Expressive Control of Indirect Augmented Reality During Live Music Performances

Lode Hoste and Beat Signer
Web & Information Systems Engineering Lab
Vrije Universiteit Brussel
Pleinlaan 2, 1050 Brussels, Belgium
{lhoste,bsigner}@vub.ac.be

ABSTRACT

Nowadays many music artists rely on visualisations and light shows to enhance and augment their live performances. However, the visualisation and triggering of lights in popular music concerts is normally scripted in advance and synchronised with the music, limiting the artist's freedom for improvisation, expression and ad-hoc adaptation of their show. We argue that these limitations can be overcome by combining emerging non-invasive tracking technologies with an advanced gesture recognition engine.

We present a solution that uses explicit gestures and implicit dance moves to control the visual augmentation of a live music performance. We further illustrate how our framework overcomes limitations of existing gesture classification systems by providing a precise recognition solution based on a single gesture sample in combination with expert knowledge. The presented approach enables more dynamic and spontaneous performances and—in combination with indirect augmented reality—leads to a more intense interaction between artist and audience.

Keywords

Expressive control, augmented reality, live music performance, 3D gesture recognition, Kinect, declarative language

1. INTRODUCTION

Today's musical performances often cover more than simply the auditive aspect and consist of complex choreographies, impressive visualisations on large screens or advanced light shows. Due to the complexity of such concerts, the augmenting functionality often consists of scripted behaviour that is synchronised with the performance. Choreographies can be kept flexible since humans can spontaneously agree to modifications, which is not the case for pre-programmed visualisations. The custom made visualisations are scripted in advance and have to precisely match to an agreed timing of the music. Therefore, artists are restricted by the predefined synchronisation between sound and images or light shows and they have limited or no means to modify and adapt the behaviour and augmentation during a live performance. We argue that through the use of non-intrusive predefined gestural expressions, artists become more flexible in adapting and spontaneously modifying the augmenting visualisation.

NIME'13, May 27 – 30, 2013, KAIST, Daejeon, Korea.

In order to enable expressive control during live music performances, non-invasive technologies such as depth cameras (e.g. Microsoft's Kinect[1]) can be used in combination with an advanced gesture recognition engine that allows the precise definition of multiple explicit and implicit gestures. In this context, gestures are defined as a sequence of body movements in time. It is important to stress that there is a crucial need for both high precision and recall in order to support gesture-based augmented live performances. While gesture actions should not be accidentally triggered, the recognition engine should recognise any gesture performed by the artist. The problem of translating a large amount of camera data into meaningful high-level gesture events has been a subject of research for many years. However, with today's available technologies it becomes possible to build an affordable gesture recognition solution which performs very well for the described scenario.

In collaboration with the music band Mental Nomad[2], we experimented with different forms of visualisations and the corresponding activation triggers, including virtual reality as well as indirect augmented reality in order to enrich parts of their live performance. The solution presented in this paper has been successfully used in multiple live concerts. The overall goal was to achieve a tighter coupling between the live music performance and the response of the audience by removing the scripted behaviour of predefined visualisations. We start by comparing our work with current state of the art solutions and then present a number of important constraints our system had to deal with. We proceed with a discussion of the implementation of our system and outline how it has been perceived as well as how it might be reused in other settings. We conclude with a discussion of some future work which will enable artists to use our system without the support of an expert developer.

2. BACKGROUND

The development of real-time gestural controls for musical performances has been an active area of development for a number of years. As Dobian [4] argues, expressiveness can be enhanced by using an intelligent recognition of gestures in combination with the corresponding mapping to actions.

Most existing toolkits for real-time gestural control focus on continuous processes such as dynamic sound synthesis [10, 6, 3] and visualisations [1, 9], rather than investigating the recognition of discrete triggers. This is mainly due to the fact that the continuous nature of sounds and visuals nicely maps to continuous input modalities. In addition, the robust recognition of an artist's complex movements in order to use them as discrete triggers is a challenging task. Another difficulty is the unforgiving nature of wrongly de-

[1] `http://www.microsoft.com/en-us/kinectforwindows/`
[2] `http://www.mentalnomad.be`

tected moves and gestures if they are used to trigger and switch between different visualisations.

