Fluid Gesture Interaction Design: Applications of Continuous Recognition for the Design of Modern Gestural Interfaces

BRUNO ZAMBORLIN, Goldsmiths, University of London and IRCAM STMS-CNRS-UPMC
FREDERIC BEVILACQUA, IRCAM STMS-CNRS-UPMC
MARCO GILLIES and MARK D'INVERNO, Goldsmiths, University of London

This article presents Gesture Interaction DEsigner (GIDE), an innovative application for gesture recognition. Instead of recognizing gestures only after they have been entirely completed, as happens in classic gesture recognition systems, GIDE exploits the full potential of gestural interaction by tracking gestures continuously and synchronously, allowing users to both control the target application moment to moment and also receive immediate and synchronous feedback about system recognition states. By this means, they quickly learn how to interact with the system in order to develop better performances. Furthermore, rather than learning the predefined gestures of others, GIDE allows users to design their own gestures, making interaction more natural and also allowing the applications to be tailored by users' specific needs. We describe our system that demonstrates these new qualities—that combine to provide fluid gesture interaction design—through evaluations with a range of performers and artists.

Categories and Subject Descriptors: H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Input devices and strategies*

General Terms: Algorithms, Design, Experimentation, Human Factors, Deployment of Gesture Interaction Systems

Additional Key Words and Phrases: Gesture interaction, design and application of gesture interaction systems, meaningful feedback, continuous and synchronous control, personalisation

ACM Reference Format:
Bruno Zamborlin, Frederic Bevilacqua, Marco Gillies, and Mark D'inverno. 2014. Fluid gesture interaction design: Applications of continuous recognition for the design of modern gestural interfaces. ACM Trans. Interact. Intell. Syst. 3, 4, Article 22 (January 2014), 30 pages.
DOI: http://dx.doi.org/10.1145/2543921

1. INTRODUCTION

Gestural interaction is starting to fulfil its potential of becoming an essential part of modern interaction design. For example, touch screen mobile devices have made gesture interfaces ubiquitous, whereas motion-based game controllers such as the Microsoft Kinect make full-body gesture interaction affordable and practical. Historically, gestural interface technologies have been developed for use in quite specific domains (including, for example, across artistic and performance domains), but general uptake was limited. More recently, and largely popularised due to the increasing interactive

The reviewing of this article was managed by associate editor Desney 5Tan.
Authors' addresses: Bruno Zamborlin, Marco Gillies, and Mark D'Inverno, Goldsmiths college, Computing Department, 8 Lewisham way, SE146NW London, UK; Bruno Zamborlin and Frederic Bevilacqua, IRCAM, Realtime musical interactions, 1 Place Stravinsky, Paris 75004, France; emails: bruno.zamborlin@gold.ac.uk, bruno.zamborlin@gmail.com.

DOI: http://dx.doi.org/10.1145/2543921

demands of the game industry, gesture interaction is starting to become more widespread. This increasing uptake signifies the huge potential promise of gesture interaction systems to provide the future way in which we will all interact with technology. We are at a stage of research into such systems where the focus needs to move from the novelty of being able to create gestural interfaces towards how we can create *effective* gestural interfaces: interfaces that genuinely go beyond what is possible with the traditional mouse and keyboard devices. In order to frame our article and the context of our research, we first provide a desiderata of what we believe to be crucial aspects for the design of modern gesture interface systems so that they become effective for users and widely adopted. Ideally, we believe that modern gesture interfaces should provide the following four properties:

(1) *Continuous control.* Users should be able to *continuously* and *synchronously* control the target application moment by moment through their gestures. Although for many applications it may be sufficient to trigger discrete events, enabling additional continuous control will extend the range of possible control mechanisms. Expressivity of human body movements cannot be fully represented as a sequence of discrete commands. Continuous synchronisation between user movements and digital processes is necessary to enable expressivity in gestural interfaces. For example, it is generally useful to include the possible modulating effect of continuous changes in gestures occurring between triggering events. This typically allows for the use of important information occurring in preparation gestures, which in turn can be used to anticipate specific control. The standard recognition techniques output results once a given gesture is finished. Using continuous control, it then becomes possible to extend such an approach by extending intermediate recognition results that become available during a gesture performance.
(2) *Tailorable for specific context.* Users should be able to define their own personal gestures (irrespective of their expertise or physical mobility) and adapt the system to the specific application context and across different physical environments in which design and/or performance are taking place. Users should be able to define a *personal vocabulary of gestures* specifically for the *target application* in hand and the *environment* in which interaction activity will take place. Systems must provide users with the flexibility to easily modify their gestures as users develop the way in which they want to interact with the system. The system must enable the user to define gestures that are natural, meaningful, and even *metaphorical* to them personally with respect to the response behaviour of that system. If systems do not allow this, then they risk being worse than traditional GUIs, as users must remember an arbitrary vocabulary of gestures that are not meaningfully grounded in their own individual movements. In such situations, it becomes at least as difficult as remembering an arbitrary set of textual commands and possibly even more difficult if gesture interaction itself is new to the user. We argue that for successful general purpose systems, users must be flexibly accommodated so that they can link their personal gestures with their intended system response. Indeed, this becomes not just desirable but necessary when gesture interaction systems are used as part of an artistic performance such as dance and other contemporary productions. In dance scenarios, for example, gestures must be specifically designed for the choreography of the piece, the specific dancers involved, and even according to the environment of the venue and available technology. If the gesture set is predefined or limited in any way, then it is difficult to see how such systems could be effectively used in performance settings in general.
(3) *Meaningful feedback.* This specific quality refers to the ability of gesture interfaces to provide meaningful feedback to users regarding how the system is interpreting

their actions. Users need to be able to readily access as much information as possible from the system during any practice or performance episode in order to understand the relationship between their action and the system's response. A certain level of satisfaction or even virtuosity comes when users can perform their actions with sufficient accuracy that they can control the system reliably enough to satisfy their intentions. This feedback should refer ideally to every action of the user with the system, including performing a movement and tuning a parameter of the system to adjust its behaviour. Without this facility, it is prohibitively difficult for users to get better at interacting with the system. Furthermore, users need to access information at *different levels of detail*. On the one hand, certain tasks might require an immediate overview of the state of the entire system, whereas on the other, tasks would require users to be able to access more detailed information regarding the analysis of their ongoing gestures at a lower level. Moreover, performers need to have the possibility of perceiving this feedback without having to look at the screen so that they are free to focus their attention elsewhere. *Different information streams* should be available to users so that they can choose what is most appropriate to the current task at hand. Finally, and perhaps most critically, users need to *access feedback synchronously and continuously* over time, just as happens when practicing a musical instrument. This immediacy in the feedback is a critical aspect of designing systems where users can effectively learn by performing. A combination of accessing different levels of information, through different available information streams (such as audio, data, visual), and having this feedback immediately and continuously as the interaction is taking place are all key to providing meaningful feedback enabling meaningful interaction for the user.

(4) *Allow expert and nonexpert use.* We want to build systems where it is the *end users* (rather than the system designers) who can define their own personal gestures. In order for this to be possible, the gesture interaction system for defining those gestures must be sufficiently *simple* and the functionality easily *accessible* by users who are not expert in either machine learning or the use of gesture interaction systems in general. The process of designing the gesture interaction for any individual, for a specific application in an environment that includes different technological components, needs to be intuitive, quick, and straightforward. When the actions of defining new gestures, testing them in a practical situation and tuning the parameters of the system are quick and tightly interleaved, and when clear guidance is provided to the user continuously over time, then the 'gesture design' workflow becomes fluid and enables a sense of flow in users' interaction with the system [Csikszentmihalyi 2008].

This ambitious goal of building *fluid gesture interaction systems* is the driving motivation of our work. We want to design, build, and test a new generation of gesture interaction systems that are sufficiently simple, responsive, and intuitive that users are fully engaged, immersed, and involved with the success of their activity so that flow is possible. In order to achieve such a system, we believe that it must, at the very least, satisfy the desiderata we have described previously. Next, we explore these desiderata in more detail by grounding our discussion through an illustrative example concerning the design of a digital musical instrument.

1.1. Use-Case: The Design of Digital Music Instruments

Let us now consider the design of Digital Music Instruments (DMIs) [Wanderley and Depalle 2004] in the context of these desiderata with the aim of trying to illuminate their significance. The evolution of computer music has made available many different sound synthesis methods than can be easily run in inexpensive computer platforms and

controlled in real time by many different kinds of input devices (e.g., MIDI controllers, computer keyboards, and motion sensors, and even by classic musical instruments). Designing a meaningful and effective mapping between the gesture of a performer and its effect on the sound of the instrument is an extremely subtle and complex task that not only matters hugely to the performer but also to the way in which an audience interprets the performance. Any gesture interface for DMI requires users to be able to define such gestures and should satisfy the four properties we described earlier:

(1) The sound needs to be affected by the gesture of the musician *continuously* during the performance. If the gesture control is limited to only triggering discrete events, little benefits might be gained compared to standard interfaces such as MIDI keyboards and controllers. Capturing the expressivity found in human performance generally requires us taking into account the intrinsic continuous nature of human motion. Considering continuous interaction, this implies taking into account both the detection of discrete triggers and the tracking of continuous variations in the gesture performance. Conductor gestures could be seen as an example, where discrete beats, tempo, and expressive elements can be communicated through the continuous hand trajectories and body motion. Precisely, the continuous trajectory is key for perception and specification of discrete events such as an isolated accent: a conductor can signal a strong accentuation through the preparation movements. This is why conductor gestures should be considered not only as a mere series of "triggering" gestures but also as complex continuous gestures that communicate both events and expressive elements. Considering beats indication and the continuous transition gestures may also help to enable musicians to anticipate new events.

