Comprehensive Model and Image-Based Recognition of Hand Gestures for Interaction in 3D Environments

J. Bernardes[1], R. Nakamura[2] and R. Tori[2,3]

[1] Escola de Artes, Ciências e Humanidades, Av. Arlindo Bétio, 1000, São Paulo, SP, Brazil, 03828-000
[2] Escola Politécnica da USP, Av. Prof. Luciano Gualberto, travessa 3, 158, sala C2-47, São Paulo, SP, Brazil, 05508-900
[3] Centro Universitário SENAC, Av. Engenheiro Eusébio Stevaux, 823, São Paulo, SP, Brazil, 04696-000

***Abstract* — Interest in gesture-based interaction has been growing considerably, but most systems still limit the recognition of hand gestures to a small set of signs. We present a model for hand gestures that allows the definition of thousands of distinct signals based on the combination of a much smaller number of gesture components. This model comprehends several different kinds of gestures, both static and dynamic and using either hand or both. The choice of types of gestures and individual components is based not only on a review of the relevant literature but also on preliminary user studies, specifically for interaction in virtual and augmented environments and in entertainment and education applications. Gesture recognition based on this model is implemented as a finite state machine that incorporates the results of algorithms for the classification of each component, but is itself independent of those algorithms. The paper also describes an unencumbered gesture recognition system built using this model and recognition strategy, a single low-cost camera and relatively simple image-based algorithms to classify hand poses, movements and location and for segmentation. Tests show the model allowed the definition of the desired gestures for three target applications, a commercial computer game and two educational 3D environments. Our system was able to recognize these user gestures, and several others, in real-time. We could also perceive a need for the recognition of incomplete gestures and for a more robust segmentation strategy.**

***Index Terms* — Digital Games; Education; Human-Computer Interaction; Image-Based Gesture Recognition; Virtual and Augmented Reality;**

I. INTRODUCTION

The main purpose and contribution of our work is the development and implementation of a model of hand gestures that allows the definition of a large number of them. The model comprehends static, dynamic, single-hand and bimanual gestures. Our gesture recognition system must also leave user hands free of any gloves, sensors, cables or markers and will, therefore, be based on captured images. This large number of gestures and the need of free hands are two of the most important system requirements, all derived from a study of interaction in virtual and augmented environments and, more specifically, in entertainment and education applications. The model and the recognition system are called Gestures2Go.

For decades, almost since the beginning of the work in human-computer interfaces, the possibility of interacting with a computer through a user's own gestures, hands and voice has drawn interest from researchers [1]. It has also captured the public's imagination and been portrayed in numerous works of fiction. One of the reasons for this attraction, even though conventional interfaces based on keyboard and mouse have proven adequate for several tasks, lies in the fact that these forms of interaction are more similar to the way in which users interact with other people and objects in their day-to-day lives. They thus make use of already well-developed skills.

In virtual and augmented environments, particularly, conventional interfaces often prove less adequate for many tasks than they do for 2D applications in front of a computer. The use of these devices with only one or two spatial degrees of freedom to perform tasks in 3D not only often requires complex interaction techniques, but may also reduce user immersion and mobility in these environments [1].

It is no surprise, then, that the use of gestures for interaction has long been researched in the context of virtual reality. At first, hand poses and positions were most often captured with datagloves. The use of these devices, however, would usually cause increased costs, require complex calibration procedures and restrict some hand and user movements [2]. Furthermore, the gloves might not fit all users and may cause allergy problems or even hygiene problems, especially when shared among multiple users. These factors may have contributed to the relatively small proliferation of gestural interfaces using datagloves as the main input device outside research institutions and some specific applications.

In the past years, however, greater processing power is being delivered in multiple cores. Low-cost image capture devices have also become widely available even for home platforms and appliances, with standard interfaces (i.e. USB) and frame rates fast enough for use in interaction. These factors have led to a renewed and broader interest in gesture-based interfaces using cameras as sensors, instead of datagloves. Even though these computer vision-based approaches are still unable to deliver the same precision in hand configuration as those gloves, at least in real-time [2], this precision is not required for a wide range of applications. It is now possible to find a variety of products and applications incorporating some simple forms of gestural interaction including, for instance, television sets, notebook computers, smart phones and even simpler applications such as web-based casual games. There is even a

Manuscript Received on September 30, 2011.
E-mail: jlbernardes@usp.br

recent industry-led attempt to promote interoperability between natural interaction devices, middleware and applications [3]. At least half of the founding members in this organization are concerned with vision-based interaction and with digital games as an application.