Bevilacqua [3] generalises the mapping between sensors and processes into one-to-one, one-to-many, many-to-one and many-to-many mappings. Their MnM mapping toolbox supports the fusion of different input modalities into one or multiple output parameters. Additional methods for interpolation, regression and recognition methods allow for dynamic sound synthesis or modifications (e.g. pitch, reverb or volume) based on the artist's movements. Similarly, Digito provides support for the expressive control of a virtual musical instrument that can be played through hand gestures [6]. Furthermore, the performer can dynamically switch between two sound synthesis options via a subtle gesture formed by tapping at various locations in space with the index finger. Other movements were continuously mapped to audio synthesis. In Digito, a Microsoft Kinect sensor was used to capture all the necessary information. A recent evaluation of the accuracy and performance of the Microsoft Kinect sensor revealed that the sensor is amongst the least accurate trackers [12]. However, it is also noted that it is the simplest system to set up due to the fact that it does not require any calibration process while being resistant to changes in lighting conditions. Note that this resistance to changes in lighting is crucial for most live music performances.

Mitchel et al. [10] present a number of discrete control mechanisms for hand-based input, including posture identification, segmented orientation and inertial peak detection. As mentioned before, these controls remain quite basic and do not entail any higher level comprehension of the gestural interaction. Barry et al. [1] present simple interpretation rules to augment a Butoh dance performance. They use contiguous sequences to generate dynamic visualisations. These visualisations are built from a hidden grammar based on motions for emotion recognition. Instead of using body movements to create audio, an implicit mapping is created for visual enhancement. However, the visualisations were not based on an explicit triggering for predefined gestures.

Dynamic Time Warping (DTW) is a popular algorithm to recognise complex patterns. Real-time extensions based on a multigrid [2] allow the DTW classifier to work on a continuous data stream, such as camera input. Note that DTW can only process a single trajectory and finding patterns between multiple joints for full body gesture recognition requires complex preprocessing methods. The setting of a global (or gesture-specific) threshold to decide whether to trigger a gesture or not is too unreliable and easily results in false positives or false negatives. Due to the lack of training data, learning-based methods including hidden Markov models (HMM) [11] are not appropriate for our purpose.

3. CONSTRAINTS

Our goal was to overcome the limited interactiveness and possibility for spontaneous adaptations and changes in live performances. The artists should be given more possibilities to interact with and influence the audience without the fixed scripted behaviour of visualisations. We argue that through the use of innovative non-invasive technologies, such as depth sensors, artists have a mean to perform expressive gesture control by using their body. We explore what properties a reliable gesture recogniser needs to fulfil in order to enable such expressive control of visualisations and indirect augmented reality during a live music performance.

It is not trivial to extract meaningful information from a continuous stream of full body tracking information by performing real-time pattern recognition. The input stream contains a lot of non-relevant movements and the recogniser has to process all this information in real time. Additionally, we did not have access to all the dance moves due to the minimal time budget for this part of the performance. This constraint is problematic for many learning-based gesture recognition approaches since the system needs to "learn" both gestures as well as non-gestures. In the discussed setup where explicit gestures and implicit dance moves are used to control the augmentation process, non-gestural movements form the norm rather than the exception. The artists requested us to trigger specific visualisations when a number of key moves are performed. Since a single song takes about four minutes and multiple artists are dancing in a more or less controlled sequence, the key movements take up less than 5% of the overall time. Furthermore, the sequence was not known and variations are common due to the influence of the audience.

To summarise, the presented scenario resulted in the following constraints:

One-shot gesture sampling Due to time constraints, we were unable to perform an iterative evaluation of the system together with the artists. Hence, single samples of the five key gestures were recorded and no *garbage data* was available to train a classifier with negative examples.

No garbage data The artist's choreography is scripted in a flexible manner. The robustness in the case of unscripted and uncommon moves was a requirement put up front by the artists. The lack of full body data for the entire choreography complicates the creation of an idle state and the evaluation phase.

High precision The use of expressive control for a couple of key movements asks for a high precision. The system should not trigger its actions unintentionally as this would break the flow of the live performance.