    Furthermore, the influence of the gesture to the sound being generated needs to happen as synchronously as possible and certainly with very low latency of less than small fractions of a second so that the latency is effectively hidden. If the latency is significant, it becomes impossible to control the instrument reliably and to give the sense to the performer and the audience that they are actually controlling the sound through their gestures.
(2) The gestures need to be *tailored* by the performer for the specific instrument, the specific performer, and the environment where the performance is taking place. Moreover, the user should be able to create the musical metaphors that he or she wants to capture through them [Wessel and Wright 2002]. Only in this way can the correlation between gesture and sound be clear, intuitive (for the performer and, as a result, for the audience), more meaningful, and easier to remember than generic precoded gestures or standard devices such as faders, knobs, and foot pedals, which do not provide any association between the gesture and its meaning in the sound domain.
(3) When rehearsing, performers need access to *meaningful feedback* about system behaviour. Detailed screen-based information about how their gestures are being interpreted by the system and how they have been interpreted during the whole performance is a fundamental tool for allowing users to understand system behaviour and evaluate their performances to achieve better results. Having said that, system feedback should also be provided in other formats that do not force performers to look at the screen. If only screen-based feedback is available during a performance, it would force the focus of the visual attention of the performer to be fixed to a specific object that would then potentially detract from the performance itself.
(4) DMI users are not necessary domain experts in gesture recognition algorithms, and gesture design tools must be accessible enough to be used by them (*allow*

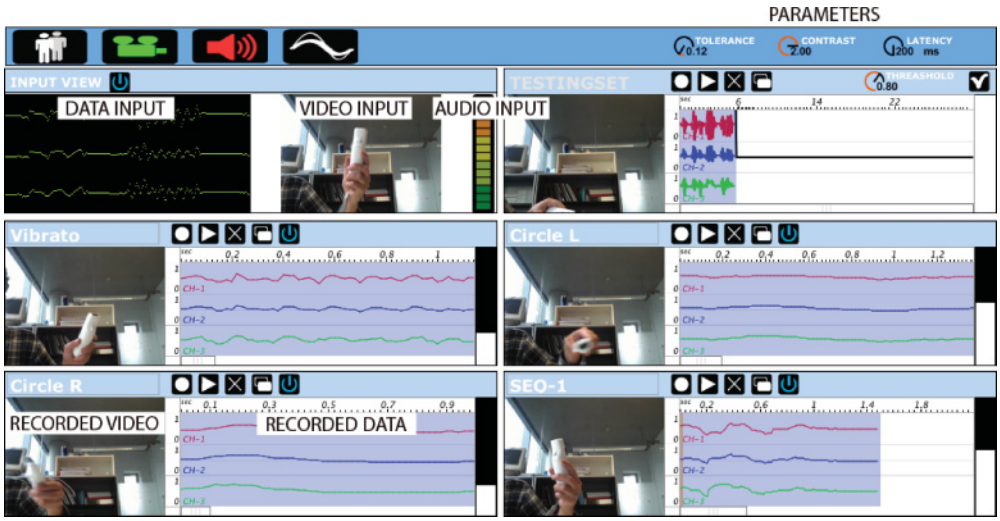

Fig. 1. Detail of the GIDE application.

expert and nonexpert use). For this reason, performers need to have feedback from the system during design and performance that guides them in understanding the system behaviour they are provoking.

As stated earlier, the explicit combination of these factors is necessary to enable fluid gesture interaction design.

1.2. GIDE: Gesture Interaction Designer Application

In this article, we present a new system called Gesture Interaction DEsigner (GIDE) (Figure 1), which attempts to address each of these four desiderata by employing an algorithm for gesture recognition called *gesture follower* that offers two critical results [Bevilacqua et al. 2010b, 2007]. First, from the moment a performance begins, it enables a continuous estimate to be calculated of which recorded gesture is the one currently being performed. This estimation happens in real time, moment by moment over time. Second, for each of these potential target recorded gestures, the algorithm provides a continuous estimation of the current temporal position of the performance within each of them. We refer to these features, respectively, as *real-time gesture recognition* and *real-time gesture following*. In this article, we provide a formal evaluation of this algorithm and show how it compares favourably to existing approaches.

We now reconsider our desiderata and provide a very brief high-level description (that will be fully elaborated in the article) of how this algorithm enables those desiderata to be met.

Real-time gesture recognition, and particularly real-time gesture following, enable GIDE to satisfy the first of our desiderata by facilitating *continuous control*. In this article, we present examples of applications that exploit our algorithm to control digital media in real time through gesture. Furthermore, the results calculated by our algorithm allow us to adopt a particular design of the GIDE application workflow to address the other points of our desiderata.

In order to achieve *tailoring*, GIDE allows users to define their gestures by recording them just once (later in the article, we will provide the details of how this feature is implemented). Moreover, GIDE supplies guidance in tuning the most important parameters of the system by providing a corresponding graphical feedback on data

visualisation and also by using metaphorical names for these parameters. This enables users to tailor the system behaviour with increased precision in intuitive ways. The act of recording gestures, testing them, and tuning system parameters are proposed as three tightly interleaved processes that make the gesture design workflow an interactive and fluid process. In this way, dancers can define gestures through performing dance, musicians can define their gestures through performing music, and players of computer games can define gestures through interacting with the game.

We take full advantage of the real-time nature of our algorithm to provide *meaningful feedback* to users relating to the output from the real-time *recognition* and *following* aspects of our algorithm mentioned previously. This feedback happens in different ways. First, we record video and audio of users when they record their gestures, and we align these information streams with the performance in real time as it is happening. This allows users to test their gesture vocabulary, seeing and hearing the playback of their recorded gestures as they are synchronised with the performance: it slows down and speeds up, exactly matching when users slow down and speed up. The attempt at constant alignment between the performance and the prerecorded gesture enables users to practice the performance of their gestures when rehearsing (which may involve recording of new improved gestures in the vocabulary.) Moreover, for a more precise comparison between current and recorded performance of gesture, GIDE allows users to visualise how the system is aligning the various streams of sensor data of the performance to the corresponding streams of the performances of the prerecorded gesture vocabulary. This visualisation offers not only a detailed measure of the differences between the performance and the reference gestures but it is also provides a clear understanding of how the system is behaving in response to the current performance.

The combination of the features described earlier enables the application to *allow expert and nonexpert use*. The real-time nature of our system, enabling continuous feedback, allows users to have a much clearer understanding of how the system is responding to their actions. In this article, we not only describe the details of the GIDE application but we also provide an evaluation of its ease of use through a case study involving 23 participants from a wide range of performance disciplines including musicians, visual and interactive artists, dancers, and programmers. By doing so, we aim to demonstrate and evaluate the potential for a new kind of *fluid gesture interaction design* and performance.

In order to achieve this, the remainder of the article is structured as follows. In Section 2, we will discuss the importance of gesture design and what we believe to be the most important work in this field. Next, we will present the algorithm for gesture recognition in Section 3 and provide a detailed evaluation of the novelty of our approach by comparing it with a widely used algorithm, Dynamic Time Warping (DTW). Section 4 then presents our GIDE system in detail, focusing on the application workflow and its usability. Next, we present details of our user evaluation study in Section 5, in which performers and artists were asked to develop gestural interfaces for their own use, before finishing in Section 6 with reflections on our contributions to this field and summarising our future areas of scientific and performance investigation.

2. BACKGROUND AND RELATED WORK

Gesture interfaces enable computer interaction through using hands or more generally with body movements. Pen [Hinckley et al. 2004], finger, and wand gestures are increasingly relevant to many new user interfaces for mobile, tablet, large display [Cao and Balakrishnan 2003; Guimbretière et al. 2001], and tabletop computers [Karlson and Bederson 2005; Morris et al. 2006]. Their applications are manifold [Mitra and Acharya 2007], ranging from medical rehabilitation [Dai 2001] and sign language recognition

[Bowden et al. 2003, 2004] to video game (using interfaces such as the Nintendo Wii and the Microsoft Kinect).

The goal of gesture recognition is to computationally analyze body movements (hand or whole body) and to associate each gesture to a predefined label (sometimes linked to a semantic meaning). In classical gesture recognition tasks, gestures are treated as whole, indivisible entities. Most approaches are designed to control discrete events once a given gesture is completed.

2.1. Gesture Recognition Systems with Continuous Output

Several systems that allow for real-time recognition have been proposed (for recent reviews, see (Turaga et al. [2008] and Mitra and Acharya [2007]). Although many systems operate in "real time," their output remains essentially discrete quantities (i.e., the gesture labels). Wilson and Bobick [1999] proposed to extend the recognition task with parameters describing gesture variations. We report here more specifically systems that were designed to provide users with a continuous flow of information characterizing their input gestures.

Visell and Cooperstock [2007] described a system based on particle filtering that tracks multiple hypotheses about user input and can display predictions of future trajectories. This system, targeting applications in physical and neurorehabilitation, was designed to allow for a close loop between the action and feedback given to the user. Williamson [2006] outlined a system for displaying information regarding uncertainty in the continuous recognition task, provided by Monte Carlo sampling methods, and its application for controlling granular synthesis as auditory display. Portillo-Rodriguez et al. [2008] presented a camera-based system based on Probabilistic Neural Networks and Finite State Machines that allows for the comparison in real time of Tai-Chi movements between a student performance and that of prerecorded ones by a teacher. The systems generates spatial sound, vibrotactile, and visual feedback based on the difference between the student and teacher gestures. Bevilacqua et al. [2010a, 2010b] developed a system called *Gesture Follower* that is designed to continuously output information about the gesture speed and similarity measures relative to a set prerecorded exemplars. This system has been used in artistic contexts for music and dance [Bevilacqua et al. 2012], and particularly in music and dance pedagogy. However, the design of such gestures requires considerable tailoring to be used in practical situations. This is a difficult process, as interaction designers are not generally domain experts in gesture or pattern recognition [Fails and Olsen 2003a], and gesture recognition systems needs to provide guidance to end users to easily define and test their gestures and adapt them for their particular needs.

2.2. Interactive Machine Learning

If gesture interfaces are to be effectively designed and tailored, there must be effective and intuitive sets of tools for interaction designers to use. At the moment, gestural interfaces are often hard coded based on ad hoc rules that make design difficult, requiring a slow back and forth between a designer and programmer to create and test the interfaces. Gestures are also limited to those for which programmers can find simple enough rules. The ability for users to define their own gestures have been demonstrated to be important in previous work [Wobbrock et al. 2009], and it is critical that their design be undertaken by movement experts rather than programmers [Hummels et al. 2006]. Although this promises generic systems that can be trained by nonprogrammers to recognisecomplex gestures, it is currently the case that most current systems (e.g., Witten and Frank [2005] and GT2K Westeyn et al. [2003]) still require a high degree of sophistication of the user. In many cases, they are required to grapple with highly conceptually difficult machine learning algorithms and thus are worse than the

hard-coded methods. The Wekinator [Fiebrink et al. 2011] is a software package that aims to make the Weka library more accessible for nonexperts, allowing users to develop real-time applications, particularly in the music domain. It provides a graphical interface to help the user in selecting and configuring different algorithms, adopting what Fails and Olsen [2003b] defined as the *Interactive Machine Learning* approach, which aims to allow users to train, classify/view, and correct the classifications. However, these systems only recognise static frames rather than temporal gestures and thus are limited in their scope and possible use.