The digital games industry, in particular, has been investing considerably in natural interaction. Of the three major videogame console manufacturers, one used gestural and pointing interaction through a handheld wireless controller as a major and successful selling point. A second manufacturer included, since the last generation of its console, a camera as an optional input device, releasing a few games using it to capture user movements and positioning as input. More recently a wireless controller with a luminous element was paired with this camera allowing more varied and precise interaction. The third manufacturer has been making major investments in interaction based primarily on body gestures, also including voice commands and face recognition and using no controllers or sensors held by or attached to the users. A peripheral integrating a camera, a depth-sensing device and a microphone is used instead. Many of the games in all these platforms are, in addition, set in 3D environments.

None of these systems, however, currently recognize a wider breadth of hand gestures, including various poses and different kinds of gestures. Hand gestures may be considered of particular interest to interaction because the human hands have a large number of degrees of freedom that users can explore with ease and are, therefore, a very flexible tool for interaction [1]. Many of these systems also use proprietary software or hardware devices. Academic solutions for image-based hand gesture recognition exist, but they are often difficult to use for interaction designers, demanding knowledge of areas such as image processing, machine learning and pattern recognition [4]. There are also many such solutions to the specific problem of sign language recognition which cannot be easily adapted for use in interaction, since the latter often lacks the complex, context-based grammar rules that may aid in the recognition of signs in language. Finally, but most importantly, many of these solutions recognize only a small set of gestures, usually limited to only one or two kinds of gesture, for instance, only static gestures or only taking in account hand trajectory, as discussed in the related work section.

This is the context in which our work is set. As afore-mentioned, its purpose is to provide a gesture model capable of describing a large variety of gestures for interaction in virtual and augmented environments, specifically for entertainment and education applications, and to develop a system to recognize these gestures.

Entertainment, particularly digital games, was chosen as a target class of applications for several reasons. Games were previously shown to be an ideal platform for testing and disseminating novel interaction devices and techniques [5] and have indeed been responsible for the popularization of several of them. These devices and techniques may then be profitably used in other areas, including education [6]. In a panel about game interfaces in SIGGRAPH 2005 [7], several researchers and industry representatives agreed that the conventional forms of interaction in games constitute one of the main barriers of entry to new users, and that new and more natural forms of interaction, such as the use of gestures, would be important not only to attract new users but also to improve interaction for the existing ones, since it has been shown that gestural interfaces in games may indeed increase user subjective satisfaction [8].

An advantage of choosing education as the other class of applications is that user interfaces for entertainment and education share several requirements, favoring (instead of efficiency or precision) ease of learning and remembering the interface, ease of use, not requiring complex preparation or precise movements, system cost and user subjective satisfaction [9]. Additionally education, as entertainment, has traditionally been an important application for both virtual [10] and augmented [11] environments due to their capacity to render and allow a more natural interaction with abstract concepts, distant or unsafe objects or situations, objects in atomic or interplanetary scale etc.

Being able to determine the proper requirements for the gesture model and the recognition system was indeed the main reason to limit the work scope to these two classes of application in 3D environments. And the ease of learning, remembering and using the gestural interface are some of the most important of these requirements. Gestures, however, as keyboard or voice commands, usually constitute an "invisible" form of interaction [1], meaning that the interface does not constantly display its options to the user, as happens with menus and graphical interfaces. It is therefore necessary to find ways to facilitate learning and memorizing which gestures do what in the interface. For this reason, several authors [1][9][12] agree that the gestures chosen for interaction must have some clear logical association with the action it triggers or commands in the interface, and must preferably be chosen from a set of existing gestures in the application domain. Entertainment and education applications, however, may address a large variety of different disciplines and domains and it is, therefore, not feasible to determine a minimal set of gestures that might be adequate to all of them. What must be done instead to facilitate gestural interaction for these applications is to allow the definition and recognition of the largest set of gestures possible, including different kinds of gestures. This increases the probability of finding an appropriate gesture for a given action in each application interface, i.e. a gesture that makes sense for that action due to some logical association or domain or cultural convention. This requirement leads to this work's main feature and contribution, a model that allows the definition of many thousands of gestures and a way to recognize them. Gestures2Go achieves this by modeling complex gestures as a composition of simpler basic components, a strategy similar to the statistical and linguistic approach, based on phonetic components, often used in speech recognition [13]. Combinations of relatively small numbers of these components result in a very large set of possible signals.