High recall On the other hand, a high recall should be obtained in order to ensure that the actions are triggered when a key movement is performed. Imagine an artist performing a special jump and no visual feedback is delivered. The required nearly perfect precision and recall complicates the task and due to the very limited recorded data, the use of expert knowledge and reasoning over a larger time period seems appropriate.

Real-time processing Any activation that is based on expressive movements should happen nearly instantaneously and we should be able to process the data as it enters the system.

Non-invasive sensor technology It was requested that the technology should be non-invasive. The embedding of sensors in clothes was not an option due to the indoor scene which implies lightweight shirts and sweating which might negatively affect the sensors. We requested that the movements should be executed in an area of $5m^2$. This allows a single Microsoft Kinect sensor to easily track the artist performing the moves.

Multiple users In our scenario, four out of five artists were moving on stage but requested to be ignored in the dedicated camera area. As it was risky to perform the recognition process on the first artist entering the dedicated area at the beginning of the song, we decided to enable a multi-user gesture recognition process that allows multiple artists to trigger the actions.

4. EXPRESSIVE CONTROL

We present several guidelines on how an augmented reality system can be implemented, specifically paying attention to constraints introduced in the previous section. There are two main parts of the application: the input side with the corresponding real-time input stream processing and the output side which takes care of the visualisation. The initial idea for the output modality was to apply a virtual reality setting where an avatar is directly controlled by one of the artists and the key gestures trigger additional visualisation elements such as fire or electricity. However, this has relatively quickly proven to be infeasible due to the fact that many moves, such as a 360 degree rotation, cannot be tracked very accurately by the Microsoft Kinect SDK[3]. Furthermore, existing avatar models did not fit well with the overall visualisation concept. An alternative solution is indirect augmented reality, where a live video capture of the artist is overlaid with certain visualisation elements which are triggered by some key gestures. This allows us to deal with some of the inaccuracies of the Microsoft Kinect SDK without noticeable visual artefacts. In Section 5 we provide further details about the implementation and the different libraries that we used.

Expressive control through non-invasive technologies creates the opportunity to enable specific visualisations as commanded by the artist. The five gestures G1 to G5 which were used to trigger the indirect augmented reality are highlighted in Figure 1 to Figure 5 as a sequence of postures. Dealing with multiple postures over time is crucial when only a very few samples are available. The reason for this is to optimise the precision as gestures should not be unintentionally triggered. Without time information, the artist might trigger several end postures while dancing, taunting the audience or even trigger the change in augmentation if the Microsoft Kinect SDK incorrectly tracks the user. Additionally, when incorporating full body gesture recognition, we do not prohibit the artist from executing other similar moves. By defining a precise body movement sequence to which the user must adhere in order to trigger the intended actions, we can optimise the system's recall and precision.

It should be noted that the interpretation of full body movements over time requires advanced software features. As mentioned earlier, existing work focuses on manipulating input to direct output with little reasoning. To achieve the necessary precision in gesture recognition, we extended the Mudra framework [7] for three-dimensional gesture recognition based on the idea of declarative control points as presented by Hoste et al. [8]. In contrast to the gesture spotting approach described by Hoste et al. [8], which introduced automated control point inferencing for two-dimensional gestures, we extended the approach with three-dimensional ellipsoids as the basic form for the definition of control points. This generic solution generates declarative code from a single sample and allows expert users to further refine the gesture definition to achieve the necessary precision requested by the artist.

In contrast to statistical or template-based solutions, Mudra provides programming constructs to enable developers to express complex patterns in space and time. These programmed patterns are defined based on a declarative rule language which offers two main features: First, developers can focus on *what* they want to recognise rather than *how* it should be implemented. In addition, these rules can be compiled into a direct acyclic graph network (Rete [5]) which allows for real-time performance.