2.3. Tools for Gesture Design

Several tools for gesture design have been released over the past decade. Crayons [Fails and Olsen 2003a] is a system for computer-vision classification of images that explicitly encourages the user to iterate through the design process by providing immediate feedback on system performance based on the training set. Exemplar [Hartmann et al. 2007] is a tool for rapid prototyping of different associations between sensor input and application logic by demonstration. It proposes techniques to both manipulate the input directly and through pattern recognition techniques to enable designers to control how users' examples are generalised to interaction rules. Wobbrock et al. [2009] presented a study for touch-screen gestures, where they asked to users to think and define gestures to associate to given tasks. Jaime Ruiz and Lank [2011] presented the results of a guessability study where they asked participants to define motion sensors–based gestures using smartphones. This work demonstrates that, in that study, consensus exists on parameters of movement and on mappings of motion gestures onto specific commands. This information was used to present a set of motion gestures and to specify an end-user inspired motion gesture set. Lü and Li [2012] presented a system for multitouch screens that allows application developers to program gestures by providing few examples, showing that the system lowers the threshold of programming multitouch gestures. Magic [Ashbrook and Starner 2010] is an accelerometers-based gesture designer tool that graphically plots recorded gestures and makes available video of the designer while performing them. It also gives feedback about the quality of the training set by testing it against a corpus of everyday activity. However, Magic is a discrete system that can only recognise gestures at the completion of their movement. No feedback is provided during the performance of such gestures.

It is our view that to fully realise the potential of gesture interfaces, we need general, usable gesture design systems that support continuous and immediate control and feedback, and in response we describe our own system—GIDE—in the next section.

3. GESTURE INTERACTION DESIGNER

In this section, we present an overview of GIDE, an application for gesture interaction design (Figure 1). The application allows for recording a series of gestures (the *gesture vocabulary*), visualising them and using them as a training set for the recognition of future gestures. GIDE adopts an interactive approach for setting up the machine learning environment in order to allow nonexpert users to take advantage of gesture recognition techniques and develop their own applications.

Keeping in mind our proposed desiderata, we want the application to satisfy the following requirements: (1) it should compute analysis regarding user performance moment by moment over time and use this information for allowing *continuous control* of the target application, (2) it should allow users to define their own gestures and *tailor* them for their needs, (3) it should provide *meaningful feedback* regarding system behaviour and the state of the system's recognition, and (4) it should provide clear guidance through the whole gesture design process and should be *easy to use* by nonexpert users in gesture recognition technologies.

GIDE has been designed to be a general purpose application for gesture recognition that can work across different application domains and media. The application is completely agnostic about the origin and type of sensor data that are used by it and can work equally well with any kind of temporal data that are regularly sampled and sent to it through the OpenSoundControl protocol [Wright et al. 2003]. We have successfully tested GIDE with several devices: motion sensors such as accelerometers and gyroscopes, a video camera using image descriptors, and sound input (e.g., microphone) using audio descriptors. However, describing in detail all of these different applications goes beyond the aims of this article, and we will instead focus on the use of two distant yet commonplace examples: finger gestures (captured using a mouse or tablet) and hand gestures (captured using an accelerometer-based motion sensor).

As we mentioned previously, a fundamental requirement for our system is to provide feedback about the state of the recognition *during* the performance. Furthermore, we want users to be able to easily record and edit their gesture vocabulary based on the feedback that they receive from the system. However, most classic machine learning algorithms, such for example Hidden Markov Models (HMMs), require the entire gesture to be completed before giving a result and therefore they are not suitable to deliver continuous feedback. We thus developed a modified version of the HMM—theGesture Follower, which has been described previously by Bevilacqua et al. [2010b]. Here, we summarise the the principles of the procedure and provide a comparison with standard DTW. We then explain in detail how the features provided by this algorithm are used by GIDE to comply with our proposed desiderata. Then, we describe several case studies of the GIDE application with the intention of demonstrating the wide-ranging potential of our system.

3.1. Features of the Algorithm

GIDE uses a variant form of the HMM for gesture recognition. HMMs can recognise sequential data using a probabilistic approach. Series of observations are modelled using a finite number of states, whose transitions are defined by transition probabilities. Each state emits observations based on a probability distribution function. Generally, the HMM's parameters are set through training procedures using a large database statistically representative of all possible variations. However, for interactive gesture design, this would require the collection of a large number of user data, with each user repeating gestures many times. This would obviously limit the interactive procedure between designing gestures and receiving feedback on the gestural interface behaviour. On the contrary, GIDE is designed to allow users to rapidly define and test gestures as required by our second desiderata: tailorability. This is why the learning procedure needs to be as quick and simple as possible. GIDE therefore uses a hybrid approach between probabilistic HMM and exemplar-based approaches such as DTW, which requires only a single-gesture example to specify gesture class. The "modified" HMM approach used by GIDE sets off the Markov models from a single example by associating each example sample data with a state, as shown in Figure 2. Such a choice leads us to consider a large number of states and is thus a less efficient decoding computation compared to standard HMM approaches. However, two points should be noted. First, the loss of efficiency was never found to be limiting in our application. Second, our approach offers the crucial advantage to closely model the data time profiles, which might be lacking in a standard HHM. In particular, similar to DTW, it is possible to temporally align the incoming data with the original example at the granularity of individual samples. Compared to DTW, our approach allows for real-time decoding during the performance (using the HMM forward procedure), whereas standard DTW is operated only at the end of the gesture. A similarity measure can be estimated with the same time granularity (at the sample level). These two features together enable

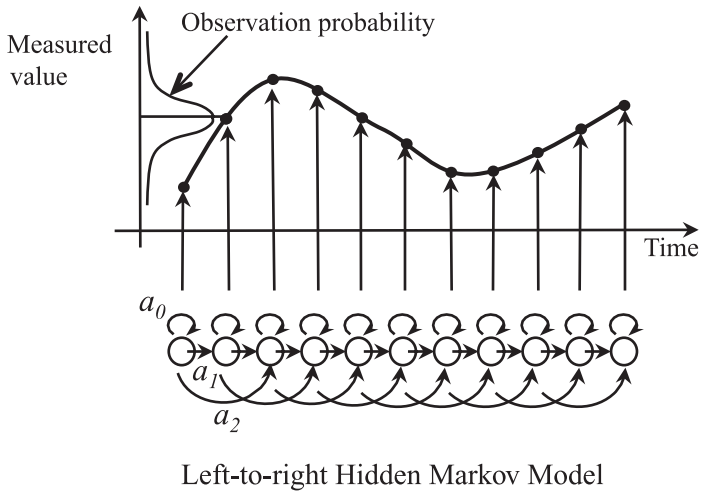

Fig. 2. Learning procedure: a left-to-right HMM is used to model the recorded reference. The HMM has a separate state for each sample of the training data.

the first desideratum: continuous control. Moreover, when performing recognition, the HMM associated with each gesture is evaluated, and the one with the highest similarity measure (i.e., the highest likelihood) is used to classify the gesture. When performing recognition, the HMM associated with each gesture is evaluated, and the one with the highest computed probability is used to classify the gesture.

As in standard machine learning techniques, the workflow is divided into two phases: learning and decoding. During the learning phase, the temporal profile of the gesture is recorded and used to create a left-to-right HMM by directly associating each sampled point to a state of the HMM. Each state i corresponds to a sample in the training data and is associated with a gaussian probability distribution b_i, which is used to compute the probability of an observation O:

$$b_i(O) = \frac{1}{\sigma_i\sqrt{2\pi}} exp\left[-\left(\frac{(O-\mu_i)^2}{2\sigma_i^2}\right)\right], \tag{1}$$

where μ_i is the ith sampled value associated with state i, and σ_i is parameter that can be interpreted as the standard deviation occurring between recorded references and performances. Since the HMM is trained using a single example, σ_i cannot be estimated and therefore must be set using prior knowledge or be dynamically adapted depending on the accuracy of the performance. This parameter σ_i is directly related to one of the important parameters of the application, which we called *tolerance*, that is defined by $2\sigma_i$ and is expressed in the same unit measure as the sensor data.

Because the states correspond to frames in the original gesture, transitions between states correspond to transitions from one frame of the original motion to another. We have three nonzero transitions probabilities: a_0, which is the probability of staying in the same frame; a_1, which is the probability of moving to the next frame; and a_2 which is the probability of jumping to two frames ahead. These transitions probabilities correspond to different speeds of performing the gesture: respectively, movements that are slower than the original, those that are the same speed, and those that are faster. In order not to bias the model towards certain movement speeds, we set these transitions probabilities to have equal values: 0.33, 0.34, and 0.33, respectively.

The decoding phase follows standard forward procedure to HMM [Rabiner 1989], corresponding to a causal inference (i.e., the inference is estimated without the knowledge

of future events, as the appropriate standard Viterbi algorithm that operates, without causality constraints, on complete gestures). This procedure requires the computation of a distribution $\alpha_i(t)$, which corresponds to the probability distribution of the partial observation sequence until time t and state i. This distribution is estimated iteratively in real time each time a new observation is received and makes it possible to compute two important values: the time progression of the sequence, which is related to the recorded example, and its likelihood. For details regarding such a procedure, please refer to Appendix A.

The likelihood estimation depends on the *tolerance* and a second parameter called *latency*. Precisely, for every incoming sensor observation, the system computes a likelihood relative to each reference gesture. These likelihoods are computed by averaging "instantaneous" likelihoods, refereed to each coming observation. The average is computed using a sliding window, whose sizedepends to the number of frames taken into account. For example, a window size of 50 frames at a frame period of 20ms will consider 1s of the performance. A high value of this parameter guarantees more stable results, but it will also add latency to the system in outputting accurate recognition estimation, typically during the transition between two gestures.

Finally, note that the computation of the selection of the correct gesture can be performed in two different manners: either selecting directly the one with highest likelihood value computed as explained earlier, or adding a constraint on the speed of the gesture performance to be in a given range, such as between half and twice the speed of the reference gesture and rejecting those outside this criteria.

This algorithm allows our application GIDE to provide the following features:

Real-time recognition. This algorithm returns a real-time moment-by-moment probability that the gesture being performed is the same as each of the prerecorded gestures in the recorded gesture vocabulary. This probability information is updated continuously while the gesture is being performed. In other words, it is updated with a frequency that corresponds to the sample rate of the incoming sensors signal (typically around 5 to 20 ms) from the very first sample of the gesture.

Following. Our algorithm also tracks a best estimate of the temporal position of the currently performed gesture compared to prerecorded ones. In other words, in real time, the system aligns users' performances to their gesture references. We refer to this property as *following* a gesture.

Quick learning. As explained previously, only one example per gesture is needed. This makes the procedure of defining new gestures quick and simple. In Section 4, we will explain in detail how this feature is used in GIDE to help in enabling an *interactive machine learning* process.

As will be explained in detail in the next section, one important feature of GIDE is the possibility of changing the parameters of the algorithm in real time and thus directly observing how this affects the behaviour of the system.

In Figure 3, we show how the *tolerance* and *latency* parameters are important and strongly affect the performance of the system. As we can see from the figure, performances of the system against the database converge to a highest peak with a tolerance value equal to 0.125 and a window size equal to 100% of the gesture size used for testing. This graph shows that an optimal setting exists, and it is important to guide users into the process of parameter tuning. In Section 4, Workflow, we will explain how the GIDE application provides real-time feedback about the influence of these parameters.

