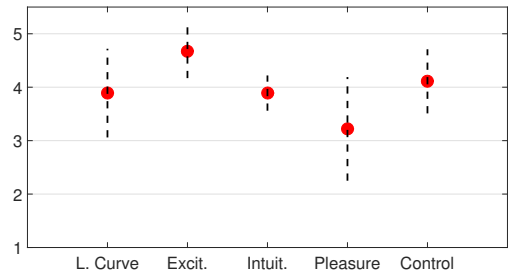

Figure 10. Visualization, in a five-point Likert scale, of the means and standard deviations for the scores of the system's learning curve, user excitement, system's intuitiveness, user pleasantness about the sonic output, and user overall system control, as derived from the questionnaire given to the users.

tem, taking both possible performer roles. After testing the system for a reasonable amount of time, the users answered a questionnaire, designed to gauge their opinion about the various features of the system, as well as its usability, in a five-point Likert scale. Additionally, the questionnaire aimed to determine the degree of collaboration between the users, as well as their preferred system mode. Qualitative comments about possible new features, improvements or alterations on existing ones were also provided in written form.

4.2.2 Qualitative Results and Discussion

The vast majority of the users showed enthusiasm and excitement while they were testing the system. In addition, the majority of the users were pleased by the control they had over the system, being able, after some exploration, to produce the melodies and sound effects they had in mind. The intuitiveness of the system was also highly rated, while, regarding its learning curve, the system was described as relatively easy to learn and play with, at least after some training with the gestures. The sonic output of the system was less well received, being overly reminiscent to "electronic music" to some of the users. A mean-standard deviation plot of these results is shown in Fig. 10.

The comments by the users about issues not covered in the questionnaire were of particular interest. In particular, some users commented that the time interval between different notes could have been controlled more directly, with the notes changing in sync with the arm postures instead of using a predetermined time interval. Another frequent comment involved the potential control of some musical parameters by more fine-grained movements, with one user pointing that "it felt somewhat unnatural to play music out of the range of a typical musical instrument". It is also worthy to note that the users felt that they were collaborating at composing the music, albeit not in the degree we would have wanted. Finally, a technical request involved the robustness of the system in the case of performer occlusions, since the system has specific requirements about performer placement.

Finally, regarding the comparison between the various modes our system offers, the majority of the users showed a preference to mode 3. This is to be expected, as the second mode involved perhaps a more intuitive, but harder to accurately control frequency and amplitude mapping, while the comparison between the roles of the second player in modes 1 and 3 favors the second.

5. CONCLUSIONS AND FUTURE WORK

This paper describes our work in using motion tracking technology to develop a system that composes music via movement. The results are generally encouraging; both the arm postures, hand positions and continuous gestures used as event triggers are easily detected and identified, and the people who tested the system were generally pleased with the result. The responses to the preliminary qualitative evaluation highlighted a number of issues that we would like to include in our future work. More specifically, we aim to improve the degree of collaboration between the two players. Also, a visual assistant, where the players could see skeleton representations of themselves mirroring their movements in real time and indicating their current contribution to the sound, could be helpful. In addition, the actual sonic output could be altered to something less intrusive than sine waves, such as the sound of an actual musical instrument. Last but not least, we are eager to explore more possible mappings, based on both full-body and gestural motion, and explore the potential of this idea in educational applications.

Acknowledgments

This work is partially supported by the iMuSciCA project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731861.
The authors would like to thank everyone who participated in the evaluation of the system for their time and their valuable comments.

6. REFERENCES

[1] R. I. Godoy and M. Leman, *Musical Gestures: Sound, Movement, and Meaning*. New York: Routledge, 2010.

[2] T. Winkler, "Making motion musical: Gesture mapping strategies for interactive computer music." in *Proc. Int. Computer Music Conf.*, Banff, 1995, pp. 261–264.

[3] https://developer.microsoft.com/en-us/windows/kinect.

[4] https://msdn.microsoft.com/en-us/library/microsoft.kinect.jointtype.aspx.

[5] M.-H. Hsu, W. Kumara, T. K. Shih, and Z. Cheng, "Spider king: Virtual musical instruments based on microsoft kinect." in *Proc. Int. Joint Conf. on Awareness Science and Technology And Ubi-Media Computing*, Aizuwakamatsu, 2013, pp. 707–713.

[6] R. Masu, A. Conci, C. Core, A. de Angeli, and F. Morreale, "Robinflock: a polyphonic algorithmic composer for interactive scenarios with children." in *Proc. Sound and Music Computing Conf.*, Espoo, 2017, pp. 77–84.

[7] L. Truchet, "The hyper-zampogna." in *Proc. Sound and Music Computing Conf.*, Hamburg, 2016, pp. 485–490.

[8] A. M. Burns, S. Bel, and C. Traube, "Learning to play the guitar at the age of interactive and collaborative technologies." in *Proc.Sound and Music Computing Conf.*, Espoo, 2017, pp. 77–84.