[3]Microsoft Kinect SDK: `http://www.microsoft.com/en-us/kinectforwindows/develop/overview.aspx`

Figure 1: Gesture G1

Figure 2: Gesture G2

Figure 3: Gesture G3

Figure 4: Gesture G4

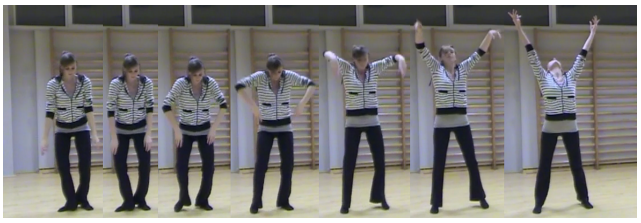

Figure 5: Gesture G5

Listings 1 and 2 show a partial implementation of gesture G1 which has been introduced earlier in Figure 1. A couple of control points per posture of a gesture and relationships of joints relatively to other joints in space, are defined in Listing 1. The code consists of a conditional element (lines 2 and 3) and spatio-temporal operators (lines 4 to 7). The conditional elements require that *there should be a relative joint from the torso (parent) to the left foot (child) that meets a number of conditions.* One of these conditions is defined on line 4 via a spatial operator specifying that the relative joint should match a three-dimensional ellipsoid with a specific size at a certain location. Lines 1 to 7 are considered to be conditional code, while the lines 9 and 10 (after the ⇒ symbol) contain activation code which will be executed whenever a condition is satisfied. Several spatio-temporal operators are provided as built-in functionality and expert developers can extend the system with their own operators. An expert further adjusts the gesture definition

according to specialisation or flexibility requirements. For each control point (ellipsoid) such a definition has to be created, whereby user-defined operators can help in dealing with distance, angles or other spatio-temporal properties.

Listing 1: Control points for gesture G1

```
1  (defrule ControlPoint3LeftFoot0
2    ?p1 ← (RelativeJoint (parent ?∗torso∗)
3                         (child ?∗foot_left∗))
4    (test (spatial:matchEllipsoid ?p1
5        -0.14259260892868 -1.68670213222504
6        0.137266874313354 0.6216448 1.506405
7        0.6314279 0 0 6.189708))
8  =>
9    (assert (Ellipsoid (name "g3FootL0")
10       (time ?p1.time))))
11
12 (defrule ControlPoint1RightHand0
13   ?p1 ← (RelativeJoint (parent ?∗shoulder_right∗)
14                        (child ?∗hand_right∗))
15   (test (spatial:matchEllipsoid ?p1
16       0.152908384799957 -1.12613391876221
17       0.0352647304534912 0.4534679 1.695701
18       0.3887017 0 0 0.1501325))
19 =>
20   (assert (Ellipsoid (name "g3HandR1")
21       (time ?p1.time))))
22 ...
```

In the definition of gesture G1 shown in Listing 2, control points are combined to form a complex gesture. Notice how all feet (torso-foot relative joint) and arms (shoulder-hand relative joint) are important in this movement (lines 2, 5, 6 and 7), even though the left arm remains steady (Figure 1). For this gesture, the steady arm contains a lot of valuable information as we optimise for a precise gesture definition to give the artist the freedom to perform other, but similar movements that should not activate the gesture definition. Lines 7 to 11 describe the movement of the arm in three phases which could even be extended if higher precision is required.

Listing 2: Gesture G1: pointing up

```
1  (defrule Gesture1
2    ?p1 ← (Ellipsoid (name "g3FootL0") (user ?u))
3    ?p2 ← (Ellipsoid (name "g3FootL4") (user ?u))
4    (test (time:before ?p1 ?p2 3.s))
5    ?p3 ← (Ellipsoid (name "g3FootR5") (user ?u))
6    ?p4 ← (Ellipsoid (name "g3HandL7") (user ?u))
7    ?p5 ← (Ellipsoid (name "g3HandR1") (user ?u))
8    ?p6 ← (Ellipsoid (name "g3HandR2") (user ?u))
9    (test (time:before ?p5 ?p6 3.s))
10   ?p7 ← (Ellipsoid (name "g3HandR3") (user ?u))
11   (test (time:before ?p6 ?p7 1.s))
12 =>
13   (assert (GestureMatch (name "Gesture1")
14       (startTime ?p1) (stopTime ?p7) (user ?u))))
```

By using a declarative language, we can easily manually refine various details without having to capture additional sample data. For instance, the angle of the arm in the end move could be less strict and the movement of the left leg can be used to optimise the precision. Additional control points can be added to make the gesture sequence more strict, including those based on other joints can be incorporated to refine the definition. The further we go back in time, the more precisely a gesture can be defined. However, this forces the user to perform the gesture as agreed in the gesture definition.