3.1.1. Implementation. The GIDE application has been implemented using the software Max (Cycling74); *MuBu* [Schnell et al. 2009]; and a C++ library called *gf* [Bevilacqua et al. 2010a], which implements the algorithm for gesture recognition described earlier.

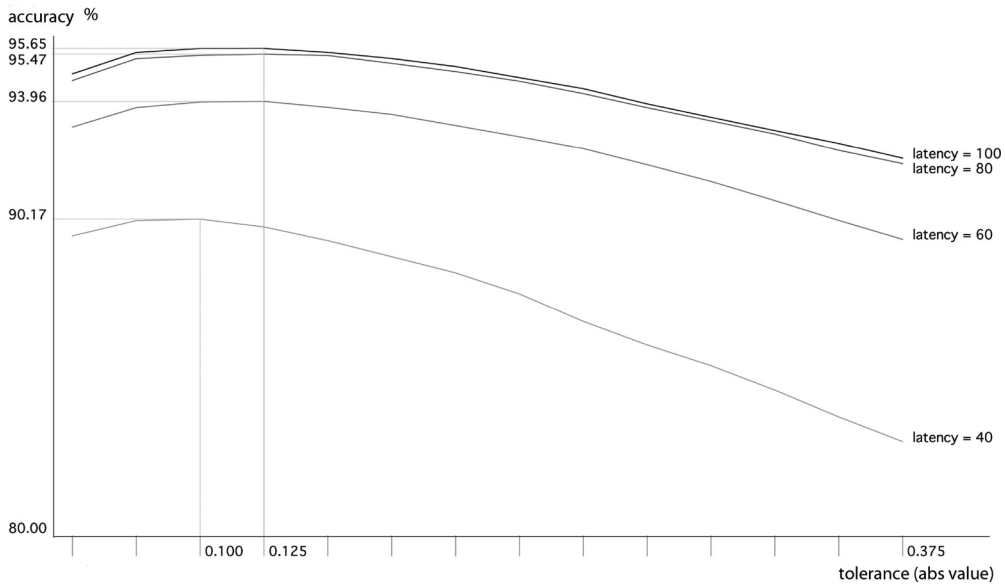

Fig. 3. Evaluation of algorithm performances in success rate using Wobbrock's 2D database. Results are shown for different values of the "latency" parameter. The *x*-axis represents variations of the *tolerance* parameter, which is a parameter of the algorithm explained in Section 3.1. The *y*-axis displays the success rate. The value of the *tolerance* parameter is expressed in the same unit as the sensor data, which in our evaluation have been normalised by their maximum value to be in the range of [0,1].

Table 1. Gesture Follower Versus DTW

Algorithm	Gesture Length (%)	Success rate (%)
GF	25	62.3953
GF	50	83.6696
GF	75	92.2431
GF	100	95.3385
GF with constraint on the speed	100	97.3675
DTW	100	97.1788

Success rate of Gesture Follower and DTW algorithms at different temporal position of the gesture length using Wobbrock's 2D database.

3.2. Algorithm Evaluation

In order to evaluate the algorithm, we used the 2D gesture database provided by Wobbrock et al. [2009]. This database contains data from 10 users drawing 16 different symbols in two dimensions. Users repeated the drawing 10 times at three different speed rates: fast, medium, and slow. The total number of examples within the database are then 10*16*10*3 = 4,800. Unlike Wobbrock's evaluation, which is offline, we evaluated our system under real-time constraints, without any data transformation that typically require knowledge about the entire gesture, such as the average scaling or rotation angle around the centroid. The only preprocessing treatment we used was the translation of gestures to the origin, which is obtained by subtracting each of the points from the previous ones and can thus be computed in real-time conditions.

For each speed rate and for each user, we iterated over the 10 examples provided, considering one series of recording as the training set and the other 9*16 for testing. In Table I, we report the success rate of our algorithm after 25%, 50%, 75%,and 100% of the gesture length, respectively. The table also reports results of our algorithm using the

speed constraint mentioned earlier. We then compare these results with the standard DTW algorithm.

As we can see in Table I, results show that Gesture Follower recognition can be almost as good as a standard offline output, such as the one provided by DTW, and that the correct answer is estimated correctly in real time in most cases. Our algorithm estimates the result correctly with 62.4% of success rate after a quarter of the gesture, 83.7% at half, 92.2% at three-quarters, and 95.3% at the end. This also shows that, as expected, the recognition rate increases with the degree of gesture completion. Interestingly, the convergence is relatively fast considering that the difference of the recognition rate between 50% and 100% is only 11%. Note also that the algorithm reaches 97.4% of the success rate if an additional constraint is added (which is slightly higher than the 97.1% given by DTW). This constraint corresponds to taking into account only gestures with duration comprised between half and twice the template duration. This is equivalent to taking into account a gesture only if its average relative speed (to the template speed) is between 0.5 and 2.

Parameters setting is critical to achieve high performances with our algorithm. In Figure 3, we report performance measurements using Wobbrock's database varying the latency and the tolerance parameters. As clearly shown in the figure, performance results converge to optimal with a given setting of these two parameters.

3.3. Implications for the Proposed Desiderata

Now that we have described what the algorithm for the recognition offers, we want to discuss how these link with our proposed desiderata for gesture interfaces.

The real-time nature of the algorithm and its moment-by-moment computation of *recognition* and *following* tasks provides information that can be used to control the target application continuously over time. In Section 4.4, we show three example applications that exploit these features to control digital media in real time through gesture.

The second desideratum, *tailorability*, is satisfied by allowing end users to define their own gesture vocabulary. As explained previously, the application allows users to define and test their gestures by recording them just once. Additionally, the parameters of the algorithm described earlier can be tuned by users when they design and test their gestures. The application provides guidance through this process by visual feedback and the use of metaphorical names. This feature will be described in detail in the next section. The act of recording gestures, testing them, and tuning system parameters are proposed as three tightly interleaved processes that make the gesture design workflow an interactive process.

Information regarding real-time gesture recognition and following are displayed to users continuously during their performance, providing *meaningful feedback* and guidance in understanding system behaviour. Such feedback is provided by the application in three different modalities: video alignment, audio alignment, and data alignment. These features are explained in detail in the next section of the article.

The combination of the features described previously allows the design of the application to be particularly *easy to use* by nonexpert users. The real-time nature of our system, and consequently the continuous feedback provided by the application, allows users to have a clear understanding of how the system responds to their actions. Furthermore, the graphical feedback on the effect that system parameters have on the performances of the system, combined with the adoption of metaphorical names, aims to make the task of parameter tuning (which has been traditionally difficult in machine learning) as easy as possible. In Section 5, we provide details about the usability of the application through a user evaluation case study.

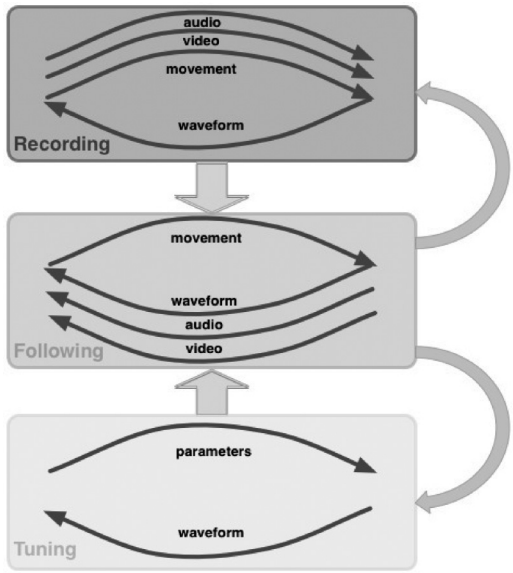

Fig. 4. GIDE workflow. The gesture design workflow divided into three tightly interleaved phases: gesture recording, gesture following, and parameter tuning.

4. WORKFLOW

As shown in Figure 4, the workflow of the application consists of three tightly interleaved phases that allow user interaction with all aspects of the design process: gesture recording, gesture following, and parameter tuning. To accomplish the *fast and focused* UI principle, the entire process is iterative, and users can quickly switch between phases during the design process. In this section, we describe these phases in detail.

4.1. Phase 1: Recording a Gesture

GIDE allows users to easily build the gesture vocabulary. They can quickly record a gesture; view it in the graphical interface; test it; and, in the case they are not satisfied with it, record it again. We refer to the set of recorded gestures as *gesture vocabulary*.

Each gesture within the vocabulary is accompanied with a graphic component called the *gesture editor*, as seen in Figure 1. A gesture can be of any length of time, and it is represented by the following components: a name, a multiwaveform for sensor data, and optionally a video sequence and an audio waveform. As we described in the previous section, our algorithm allows users to define gestures by recording them just once. This ability has allowed us to design this phase to be as close as possible to the one of recording audio and video in standard AV production sequencers, making the handling of sensor data as straightforward as possible.

At the beginning of a new session, the gesture vocabulary is empty. The user can then decide to record a gesture either by pressing the record button via the GUI or triggering the "record" command remotely. In the Evaluation section later in this article, participants were able to trigger the record function both with the mouse and with a Nintendo Wiimote.

Once this happens, the application starts recording incoming sensor data together with videoand audio from attached cameras and microphones. Users are encouraged to also record a sound while recording a gesture (e.g., spoken sentences), in order to have richer feedback during recognition. This multimodal stream of data is graphically represented in the *Input View* on the top left of the application. Furthermore, during

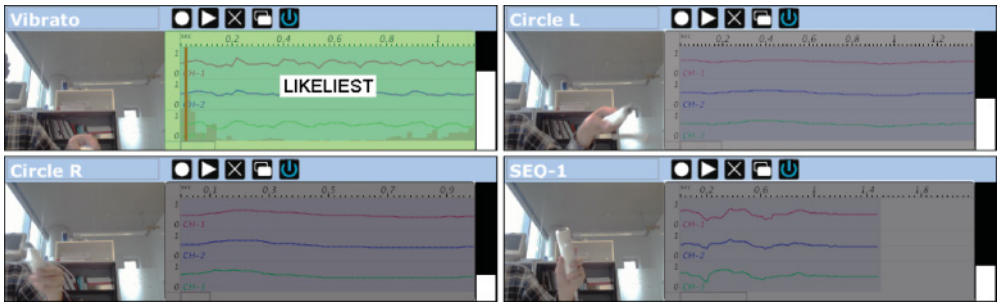

Fig. 5. Real-time gesture recognition. GIDE allows for real-time gesture recognition by continuously computing a likelihood measure between the performance and every prerecorded gesture. The likelihood of a gesture graphically corresponds to a level of transparency. The green gesture is the likeliest one. The *contrast* parameter increases or decreases the difference between high and low likelihood values, as shown by the associated colours.

the recording, the user can see the recorded data in the *gesture editor* related to the gesture that is being recorded.