Other requirements for Gestures2Go may also be gleaned from this discussion and from other authors. For ease of use, the recognition must tolerate small variations in gesture performance [1][9][12]. The calibration and preparation time before actually using the system must be minimal [14]. The system should use widely available and preferably low-cost equipment. No gloves, sensors, markers or cables should be attached to or held by the user: cameras should be used as the only input device. As in other camera-based interfaces, it is important to allow visual feedback to the users because it is

difficult for them to infer their position in relation to the device's field of view [15]. While it is possible to handle input delays of the order of one second using proper interactions techniques such as those used in massively multiplayer games to handle network lag [16], the system should be able to recognize gestures in real-time, a rate of at least ten times per second [17], and should additionally demand as little processing time and resources as possible, freeing these resources to other tasks common in 3D environments such as rendering and simulation. Finally, using the system as an interaction designer should not require knowledge of areas such as image processing, machine learning, pattern recognition etc.

II. RELATED WORK

The literature about gesture recognition is vast [2][18][19] since it is a topic of intense interest for researchers in interaction and in virtual and augmented reality. It constitutes a complex problem that cannot be considered well-solved, particularly when using image capture devices as the only sensor. Several different solutions, therefore, co-exist. In this vast literature, however, it is rare to find works that model gestures as a combination of simpler components that may be recognized separately and then combined to achieve a large set of possible signals.

One of these systems [20] uses a depth-sensing camera and models gestures as a combination of 12 basic movements and 7 one-hand poses, plus 1 pose with both hands touching. The authors report testing the system with 20 gestures, but discuss how combining these elements may result in over a hundred possible gestures.

The Open Gesture Recognition Engine (OGRE) [21] uses different strategies to recognize small sets of static gestures and dynamic gestures defined simply by hand trajectories, but also models a third kind of gesture, called "staged paths" by the authors, which is composed of a series of hand poses linked by linear movements, which gives the gesture model some extra flexibility and breadth. An interesting feature of OGRE, not common in other works, is that it explicitly allows the definition of subsets of gestures to be recognized at a given moment, reducing cost and errors in the recognition. Since, for interaction, the user is unlikely to need or even be able to remember a large set of gestures, this is a valid strategy. The large set of possible gestures, however, should still exist, to allow the choice of the best subset in each interaction context.

Another engine for gestural interaction [22] defines gestures as either static or dynamic based only on hand trajectory, but not a combination of both.

Finally, an earlier work [13] already proposes modeling a gesture as a combination of simpler components (hand pose, movement and location relative to the body, components based on an analysis of sign languages), similarly to the statistical and linguistic approach based on phonetic components used in speech recognition. That work discusses how this approach allows the definition of a large number of gestures based on a much smaller number of components, but it does not describe how to implement the recognition based on this model and only relates the implementation and testing of dynamic gestures based on hand trajectories.

Much more common in the related literature, particularly for sign language recognition, are those that model complex gestures "all at once", without decomposing the gesture or attempting to recognize its components separately. In this case, gestures are usually modeled as a sequence of temporal states. Hidden Markov Models (HMMs) are popular tools to model them in such way [18], but several other tools are also popular. These models are, obviously, not well-suited to recognize static gestures. Even for dynamic gestures that differ mostly by hand pose, instead of movement, they often prove inadequate, due to using simplified pose representations to reduce the size of the state vector. Finally, these systems are often limited to a number of gestures of the order of 50 to 100 [18][19], due to the increasing costs of training and recognition with larger sets. While it should be possible to recognize, at each moment, only a subset from a larger number of possibilities, these works usually do not discuss this possibility or relate tests regarding it. The Gesture and Activity Recognition Toolkit [23] exemplifies this approach and offers tools to simplify the use of HMMs to recognize such patterns by non-specialists, but it still demands some basic knowledge of HMMs.

Still more common is the modeling and recognition of only a limited number of static gestures. Many different strategies are used for this. HandVu [24] is an example that identifies six distinct poses of one hand and robustly tracks that hand even in cluttered and moving backgrounds. It is also another tool which is quite simple to use, not requiring specialized knowledge.

III. THE GESTURES2GO MODEL

Gestures2Go models gestures as a combination of three simpler components. These components, chosen based on an analysis of sign languages [25], are hand pose, the movement described by its center and the location relative to the user's body where the gesture is initiated. The pose is further subdivided into hand configuration and orientation.