[9] A. Rosa-Pujazon, I. Barbancho, , L. J. Tardon, and A. M. Barbancho, "Conducting a virtual ensemble with a kinect device." in *Proc. Sound and Music Computing Conf.*, Stockholm, 2013, pp. 284–291.

[10] A. Rosa-Pujazon, I. Barbancho, L. J. Tardon, and A. M. Barbancho, "A virtual reality drumkit simulator system with a kinect device." in *Int. Journal of Creative Interfaces and Computer Graphics, Vol.6, No. 1*, 2015, pp. 72–86.

[11] M.-J. Yoo, J.-W. Beak, and I.-K. Lee, "Creating musical expression using kinect." in *Proc. Int. Conf.on New Interfaces for Musical Expression*, Oslo, 2011, pp. 324–325.

[12] A. N. Antle, M. Droumeva, and G. Corness, "Playing with the sound maker: do embodied metaphors help children learn?" in *Proc. Interaction design and children Int. Conf.*, Chicago, 2008, pp. 178–185.

[13] T. Grill, "Two-dimensional gesture mapping for interactive real-time electronic music instruments," *Master thesis, University of Music And Performing Arts, Vienna*, 2008.

[14] D. Brown, C. Nash, and T. Mitchell, "Gesturechords: Transparency in gesturally controlled digital musical instruments through iconicity and conceptual metaphor," in *Proc. Sound and Music Computing Conf.*, Hamburg, 2016, pp. 491–497.

[15] J. Carreira and P. Peixoto, "A vision based interface for local collaborative music synthesis." in *Proc. Int. Conf. on Automatic Face and Gesture Recognition*, Southampton, 2006, pp. 591–596.

[16] S. Mitra and T. Acharya, "Gesture recognition: A survey," in *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), Vol. 37, No. 3*, 2007, pp. 311–324.

[17] M. Reyes, G. Dominguez, and S. Escalera, "Featureweighting in dynamic timewarping for gesture recognition in depth data," in *Proc. IEEE Int. Conf. on Computer Vision Workshops*, Barcelona, 2011, pp. 1182–1188.

[18] S. Celebi, A. S. Aydin, T. T. Temiz, and T. Arici, "Gesture recognition using skeleton data with weighted dynamic time warping." in *Proc. Int. Conf. on Computer Vision Theory And Applications*, Barcelona, 2013, pp. 620–625.

[19] X. Zhao, X. Li, C. Pang, X. Zhu, and Q. Z. Sheng, "Online human gesture recognition from motion data streams," in *Proc. ACM Int. Conf. on Multimedia*, Barcelona, 2013, pp. 23–32.

[20] M. Raptis, D. Kirovski, and H. Hoppe, "Real-time classification of dance gestures from skeleton animation." in *Proc. ACM SIGGRAPH/Eurographics symposium on computer animation*, Vancouver, 2011, pp. 147–156.

[21] G. T. Papadopoulos, A. Axenopoulos, and P. Daras, "Real-time skeleton-tracking-based human action recognition using kinect data." in *Proc. Int. Conf. on Multimedia Modeling*, Dublin, 2014, pp. 473–483.

[22] F. Negin, F. Ozdemir, C. B. Akgul, K. A. Yuksel, and A. Ercil, "A decision forest based feature selection framework for action recognition from rgb-depth cameras," in *Proc. Int. Conf. on Image Analysis and Recognition*, Varzim, 2013, pp. 648–657.

[23] M. E. Hussein, M. Torki, M. A. Gowayyed, and M. El-Saban, "Human action recognition using a temporal hierarchy of covariance descriptors on 3d joint locations." in *Proc. Int. Joint Conf. on Artificial Intelligence*, Beijing, 2013, pp. 2466–2472.

[24] https://www.portaudio.com/.

[25] S. Fothergill, H. Mentis, P. Kohli, and S. Nowozin, "Instructing people for training gestural interactive systems." in *Proc. SIGCHI Conf. on Human Factors in Computing Systems*, Austin, 2012, pp. 1737–1746.

[26] D. Fleet and Y. Weiss, "Optical flow estimation," in *Handbook of mathematical models in computer vision*. Springer, 2006, pp. 237–257.

[27] M. Lionello, M. Mandanici, S. Canazza, and E. Micheloni, "Interactive soundscapes: Developing a physical space augmented through dynamic sound rendering and granular synthesis," in *Proc. Sound and Music Computing Conf.*, Espoo, 2017, pp. 77–84.

[28] R. Ryan and E. Deci, "Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being," in *American Psychologist, Vol.55, No. 1*, 2000, pp. 68–78.

[29] S. Fels, A. Gadd, and A. Mulder, "Mapping transparency through metaphor: towards more expressive musical instruments," in *Organised Sound, Vol. 7, No. 2*, 2002, pp. 109–126.