After the recording, the user can select a part of the gesture (e.g., to discard a silence at the beginning or the end), zoom, scroll, and play back at different speeds. A "Pop-Out" button is available to open a new resizable window if more space is required. It is also possible to add a *temporal marker* in a specific point of the multiwaveform by double-clicking on it.

4.2. Phase 2: "Follow" Mode and Real-Time Feedback

When the gesture vocabulary contains at least one gesture, users can start evaluating their vocabulary with a real performance. GIDE supports the traditional batch testing, present in classic machine learning tools, as well as a real-time testing called *follow* mode. Batch testing is described in more details in the next section.

In *follow* mode, as users perform a live gesture, the application gives a moment-by-moment probability estimate of which gesture they are performing and *where* they are within that gesture. The system performs continuous recognition based on incoming data and gives a real-time estimation on its similarities against each prerecorded gesture.

The probability of each gesture to be recognised is represented visually by the transparency of its associated editor, whereas the one with highest probability becomes green (Figure 5). As well as the current probabilities for each gesture, we have a precise estimate of the temporal position within the gesture. As mentioned previously, we refer to this feature as *following* a gesture.

This enables GIDE to provide a real-time multimodal feedback on the recognition during the performance. This multimodal feedback is composed by the three following aspects:

Video alignment. Each video panel plays back the prerecorded video synchronised with the performance. Typically, this allows users to compare their performance with the corresponding video image of themselves when performing the recorded gesture.
Audio alignment. The audio that users recorded is played back synchronously during the performance. The application allows the user to decide between playing back only the audio of the likeliest gesture or to do a *mix* (i.e., associating each gesture to a volume playback that is proportional to its likelihood). In this way, users have an auditory feedback on which gesture has been recognised. They also have an auditory

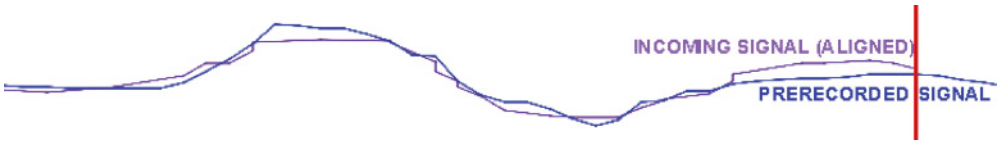

Fig. 6. Real-time gesture following. GIDE allows for real-time gesture following, aligning the performance with prerecorded gesture continuously over time. In the figure, we see the waveform of the incoming data stream (purple) aligned frame by frame with the corresponding position of the prerecorded one (blue). The red cursor represents the temporal position of the gesture.

feedback of the alignment of their performance with the prerecorded gesture, as the prerecorded sound is played back on the temporal position of the *follower*.
Waveform alignment. The temporal position of the performance within the prerecorded gesture is displayed through a red cursor over the data multiwaveform (Figure 6) together with a *probability function*, an orange waveform that displays the probability associated with every frame. Furthermore, we have implemented what we call an *alignment view* which, when enabled by the user, displays a pink multiwaveform superposed to the original one, representing the incoming sensor data aligned to the reference gesture frame by frame. In this way, the user can clearly see the difference between his performance and the prerecorded one as a vertical distance between the two multiwaveforms in every point. This representation works particularly well in association with the *tolerance* parameter described later in the article.

4.2.1. Batch Testing. Within phase 2, GIDE also supports batch testing by a facility called the *testing performance*. In our design, we have given the testing performance a very similar appearance to the gestures contained in the gesture vocabulary. The testing performance allows the user to record an arbitrarily long real-world dataset and then test it iteratively against the gesture vocabularywhile changing gestures and parameters to obtain best results.

When the user clicks the testbutton, the system reads, in a row, all of the data stored into the testing performance, as if this data were coming in real time from a performance by a user and thus instantly highlights all areas in its multiwaveform where the likeliest gesture reached a thresholdgiven by the user.

As for the gesture vocabulary, it contains both sensor data, video, and audio and supports the *retrospection* property: users can select gestures, play them back, and re-record.

Thanks to the low computational cost of the algorithm, the time for testing a dataset of a few minutes is typically a few milliseconds. This provides the user with information in order to redefine the gesture vocabulary and tune parameters in an interactive way, seeing the results appear instantaneously in the graphical interface.

4.3. Phase 3: Parameters Tuning

As previous studies have shown [Fiebrink et al. 2011], users often have difficulties understanding how to tune parameters of machine learning algorithms. However, as we show in Figure 3, parameters of our algorithm are critical to reach high performance in the recognition task. We provide support for this process in three different ways.

First, we have assigned a name for each parameter, which aims to supply a useful *metaphor* for the user describing a common digital media practice. Second, we have added short *text hints* about how to use each of the parameters. Finally, the effect of two of the three parameters (tolerance and contrast) has a corresponding *representation* in the graphical interface, helping users to better understand how they affect system performances.

(1) *Tolerance.* The role of this parameter in the algorithm corresponds to a constant standard deviation of the HMM, as has been explained in Section 3.1. This parameter has been explained to users as *how much the performance is allowed to be different from prerecorded gestures*, and we have therefore named it *tolerance*. This name is a useful example of a metaphor based on common digital media practice: in Adobe Photoshop, there is a parameter with the same name that determines the range of colour that the Magic Hand tool selects. Similarly, in GIDE, this parameter is graphically associated with the *thickness* of the sensor data multiwaveform and shows the range of values that determine whether the performance belongs to the gesture. We have found that this works particularly well in combination with the *waveform alignment* described in the previous section, as users can see the distance of their aligned gesture compared to the tolerated range of values.
(2) *Latency.* The actual probability that each gesture recorded in the vocabulary is a match for the current performance is computed as an average of probabilities calculated for each frame and stored on a sliding window. Thus, the size of this window specifies the amount of time taken into account for accurate gesture estimations. The effect of this parameter is basically to affect the latency of the system. If the parameter is set high, it will recognise gestures highly reliably but will react slowly to changes in user input. If it is set low, the system will react faster but less reliably.
(3) *Contrast.* In our system, the value of the probability of each gesture in the vocabulary is normalised such that their sum is always equal to 1. For practical reasons, we have designed a parameter to tune this normalisation in order to increase or decrease the difference between high and low probability values. The definition that is given to the users is as such: *turning up the contrast parameter heightens the differences between gestures.*

    The word *contrast* works as a metaphor, as we think about the contrast of an image quickly. As gesture probabilities are graphically represented as the transparency of their associated gesture editor, tuning up the contrastparameter will increase the contrast of the colours of such editors. Figure 5 shows the application with the contrastparameter set to a high value.

4.4. Example Applications of GIDE

Having looked at the workflow of the application for designing new gestures, we now move our attention to consider how user-defined gesture following can be applied in real-world scenarios. Here, we show three different stand-alone applications for the gestural control of digital media. Theses scenarios are based on cases that were previously prototyped with the *gesture follower* but without the integration of the user interface of GIDE.

(1) *Video scrubbing.* This first application is inspired from use in the installation *if/then installed* (by Siegal, Bevilacqua, Berenger, Goidell, Lambert; http://www.thebakery.org/interactive-if-then-installed). This installation, using the gesture followeralgorithm, demonstrated the interest of a videoscrubbing, which is explained later. The installation was designed using predefined gestures. The use of the GIDE interface, allowing users to add their own gestures, could extend this interaction paradigm to a wide range of applications.

    The gesture-driven video scrubbing works as follows. First, users select different video files, one for each gesture they want to learn. Then, as soon as they start recording a new gesture, the corresponding video file is played back. This allows users to *mime* to the video while they record their gesture. When users switch to *follow* mode, the video of each of the recorded gestures in the vocabulary is aligned to the most likely position and played back by GIDE. The user can switch between

the *likeliest* mode, where only the video that corresponds to the likeliest gesture is played, and the *mix* mode, where all videos are played back and their transparency is mapped with the likelihood of the associated gesture. This technique could be used to implement a gesture interface for VJing, where users can continuously control the video playback through their own gestures.

(2) *Supervised continuous sonification.* The second application is similar to the one described previously but uses sound instead of video. It allows the user to continuously align the playback of a sound file with a gestural performance. This paradigm was previously validated in pedagogical scenarios where students can "conduct" recorded music using gesture input [Bevilacqua et al. 2010b, 2007]. First, the user loads different sound files, with each one associated with an empty slot in the gesture vocabulary. Second, as soon as the user starts recording a gesture, the loaded sound associated to it is played back. Thus, the user listens to the sound when recording the gesture, adapting the performance with the tempo of the sound or *mime* the sound itself. When the user switches to *follow* mode, the sound is played back following the temporal position of the performance, which is given by GIDE. So, when the speed of the performance slows down, the sound playback slows down as well; when the performance accelerates, the playback accelerates as well. The sound is thus continuously *aligned* with the performance. This allows the user for a *supervised sonification* of a gesture based on previous recording. In order to keep the sound more natural and close to the original one, we employ a phase vocoder. This technique allows to leave the pitch of the sound unchanged while changing its playback speed. As for the previous application, two options are available: in the *likeliest* mode, only the sound associated to the likeliest gesture is played back; in the *mix* mode, all of the sounds are played back and their volume corresponds to their likelihood.

(3) *Triggering.* The last application allows to trigger a series of digital media based on discrete temporal positions along a single gesture. Such scenario is similar to previous artistic applications of the gesture follower, allowing gesture-based system to trigger sound processes, as described in [Bevilacqua et al. 2012]. We have designed GIDE so that it is possible to place a named marker at a specific temporal position within a gesture. During a performance, at any time the temporal position of the likeliest gesture reaches one of these temporal markers, a different sound of video can be played. Other types of digital media, such as MIDI notes and light control, could be controllable in the same way.

Having described the interaction design process and looked at the architecture of the application in detail and especially its novel mechanisms for real-time feedback, we now move on to a presentation of our set of experiments to show how the system was used in practice.

5. USER EVALUATION

We perform a user study to assess how well the application fits our desiderata when used by artists, musicians, and designers. Specifically, we were interested in evaluating how the multimodal feedback that happened during performance could be used to help design and modify gesture vocabularies. Moreover, we set up our experiments with the intention of assessing the importance of each component of the multimodal feedback channels: the video of the user performing the gesture, the auditory feedback, the sensor multiwaveform, and its cursor and the waveform alignment. Finally, we wanted to evaluate the degree of which using metaphorical, intuitive names along with graphical feedback for parameters helped users tune them effectively.

5.1. Participants

In order to gain a better understanding of how users might experience GIDE, we organised a workshop, inviting 23 participants from different domains, including 5 professional music players, 13 electronic music performers, 3 visual or interactive artists, 1 dancer-choreographer, and 1 programmer. All participants had experience of using digital media in their artistic practice and were familiar with many standard digital production tools such as Digidesign ProTools and Apple Final Cut. In addition, 15 of the 23 were familiar with visual programming environments such as MaxMSP and PureData, 7 of these were also used textual programming language in their artistic practice, and 4 of these had some familiarity with machine learning theory. Eight of the 23 had no familiarity with anything other than standard digital media production tools.