Furthermore, the declarative definition of gestures based on the Mudra framework allows for an easy explanation to the artist on how a gesture is implemented and to what constraints their movements have to adhere. We do neither require extensive training data to implement new gestures, nor other non-gesture data. The Mudra recognition engine is implemented in the C programming language and is fast enough to process continuous full body movement of multiple users over a period of time. Furthermore, it allows precise detection (i.e. reducing false positives) and also results in a high recall. In this application, gestures occur in a fixed sequence which means that the activation of gestures can be further refined by adding a previous gesture activation as a conditional element.

In a declarative language, unification can be used to group certain entities. In this case, by using a single logical variable for conditional elements on the user field, we automatically enable support for multiple users. The user identifier that triggers the gesture will be passed to the final application (line 14), such that the correct user will be augmented with the appropriate visual elements.

Barry et al. [1] mention the *"trade-off between recognition quality and the delay of the real-time recognition"* as one element of future work. By using a declarative gesture spotting language, gestures can be recognised precisely and in real time. It does not require a preprocessing step that splits up the continuous stream into possible gesture candidates, but rather computes the incremental gesture by using the efficient Rete algorithm.

We conclude this section by shortly revisiting how the constraints of the expressive control are met. Our three-dimensional ellipsoid Mudra extensions allow us to create expert gesture definitions instead of statistical or template-based gesture recognition solutions. This overcomes two major obstacles: the lack of gesture and non-gesture data as well as the fact that experts can manually configure the precision and recall for all gestures by defining control points and parameterising spatio-temporal functions. The Mudra engine compiles these definitions into a Rete engine which maintains incremental results through a time sliding window, enabling the real-time processing of a continuous input stream. The input stream consists of full body tracking information (30 Hz and 20 joints) originating from a non-invasive Microsoft Kinect sensor. By relying on a declarative language and unification, the support for multiple users at the same time only requires a single additional variable. This is in huge contrast to imperative languages where intermediate state management of complex full body gestures is very hard to implement and modify.

5. IMPLEMENTATION

The overall architecture of our system is outlined in Figure 6. The input layer consists of the Microsoft Kinect sensor, which is responsible to capture the full body movements of users, and the Mudra recognition engine, configured with the corresponding declarative gesture rules. The Kinect SDK and the Mudra framework are interconnected through the Open Sound Control (OSC)[4,5,6] protocol. Our OSC components for the Microsoft Kinect SDK[7] or OpenNI-NITE[8] have been made available under an open source license. Note that the OSC protocol acts as a decoupling layer between the software architecture and the physical space. It enables us to use a dedicated laptop for user

[4] http://opensoundcontrol.org
[5] http://liblo.sourceforge.net
[6] http://www.ventuz.com/support/help/v3_01/OpenSoundControl.html
[7] https://github.com/Zillode/OSCeleton-KinectSDK
[8] https://github.com/Zillode/OSCeleton-OpenNI

tracking based on the Microsoft Kinect SDK in the front of the stage and to transmit the data through Ethernet to another laptop positioned backstage.