Ten of our participants were PhD students in music and computational art, two were university faculty members in music, and the others were independent artists. All participants were between 23 and 35 years of age, and the experiments took place over six sessions in London and Edinburgh. Each session had between 3 and 8 participants and lasted about 2-1/2 hours. We named the workshop "Workshop on real-time gesture recognition for performing arts," and all participants applied spontaneously and were not remunerated.

5.2. Workshop Procedure

Every session of the workshop has been divided into five parts, and participants were asked to fill in the relative section of our questionnaire after each part. All sessions were video recorded in theirentirety. The seven parts are introduced below:

(1) Introduction to gesture recognition systems. We started each workshop by explaining to users the general concept behind gesture recognition and the difference between direct mapping from sensor data and control parameters as opposed to user-defined gesture recognition.

The first video depicts the *video scrubbing* paradigm described previously and can be found at http://tiny.cc/9mpibw. It detailed an interactive installation that was built several years ago (using the previous *gesture follower* system) and shows a dancer performing live in front of a big screen. The screen displays a recorded second dancer that appears to mirror the live dancer doing the same actions at the same time. This video clearly shows an interaction based on a continuous output of the gesture recognition system.

The second example, called *Augmented Violin*, explains the *supervised continuous sonification* paradigm (also using the previous *gesture follower* system). It shows a live violin player performing a piece at different speeds while a second recorded violin (recorded by the same player) accompanies the live performance following the tempo.

The section of the questionnaire relative to this first part asked users to evaluate whether they felt that they understood the basic concept of continuous gesture recognition and what the interaction between the gesture and the sound was in the two videos we showed.

(2) Playing with an existing application. We showed users a video about an interactive installation called *Granularia* that we presented at the Festival of Science in Genoa in 2010 (which is publicly available at http://tiny.cc/d1bux). This installation allowed the user to control different sound engines by moving a mobile phone on the air. Specific gestures were recognised and used to trigger associated sounds. We then asked users to try the same application using a Nintendo Wiiremote in order to become familiar with the possibilities offered by supervised gesture interaction.

Through the questionnaire, we asked whether users understood the goal of this application and were able to control this system reliably.

(3) Designing a single gesture. At this point, we asked users to run the GIDE application on their computers. We demonstrated how to record and follow a gesture through the application using a Nintendo Wiiremote. We explained that both video and audio were being recorded, and we showed a basic example of an association between a gesture and a vocal sound. We then asked users to try to record their own gesture in their computers and evaluate the different components of the application.
(4) Building a gesture vocabulary and testing the recognition in real time. In this phase, users were asked to record several gestures to create a vocabulary and experiment whether or not they could be triggered in a subsequent performance. They tested the gestures that they had recorded switching the application in the *follow* mode and performed similar gestures again looking at the various real-time feedbacks provides by the application, as explained in Section 4.2. We also gave particular hints about how to tune parameters as explained in Section 4.3. As they were performing this task, we recorded and monitored the strategies that they took to record their gestures and tune the proposed parameters.
(5) In the batch testing part of the workshop, we explained to users how to evaluate performances using the batch testing feature that we described in Section 4.2.1.
(6) Developing an application for gesture sonification. As users became familiar with the application, we showed them how to use information provided by GIDE to accomplish the task explained in Section 2—supervised continuous gesture sonification. We showed how to create an application in the MaxMSP environment (http://www.cycling74.com) that loads sound files and developed a mimicking paradigm between gestures and sounds through gesture recognition: the provided application allows users listening to a sound to mime it with their hands and then perform the same gesture again at a different speed and hear the stretched sound.
(7) Developing an application for video scrubbing. In this part, we explained the video scrubbing paradigm (Section 4.4) and showed users how to build their own application using the MaxMSP environment.

5.3. Measures

We evaluated the study through both a questionnaire and a semistructured interview. After each of the seven sections of the workshop just described, we asked participants to fill out the relevant part of the questionnaire. For all but the first two sections of the workshop, we asked users if the application worked as they expected to accomplish the proposed task. Furthermore, we asked them to evaluate the different components of the application, which are the video scrubber, the waveform cursor, the alignment view, the probability waveform, the tolerance parameter, the responsiveness contrast, and the background colour changing. Questions about both usability and evaluation were repeated for each section of the workshop, with each question presented as a seven-point Likert- like scale. Users were invited to add commentaries at every stage. The semistructured interview took place after the session, and participants were asked how happy they were with their results, the kind of strategies they used to design their gestures, the level of usability and usefulness of the system, and whether they would use this application for their works.

5.4. Results

In this section, we first present the results of the questionnaire and relate them to the achievement of our proposed desiderata. We then discuss the different strategies used

by participants. Finally, we debate some issue of the application arising during the workshop.

5.4.1. Questionnaire. The questionnaire results are shown in Table II. The questionnaire responses were analysed with a one-sample Wilcoxon signed rank measuring the difference between the sample responses and the midpoint of the scale (4). The mean of all answers was above the midpoint (indicating a favourable response), with all but two being significant to at least $p = .1$ and the majority being significant to $p = .001$. Some of the later questions had a lower number of responses due to not all participants reaching the last part of the study in the allocated time. This might account for the lower levels of significance to the later questions. For all sections, participants were asked whether the system worked as they expected; in all sections the mean answer was higher than the midpoint with $p = .01$, except the final (video scrubbing) section, where the significance was $p = .05$. For the final two sections, they were asked whether they understood the interaction between their gestures and the sound or video; mean answers to both these questions were significantly above the midpoint to $p = .01$. In the final two sections, they were also asked whether they could control the system reliably; mean answers to these questions were significantly about the midpoint to $p = .05$ (audio condition) and $p = .1$ (video condition). Participants were also asked to rate the usefulness of each of visualisations for each stage of the study. Mean ratings were significantly above the midpoint in all but two cases. There was no significant difference in the ratings of different visualisations.

5.4.2. Achievement of the Desiderata. In this section, we describe participants' behaviour at each stage of the study in respect of the desiderata of gesture interfaces we proposed in the introduction of the article.

(1) *Continuous Control.* Continuous control has shown to be a very important feature of GIDE in terms of the range of applications that it can allow to control. In Section 3.1, we showed that our algorithm is capable of real-time control, and our participants answers to the questions "Did you understand the interaction between the gesture and the [sound/video]?" and "Are you able to control the system reliably" shows that they were able to understand and use continuous control. Furthermore, in the semistructured interview at the end of the workshop, when we asked participants to imagine how GIDE could be useful for their own practice, they answered enthusiastically. In addition, many of the strategies they developed, described in Section 5.4.4, are clearly inspired by the possibility of controlling the target application continuously over time.
(2) *Tailorability.* All of our participants were able to record gestures of their own design and use those gestures effectively, demonstrating their ability to tailoring the application by tuning parameters of the algorithm and using different strategies for designing gestures. We discuss these points in detail in Sections 5.4.3 and 5.4.4.
(3) *Meaningful Feedback.* The continuous control features of our algorithm also enable us to give the different forms of real-time multimodal feedback described in Section 4. The questionnaire responses show that the participants found these to be useful. In our semistructured interview, when we asked them which component they found the most useful between video, audio, and waveform, almost all participants claimed that they could not decide, as it was the combination of all of them together that was needed to be useful. This is supported by the lack of significant difference in our questionnaire results. Some remarked that only one of them on its own would not be enough for a controlled performance.

    However, in terms of ranking, they took *video* to be the most important at the beginning, but slightly less important compared to the others after working with

Table II. Questionnaire Results

QUESTION	N	mean	W	
1 - Introduction				
Did you understand the interaction between the gesture and the sound?	19	6.63	190	****
2 - Playing with an existing application				
Do you understand the goal of the application?	15	6.93	120	****
Are you able to control the system reliably?	15	5.93	105	****
3 - Designing a single gesture				
Did it work as you expected	22	6.00	210	****
video	22	6.05	224	****
waveform cursor	22	5.86	241	****
warping view	22	6.00	231	****
probability waveform	22	5.27	183.5	***
tolerance parameter	21	6.43	231	****
4 - Building a gesture vocabulary and testing the recognition in realtime				
Did it work as you expected	22	5.59	219	****
video	22	5.91	206	****
waveform cursor	22	5.77	204	****
warping view	20	5.10	169.5	**
tolerance	22	6.09	210	****
responsiveness contrast	21	6.05	229	****
background colors	22	5.55	162.5	****
5 - Batch testing				
Did it work as you expected	20	6.30	190	****
video	18	5.39	114	**
waveform cursor	19	5.47	161	***
warping view	19	5.37	103.5	**
tolerance	18	6.44	153	****
responsiveness contrast	18	6.33	166.5	****
background colors	18	5.83	144	***
batch testing	19	6.42	190	****
6 - Developing an application for gesture sonification				
Did it work as you expected	11	5.82	64	***
video	10	5.90	42.5	**
waveform cursor	11	6.27	55	***
warping view	10	5.20	29	
tolerance	11	6.55	66	***
responsiveness contrast	11	5.73	57	**
background colors	11	5.36	30	*
batch testing	9	6.44	45	***
Did you understand the interaction between the gesture and the sound?	10	6.10	55	***
Are you able to control the system reliably?	9	5.22	26.5	**
7 - Developing an application for video scrubbing				
Did it work as you expected	10	5.80	52	**
video	10	6.40	45	***
waveform cursor	9	6.44	36	***
warping view	9	6.44	45	***
tolerance	9	6.00	42	**
responsiveness contrast	8	5.75	32.5	**
background colors	7	5.57	23	
batch testing	7	6.43	28	**
Did you understand the interaction between the gesture and the video?	10	6.00	45	***
Are you able to control the system reliably?	9	5.33	31	*

Note: The significance levels are ****$p < .001$, ***$p < .01$, **$p < .05$, * $p < .1$. N is the number of participants answering a question. W is the Wilcoxon signed-rank statistic.

the system, as they could begin to remember their gesture and not need it so much. They remarked that if they did not work on a piece for several days, they would then need to look back at the video to remember how they performed, as the waveform alone would not be sufficient.

Participants described the *sensor multiwaveform* as a useful way to get an overview of the whole gesture and that the red *cursor* over the waveform was extremely useful in initially understanding the concept of following a gesture. They claimed that even if they were not familiar with acceleration, they understood the relationship between the gesture and the waveform quite soon. However, many of them also said that they were not able to discriminate between the three different axes of the acceleration, and they would focus more on the global *motion* of the gesture. This emphasises the point made by others that raw motion data is often not expressive enough, and we need a better representation (e.g., see Kratz and Ballagas [2009] and Linjama et al. [2008].)

Almost all participants recorded their voice in association with the gesture and usually associated specific words or sounds to particular parts of a gesture instead of recording one continuous sound. For example, different syllables or screams were often associated with the more energetic parts of the gesture. Auditory feedback was felt to be the most precise one about the likelihood and the speed of their performance. Some participants, especially the performers, expressed particular interest in the ability to use auditory feedback, as it permitted them not to have to be forced to look at the screen, thus enabling performances to be more free. However, they said that this was possible only after a number of iterations, as this freedom relied on the ability to remember their gesture vocabulary quite well.

Participants familiar with audio-video production tools found a clear analogy in GIDE even if they had never seen an acceleration waveform before.

(4) *Allow Expert and Nonexpert Use.* It was clear from the completed questionnaires and interviews that participants understood the goal and the behaviour of the application after just a few minutes of testing. A nice surprise for us was that almost everybody found it fun as soon as they started to perform the recording-testing loop by themselves. A lot of the fun was due to the surprise of participants in seeing and hearing themselves, through software, that they could control their gestures and that they were quickly successful in designing and reperforming their own gestures.

5.4.3. Parameter Tuning. In general, users tuned parameters quite often when defining their gestures, and when asked, *When you were not satisfied with a gesture, did you prefer to record it again or find a better parameters setting?* they all claimed that they did both. A user said: "*If the system is kind of working but not very well, I try to play with parameters, but when it doesn't work at all I preferred to re-record again.*" All other participants of that session agreed with him, and we saw this behaviour across the workshops in general.

During the first task of the workshop, when we asked participants to record only one gesture and *follow* it, initially they usually reperformed the same gesture either in the same way or slower and judged the result based on the auditory feedback and the red cursor. Often, when they tried to perform something that was too different from the prerecorded gesture, they understood that the system was not following the gesture very well by seeing the red cursorsuddenly *jumping* to different temporal positions and by receiving a noisy auditory feedback. In those cases, they were able to understand the problem quite quickly and work to find a solution, either recording the gesture again or changing the *tolerance* parameter. We gave them practical tips about this last parameter, such as *If the cursor starts going forward by itself, it means that the system is*

too tolerant or *If the cursor starts jumping too much and the audio starts becoming mad even if you are performing well, turn up the tolerance a bit.* The association between the tolerance and the thickness of the waveform was straightforward to understand for all our participants.

When we asked users to record more than one gesture to experiment with recognition, they all started recording very basic gestures, each very different from the other. This allowed them to make the system work immediately and get a real sense of its behaviour before moving on to record more complex and subtle gestures.

Using an accelerometer as an input device, some of them initially recorded gestures in which the motion in one axis was clearly predominant compared to the others, which was useful because it often enabled them to achieve a quicker and better understanding of the meaning of the waveform.

The *contrast* parameter was used mainly when the system was uncertain between two or more gestures. By increasing this parameter, they could see the likeliest gesture more clearly, referring to it as the *green gesture*, pointing out that the association between the likelihood and this specific colour was pretty clear. On the other hand, when colours started flickering too much, they knew quickly that it meant that it was a good idea to decrease the value of the contrast.

One of the more surprising results for us was that the *latency* parameter was much less used than the others. Some participants admitted that the effect of this parameter was not clear; reflecting on it at the time, we thought it was because there was no graphical feedback associated with this parameter, which provoked a fear in the user to changing the value of this parameter. In response, in the next version of GIDE, we will release a graphical feedback to this parameter—highlighting the part of the data input view that corresponds to the amount of time specified by the parameter.

5.4.4. Participant Strategies. GIDE proved to be a tool that strongly engaged users from different backgrounds and for different applications, and here we discuss some of them. Both a professional piano player and a professional dancer spent most of their time training themselves to perform their own gestures reliably by recording quite expressive gestures and using their voice as audio at the same time. They both used the *warping view* function to measure the differences between performance and the prerecorded gesture and were not satisfied until the audio output was sufficiently close to what they recorded. It was interesting to see how they played with the value of the *tolerance* parameter quite a lot. The dancer said that she had a much better concept of accelerometers after trying to perform the same gesture several times and watching at the *warping view*.

The dancer particularly liked the *testing performance* component. She said that the user of the application and the performer would often not be the same person, and this enabled her to record the performance just once and work on parameter tuning later. She also explained that the *following* mode is interesting for live situations, where the choreography and other theatrical and technical affects need to be directly synchronised with the performer and thus not requiring a human to trigger them. She also suggested a new feature that we had not thought of previously. Her idea was to build a gesture vocabulary *from* the testing performance: this means recording the testing performance before, then copying and pasting certain parts of it for defining gestures.

Computer artists generally preferred a more methodical approach, tending to think about their gestures in advance and then record one gesture after the other. They usually tested one single gesture or a fixed series of gestures, changing one parameter at time. Other contemporary music performers took a radically different strategy and recorded quite complex gestures and then they played with the system, trying to confuse

it. One user said, "I got how the system works, now I just want to hack it." In doing that, they also played a lot with the tolerance and the contrast, keeping the contrast high enough to see major differences between their gestures. This shows that participants with different artistic background developed different strategies for using GIDE that were tailored to their needs.

Finally, it is worth mentioning that working with accelerometers caused some problems. First, participants who were not familiar with this kind of device found it a lot more natural to think in terms of absolute position. Furthermore, they were surprised to see that due to hardware limitations of accelerometers, slow movement was not recognised.

6. DISCUSSION

In this article, we have described the design, implementation, and evaluation of GIDE—a novel gesture interaction system. We believe that it is sufficiently simple, responsive, and intuitive to enable users to become fully engaged, immersed, and involved with the success of their activity that a state of flow [Csikszentmihalyi 2008] is possible. We have proposed a set of four desiderata that we believe define a new approach to the design and use of the next generation of gesture interaction systems, which we call *fluid gesture interaction design*. The four qualities critical to the design of such systems are that (1) it enables target applications to be controlled continuously and synchronously over time; (2) it can be tailored and personalised according to the individual, the activity in which individuals wish to engage, and the environment and context in which that activity is taking place; (3) it provides users with meaningful feedback as to how the system is currently interpreting their gestures; and (4) it is easy to use and thus can be readily adopted and used by anyone irrespective of level of expertise and background.

This section summarises the main contributions and limitations of the proposed work and outlines future research.

6.1. Contributions

The research described in this article has demonstrated the value of our approach in several different ways. First, through a quantitative evaluation on a standard set of 2D gestures, we demonstrated that the algorithm used by GIDE can perform as well as that of DTW, which is a standard template-based algorithm in the field of gesture recognition systems. Nevertheless, unlike DTW, the algorithm can provide moment-by-moment information *during* the gesture performance itself, such as the relative speed, and a early estimation of the recognition results. The recognition efficiency of the algorithm depends on the tuning of two parameters called *tolerance* and *latency*. We have demonstrated that the dependence of these parameters to the recognition efficiency follows wide bell-shaped curves with one maximum. What this means is that whatever the initial value of the parameters, the optimal values can easily be found manually by any user through simple trial and error. The user simply tunes the parameters so as to increase recognition efficiency, and when no further increase can be found, the user can then be confident that he or she is operating with a system at maximal recognition efficiency. This ability—for users to easily tune system parameters for maximum efficiency—is an extremely important result in terms of demonstrating how the system can be used by a range of users with different expertise.

The second aspect of our work concerned an evaluation of the system through a workshop with 23 users from different performance backgrounds. Through the analysis of the questionnaires that each participant completed at various stages of the experiment and a semistructured interview, we have evidenced how our system has met each of the four desiderata for fluid gesture interaction design. We have described how quickly participants were able to understand and enjoy using the system, engaging with the

real-time gesture *recognition* and gesture *following* during the workflow of gesture design almost immediately. Almost all users taking part in the evaluation were able to control the application reliably after a few minutes and develop a set of gestures to control the system in a way that they found satisfying with system behaviour, thus meeting intention and expectation. The evaluation demonstrated how users could seamlessly move between recording new gestures, testing them, and tuning parameters, and this enabled the relationship with the system to be fluid and spontaneous. Participants adopted different working strategies, imagining a wide range of possible applications across different domains, and were extremely enthusiastic about the future potential for developing new performances in their own creative practice.

One of the main contributions of our research is that we have developed a system that enables users to *design by doing*, where gesture interfaces are created by performing gestures. Our participants' enthusiasm supports the conclusion of Fiebrink et al. [2011] that this *embodied* way of designing gestures is both liberating for users and allows for the creation of rich styles of interaction. The fact that GIDE gives real-time feedback about the recognition process in different modalities was found by participants to be extremely useful at different stages in the workshop. They claimed that the combination of (1) video to remember the details of the gesture, (2) audio for precise and instant feedback, and (3) data waveform for the ability to see an overview of the gesture over time and its temporal alignment with the prerecorded gesture was critical to their engagement with the system. This real-time feedback on the performance of gestures also helped participants understand the function of the various parameters of the algorithm and thus were able to tune these parameters to produce their desired results effectively and efficiently. We believe that this is an especially significant result, as the parameters setting has been a challenge for interactive machine learning [Fiebrink 2010]. If nonexpert users are able to effectively tune these parameters as we have described in this article, it opens the way to using more sophisticated machine learning algorithms, such as HMMs, that require considerable amount of tuning in order to be effective. Moreover, it paves the way for experimenting with how real-time feedback can become a crucial feature of future interactive machine learning research in general.

6.2. Limitations

In this article, we have attempted to demonstrate the importance, to our method, of making the process of gesture definition as quick and engaging as possible, with users recording gestures just once, in order to make this operation part of the interactive workflow. This enables the whole recognition process easier to understand, as gesture performed by users simply need to be similar to the ones they prerecorded. This is very different from the standard approach, where systems create models of gestures based on a large number of examples thath are hidden to the user and thus make the system much less transparent. However, our method has certain limitations, as it does not support generalisation, and any performance needs to be sufficiently similar to the set of prerecorded gestures. Although it is important not to lose the feature of "quick recording," adding new system functionality that enables users to record more than one example per gesture, and therefore create a more complex and flexible model of the gesture defined by the user, is very much central to our ongoing research investigation. We envisage achieving this is by allowing users to record one or more examples for each type of gesture and then building separate HMMs for each recorded example. Then, at runtime, recognition can be achieved by considering the correct type of gesture to be the one that corresponds to the HMM with the highest likelihood value. However, this would require us to design new methods for data visualisation and auditory feedback,

as well as new ways of interacting with the system, but in such a way that would still keep the workflow as fluid and intuitive as it currently is.