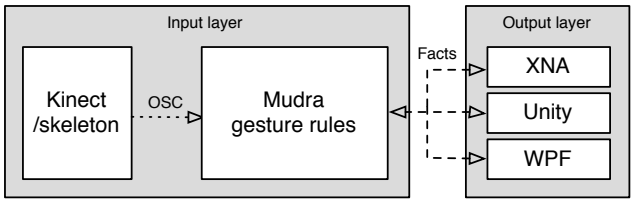

Figure 6: Architecture

Mudra processes the continuous OSC stream and triggers the programmed assertions whenever the artist performs the defined gestures. These assertions are transported to the output layer through two-way TCP/IP key-value pairs as communication protocol. These pairs contain simple values (i.e. integers, floats and strings) such as the name of the gesture and the time it has been recognised. The values are simple enough to be supported by various programming languages. We experimented with multiple technologies to create a virtual reality environment based on the Unity3D framework where an avatar is controlled by the artist. The predefined gestures then triggered various flames that were attached to one or multiple joints. However, as mentioned before, a direct mapping between the artist and a virtual avatar resulted in noticeable artefacts when the artist performs a move that was not correctly tracked by the Microsoft Kinect SDK. We switched to the Windows Presentation Foundation (WPF)[9] for implementing the augmented reality part. Our augmented reality solution is more forgiving for incorrect tracking and jittering of joint coordinates. Unfortunately, the WPF implementation is unreliable when running for several hours due to internal memory leaks and also results in relatively poor performance when a lot of particles in the simulation of the flames are used. Nevertheless, these particles are crucial to form a realistic visualisation of fire and other effects. Finally, Microsoft XNA[10] was chosen as an ideal platform for our solution as it is a lower level library providing more control over the performance variables. Inside the XNA output module, particle effects are used to the create the fire animation. This fire animation can be attached to the location of a user's skeleton joints. Since the performed song is about fire, we gradually enable more fire over time and every time the artist performs a specific gesture an additional joint is put on fire. This allows the artist to decide when they want to enable additional visualisations layers. In particular, when gesture G1 is performed, the right hand is set on fire and the fire keeps burning even when the artist moves further, creating a visually appealing fire trail as, for example, shown in Figure 7. Whenever gesture G2 is performed, both hands will be on fire. The same applies to gestures G3 and G4 to activate the fire for the feet. Finally, when gesture G5 is detected, the whole body is set on fire as shown in Figure 9.

The decoupling between input and output layers through TCP/IP further allows us to experiment with other types of activations such as enabling different types of visualisations or triggering light programmes in the future. Note that activations could also be triggered by attentive staff, but for many artists this is not an option for their performances due to budget restrictions. It should also be stressed that this approach is tailored towards precise gesture recognition. This is in contrast to many related approaches that generate direct feedback in terms of sound synthesis or dynamic visualisations. However, the system can also serve as an additional module for these approaches, for instance to switch between scenes or enabling more complex behaviour than a direct mapping between input and output.

The presented system provides an ideal solution for scenarios where precise gesture recognition is required to activate certain actions. Currently, the declarative gesture rules are composed by an expert user. However, in the near future we plan to provide an integrated development environment (IDE) that is suitable to create complex three-dimensional full body gestures by end users. The IDE will allow the graphical coordination of control points which are transformed into readable declarative rules that subsequently still can be modified by the expert user.

6. DISCUSSION AND CONCLUSION

The use of our gesture-based indirect augmented reality during three live performances of the band Mental Nomad is shown in Figures 7, 8 and 9. We are happy that the artists were extremely satisfied with the accuracy of the system and the added value it produced in terms of an excited audience and more flexibility for improvisation. Indeed, the reception from the audience was intense, partly due to the augmented reality stream, but mainly also due to the matching visual overlays that gradually increased during the song. We also received comments from the audience that the visualisations were extremely accurate and synchronised with the performed choreography.

Figure 7: Live performance (after gesture G3)

Figure 8: Live performance (gesture G4)

The recognition engine performed flawlessly without any false positives or missed gestures during all three live performances that took place in 2012 with an audience of about 1500 people. The mix between continuous augmented reality and discrete activations of the visualisation layers provides an interesting view on the two large screens that have been installed at the sides of the stage as illustrated in Figure 10. The Kinect sensor was positioned at the front of the runway and the optimal tracking area was labelled on

[9] http://msdn.microsoft.com/en/library/ms754130.aspx

[10] http://msdn.microsoft.com/en-us/centrum-xna.aspx

Figure 9: Live performance (gesture G5)

the ground as an aid for the artists. All Kinect skeleton data was transmitted trough OSC to a backstage laptop connected to the two large screens.

The live music performance was recorded and our paper is accompanied by parts of the video material showcasing a small part of the performance. Currently, our system was embedded in a larger album presentation and was only used to augment a single song. However, based on the good performance and the positive feedback from both the artists as well as the audience, we hope that these kind of systems become more ubiquitous in future performances. One limitation of our approach entailed by the Microsoft Kinect sensor and SDK, is the limited skeleton tracking accuracy and the inability to function under certain conditions (e.g. direct sunlight). However, more advanced depth sensors and tracking algorithms could be used in other settings.