Another problem of the proposed approach is in handling the output during the very beginning of a gesture, when the likelihood of recognising the correct one is still low (Table I). This results in a period of uncertainty when the system is first started and also when a user transitions from making one gesture to another. For certain scenarios, this issue can be handled at the application level. As shown in the two applications described in Section 4.4, audio and video scrubbing, we can use this feature to blend between different media based on the likelihood value of their associated gestures. As GIDE provides continuous likelihood estimates for each gesture, transitions between gestures will correspond in smooth transitions between different likelihood levels until the algorithm gets a clear result. This was a reasonable solution for the applications that we described in the article.

However, for different applications where a more defined segmentation is required, further research is needed. For instance, it would be possible to add constraints to avoid users recording gestures that are too similar at the beginning. Another solution could be to add a variable latency to the system to compensate the initial time of incertitude.

6.3. Conclusions and Perspectives

Although we evaluated our system with users chosen from the performing arts domain specifically, we believe that the novel functionality offered by GIDE is important for any modern gesture interface. Our participants used a number of different design strategies and developed different forms of interface, influenced by their different artistic backgrounds. This suggests both that different design strategies will be required for different domains and that GIDE successfully supports them. Our criterion of *tailorability* is therefore likely to be increasingly important as we target more diverse domains of application. There is, however, one aspect of the current application that is still quite domain specific. We have used visualisations that will be familiar to users of digital arts production tools (e.g., waveforms) and have used metaphors drawn from these tools (e.g., our parameter names). Our evaluation demonstrated that this familiarity was particularly helpful in understanding the effect of parameters. However, it is our intention to focus future research on the design of new visualisation tools and metaphors that can be appropriate to users from any background and to any conceivable application domain. Another venue for investigation in the near future is to exploit the real-time feedback of our system in order to build a *predictive feedback system* that provides guidance by estimating the next incoming action of the user.

In conclusion, we believe that our new notion of fluid gesture interaction design provides the platform for the future widespread uptake of gesture interaction systems across a whole range of activity in the near future.

A HMM PROCEDURE

As described in Rabiner [1989], the forward procedure can be used to estimate the probability distribution of a sequence of observation $O_1, O_2, \ldots O_t$. This requires the computation of the $\alpha_i(t)$ variable that corresponds to the probability distribution of the partial observation sequence until time t and state i. It is computed inductively as follows:

Initialisation

$$\alpha_1(i) = \pi_i b_i(O_1) \qquad 1 \leq i \leq N, \tag{2}$$

where π is the initial state distribution, and b is the observation probability distribution.

Induction

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_t + 1) \qquad 1 \leq t \leq T-1, 1 \leq j \leq N, \tag{3}$$

where a_{ij} is the state transition probability distribution.

From the $\alpha_i(t)$ variable, we can compute two important quantities:

(1) Time progression of the sequence, related to the recorded example:

$$time\ progression\ index(t) = argmax[\alpha_t(i)] \tag{4}$$

This last value can be alternatively estimated by the mean (expected value) of the distribution $\alpha_i(t)$.

(2) Likelihood of the sequence:

$$likelihood(t) = \sum_{i=1}^{N} \alpha_t(i) \tag{5}$$

This quantity can been used directly as a *similarity measure* between the gesture being performed and the recorded reference. Other similarity measures could also be derived by combining the *likelihood* and the smoothness of the *time progression* index.

REFERENCES

Daniel Ashbrook and Thad Starner. 2010. MAGIC: A motion gesture design tool. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems (CHI'10)*. Elizabeth D. Mynatt, Don Schoner, Geraldine Fitzpatrick, Scott E. Hudson, W. Keith Edwards, and Tom Rodden (Eds.). ACM, 2159–2168.

Frederic Bevilacqua, Florence Baschet, and Serge Lemouton. 2012. The Augmented string quartet: Experiments and gesture following. *Journal of New Music Research* 41, 1 (2012), 103–119.

Frederic Bevilacqua, Fabrice Guédy, Norbert Schnell, Emmanuel Fléty, and Nicolas Leroy. 2007. Wireless sensor interface and gesture-follower for music pedagogy. In *Proceedings of the International Conference on New interfaces for Musical Expression (NIME'07)*. New York, NY, 124–129.

Frederic Bevilacqua, Norbert Schnell, Nicolas Rasamimanana, Bruno Zamborlin, and Fabrice Guedy. 2010a. Online gesture analysis and control of audio processing. In *Musical Robots and Interactive Multimodal Systems*. Springer Tracts in Advanced Robotics.

F. Bevilacqua, Bruno Zamborlin, Anthony Sypniewski, Norbert Schnell, F. Guédy, and Nicolas Rasamimanana. 2010b. Continuous realtime gesture following and recognition. In *Proceedings of the 8th International Gesture Workshop*. 73–84.

Richard Bowden, David Windridge, Timor Kadir, Andrew Zisserman, and Michael Brady. 2004. *A Linguistic Feature Vector for the Visual Interpretation of Sign Language*. Springer-Verlag, 391–401.

Richard Bowden, Andrew Zisserman, Timor Kadir, and Mike Brady. 2003. Vision based Interpretation of Natural Sign Languages. In *Proceedings of the 3rd International Conference on Computer Vision Systems*. ACM, 391–401.

Xiang Cao and Ravin Balakrishnan. 2003. VisionWand: Interaction techniques for large displays using a passive wand tracked in 3D. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology (UIST'03)*. ACM, New York, NY, 173–182.

M. Csikszentmihalyi. 2008. *Flow: The Psychology of Optimal Experience*. HarperCollins.

Y. Dai. 2001. An associate memory model of facial expressions and its applications in facial expression recognition of patients on bed. In *Proceedings of the IEEE International Conference Multimedia Expo*. 772–775.

Jerry Fails and Dan Olsen. 2003a. A design tool for camera-based interaction. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI'03)*. 449.

Jerry Alan Fails and Dan R. Olsen Jr. 2003b. Interactive machine learning. In *Proceedings of the 8th International Conference on Intelligent User Interfaces (IUI'03)*. ACM, New York, NY, 39–45.

Rebecca Fiebrink. 2010. Real-time interaction with supervised learning. In *Proceedings of the 28th of the International Conference Extended Abstracts on Human Factors in Computing Systems (CHIEA'10)*. 2935.

Rebecca Fiebrink, Perry R. Cook, and Daniel Trueman. 2011. Human model evaluation in interactive supervised learning. *Machine Learning* (2011), 147–156.

François Guimbretière, Maureen Stone, and Terry Winograd. 2001. Fluid interaction with high-resolution wall-size displays. In *Proceedings of the 14th Annual ACM Ssymposium on User Interface Software and Technology (UIST'01)*. ACM, New York, NY, 21–30.

Björn Hartmann, Leith Abdulla, Manas Mittal, and Scott R. Klemmer. 2007. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'07)*. 145.

Ken Hinckley, Gonzalo Ramos, Francois Guimbretiere, Patrick Baudisch, and Marc Smith. 2004. Stitching: Pen gestures that span multiple displays. In *Proceedings of the International Working Conference on ACM Advanced Visual Interfaces (AVI'04)*.

Caroline Hummels, Kees C. J. Overbeeke, and Sietske Klooster. 2006. Move to get moved: A search for methods, tools and knowledge to design for expressive and rich movement-based interaction. *Personal and Ubiquitous Computing* 11, 8 (Nov. 2006), 677–690.

Amy K. Karlson and Benjamin B. Bederson. 2005. Applens and launchtile: Two designs for one-handed thumb use on small devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'05)*. ACM, 201–210.

Sven Kratz and Rafael Ballagas. 2009. Unravelling seams: Improving mobile gesture recognition with visual feedback techniques. *Techniques* (2009), 937–940.

Yang Li, Jaime Ruiz, and Edward Lank. 2011. User-defined motion gestures for mobile interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 197–206.

J. Linjama, P. Korpipää, Juha Kela, and T. Rantakokko. 2008. ActionCube: A tangible mobile gesture interaction tutorial. In *Proceedings of the 2nd International Conference on Tangible and Embedded Interaction*. ACM, 169–172.

Hao Lü and Yang Li. 2012. Gesture coder: A tool for programming multi-touch gestures by demonstration. In *Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems (CHI'12)*. ACM, New York, NY, 2875–2884.

S. Mitra and T. Acharya. 2007. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 37, 3 (2007), 311–324.

Meredith Ringel Morris, Anqi Huang, Andreas Paepcke, and Terry Winograd. 2006. Cooperative gestures: Multi-user gestural interactions for co-located groupware. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*. ACM, 1201–1210.

Otniel Portillo-Rodriguez, Oscar Osvaldo Sandoval-Gonzalez, Emanuele Ruffaldi, Rosario Leonardi, Carlo Alberto Avizzano, and Massimo Bergamasco. 2008. Real-time gesture recognition, evaluation and feed-forward correction of a multimodal tai-chi platform. In *Proceedings of the 3rd International Workshop on Haptic and Audio Interaction Design (HAID'08)*. Springer, 30–39.

L. R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* 77, 2 (1989), 257–286.

Norbert Schnell, Axel Röbel, Diemo Schwarz, Geoffroy Peeters, and Riccardo Borghesi. 2009. MuBu and friends: Assembling tools for content based real-time interactive audio processing in Max/MSP. In *Proceedings of the International Computer Music Conference (ICMC'09)*.

P. Turaga, R. Chellappa, VS Subrahmanian, and O. Udrea. 2008. Machine recognition of human activities: A survey. *IEEE Transactions on Circuits and Systems for Video Technology* 18, 11 (2008), 1473–1488.

Y. Visell and J. Cooperstock. 2007. Enabling gestural interaction by means of tracking dynamical systems models and assistive feedback. In *Proceedings of the IEEE International Conference oSystems, Man and Cybernetics (ISIC'07)*. 3373–3378.

M. M. Wanderley and P. Depalle. 2004. Gestural control of sound synthesis. *Proc. IEEE* 92, 4 (April 2004), 632–644.

David Wessel and Matthew Wright. 2002. Problems and prospects for intimate musical control of computers. *Computer Music Journal* 26, 3 (Sept. 2002), 11–22.

Tracy Westeyn, Helene Brashear, Amin Atrash, and Thad Starner. 2003. Georgia Tech gesture toolkit: Supporting experiments in gesture recognition. In *Proceedings of the 5th International Conference on Multimodal Interfaces*. ACM, 85–92.

John Williamson. 2006. *Continuous Uncertain Interaction*. PhD dissertation, University of Glasgow, Scotland.

A. D. Wilson and A. F. Bobick. 1999. Parametric Hidden Markov Models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21, 9 (1999), 884–900.

I.H. Witten and E. Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. Vol. 54. Morgan Kaufmann.

Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. 2009. User-defined gestures for surface computing. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems (CHI'09)*. 1083.

Matthew Wright, Adrian Freed, and Ali Momeni. 2003. OpenSound control: State of the art 2003. In *Proceedings of the 2003 Conference on New Interfaces for Musical Expression (NIME'03)*. 153–160.

Received March 2012; revised September 2012; accepted June 2013