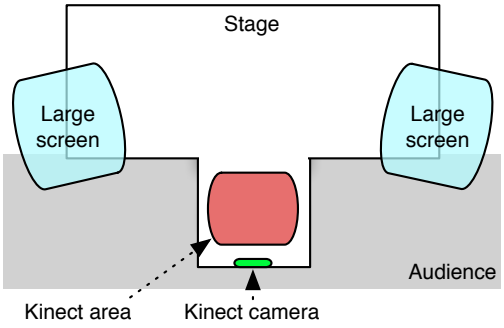

Figure 10: Stage configuration

We demonstrated an application of our general gesture recognition engine. The presented solution allows precise gesture recognition without requiring multiple gesture samples. Nevertheless, despite the lack of non-gestural expressions and the inaccurate tracking provided by the Microsoft Kinect SDK, the presented system for expressive control of indirect augmented reality performs very well. Finally, our declarative real-time classification platform can also be used to implement discrete gesture-based triggers in other settings, including the expressive control of light shows or when activating different sound synthesis features. Other future plans include the development of an integrated development environment to simplify the definition of complex gesture by end users.

Acknowledgements

The work of Lode Hoste is funded by an IWT doctoral scholarship. We further thank Toon Duwee for implementing the initial prototypes forming part of this project.

7. REFERENCES

[1] M. Barry, J. Gutknecht, I. Kulka, P. Lukowicz, and T. Stricker. Multimedial Enhancement of a Butoh Dance Performance - Mapping Motion to Emotion with a Wearable Computer System. In *Proceedings of the 2nd International Conference on Advances in Mobile Multimedia (MoMM 2004)*, Bali, Indonesia, 2004.

[2] F. Bettens and T. Todoro. Real-Time DTW-based Gesture Recognition External Object for Max/MSP and PureData. In *Proceedings of the 6th International Conference on Sound and Music Computing (SMC 2009)*, Porto, Portugal, 2009.

[3] F. Bevilacqua, R. Müller, and N. Schnell. MnM: a Max/MSP Mapping Toolbox. In *Proceedings of the 12th International Conference on New Interfaces for Musical Expression (NIME 2012)*, Vancouver, Canada, 2005.

[4] C. Dobrian and D. Koppelman. The 'E' in NIME: Musical Expression with New Computer Interfaces. In *Proceedings of the 6th International Conference on New Interfaces for Musical Expression (NIME 206)*, Paris, France, 2006.

[5] C. L. Forgy. Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *Artificial Intelligence*, 19(1), 1982.

[6] N. Gillian and J. A. Paradiso. Digito: A Fine-Grain Gesturally Controlled Virtual Musical Instrument. In *Proceedings of the 12th International Conference on New Interfaces for Musical Expression (NIME 2012)*, Ann Arbor, USA, 2012.

[7] L. Hoste, B. Dumas, and B. Signer. Mudra: A Unified Multimodal Interaction Framework. In *Proceedings of the 13th International Conference on Multimodal Interaction (ICMI 2011)*, Alicante, Spain, November 2011.

[8] L. Hoste, B. D. Rooms, and B. Signer. Declarative Gesture Spotting Using Inferred and Refined Control Points. In *Proceedings of the International Conference on Pattern Recognition (ICPRAM 2013)*, Barcelona, Spain, February 2013.

[9] H. Ip, Y. Hay, and A. C. Tang. Body-Brush: A Body-driven Interface for Visual Aesthetics. In *Proceedings of the 10th International Conference on Multimedia (Multimedia 2002)*, 2002.

[10] T. Mitchell, S. Madgwick, and I. Heap. Musical Interaction with Hand Posture and Orientation: A Toolbox of Gestural Control Mechanisms. In *Proceedings of the 6th International Conference on New Interfaces for Musical Expression (NIME 2006)*, Paris, France, 2006.

[11] A.-Y. Park and S.-W. Lee. Gesture Spotting in Continuous Whole Body Action Sequences Using Discrete Hidden Markov Models. In *Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation (GW 2005)*, Ile de Berder, France, May 2005.

[12] G. Vigliensoni and M. M. Wanderley. A Quantitative Comparison of Position Trackers for the Development of a Touch-less Musical Interface. In *Proceedings of the 12th International Conference on New Interfaces for Musical Expression (NIME 2012)*, Vancouver, Canada, 2012.