A gesture may be composed of more than a single pose. Indeed, changing between poses, regardless of whether the center of the hand moves or not, is a common way to perform dynamic gestures. When the hand is moving, however, changes in hand pose rarely add meaning to the gesture [26]. The model, therefore, ignores changes in hand pose during movements and establishes that a gesture is composed of only two poses: an initial and a final one. These poses are further used in the model as a simple way to segment gestures in time, especially because poses must be maintained for a set and configurable amount of time to be accepted as such [27]. While this is not always valid for human gestural interaction, particularly for gestures used along with speech in natural communication [28], it is a valid strategy for issuing commands and manipulating objects in 3D environments, some of the most common tasks for gestures [1].

In this model therefore, for a single hand, a gesture definition must necessarily include one initial and one final hand pose. These two poses may be the same, allowing the definition of static gestures. Additionally, a gesture definition may optionally specify one of 12 basic movements that the center of the hand must perform between the initial and final poses, or no

movement, in which case the hand should not move further than a configurable tolerance from its initial position during the gesture. Hand movement, unlike the initial and final poses, may be left undefined, in which case movement recognition is ignored by the model and users are free to move their hands as they wish, or not at all, while performing a gesture. This feature can be rather useful when taking advantage of gesture parameters such as hand position for interaction. Finally a gesture definition may include, also optionally, one of 12 initial locations relative to the user's body. A final detail in the model is that, while the definition of initial and final poses is mandatory, only the hand configuration needs to be defined. Hand orientation may be left undefined to be used as a gesture parameter, in cases in which its determination may be difficult (such as a closed fist) or simply when it is not important. This definition is shown in a more concise and clear manner below, using extended BNF notation [29].

oneHandGesture = handConfig, [orientation], [location], [movement], handConfig, [orientation];

While this definition is valid for gestures with one hand, Gestures2Go also allows the use of bimanual gestures. The model, however, does not support any synchronization between the two hands beyond the fact that, for a bimanual gesture to occur, one hand's initial pose must be recognized before the other's final pose is. A gesture definition, then, contains the definition of an *oneHandGesture* for one of the hands and, optionally, a second one for the other, describing a simple bimanual gesture.

gesture = oneHandGesture, [oneHandGesture];

The twelve basic movements and initial locations for gestures proposed as part of this model are discussed further in the Components subsection, after a discussion of the preliminary user studies that aided in their definition. They may be considered a basic contribution of this work. The Components subsection also discusses the discretization of hand orientation and the choice of hand configurations, thus defining all the terminal symbols for the BNF expressions above (these definitions are not shown in BNF, however, since they would be quite simple and redundant with the discussion in that subsection).

In this way, the model is capable of handling several different kinds of gestures. Static gestures, in which a hand pose is maintained for a given period of time, can be defined by choosing that pose as both initial and final and either defining the movement as "none" or leaving it undefined. Dynamic gestures without a pose change, in which the hand trajectory is the most important element, are defined in a similar way but choosing one of the basic movements instead of none or undefined. Gestures defined by a change in hand pose are defined by choosing different initial and final poses and may or may not have a movement component. Curiously, the latter kind of gesture is discussed much less frequently in the literature than the first two, despite being used frequently in communication. Combining the components in this manner also means that, even for gestures with a single hand, less than a dozen of each of these components are enough to define over a hundred thousand gestures.

Despite being capable of defining such a large number of gestures, Gestures2Go is actually a simple model and its gestures can be recognized with a relatively simple implementation. This simplicity, however, does have its drawbacks, as the model has its limitations. The lack of mechanisms to define the timing of actions performed by each hand in bimanual gestures has already been mentioned. This can be ameliorated by using time as a gesture parameter, but it might not be simple. The model also limits gesture definitions to two poses (and their orientations), thus it cannot model repetitive gestures, such as waving or mimicking a sawing movement, which are commonly used in several domains. These actions may, instead, be recognized as a sequence of simpler and similar gestures.

3.1 Preliminary User Studies

The requirement for all these possible gestures of all these different kinds was derived not only from a detailed analysis of the relevant literature, which is barely discussed here but also from some preliminary user studies. While Gestures2Go was still in the research and planning stages, these studies were conducted to surmise which gestures and kinds of gestures users freely attached meaning to and wished to use for interaction in specific 3D environments. The main goal for these tests was determining functional characteristics of the model, so they were performed by a reduced number of expert users, most with knowledge of 3D interaction. Seven users took part in these tests, four male and three female with ages varying from young adults to middle-aged. Most of these users fit in more than one of these categories, which are of particular interest to our target applications: teachers, students, players of digital games or developers/researchers of 3D applications. Only two of these users, in fact, had little to no experience with augmented or virtual environments, including games. Four users were right-handed and three left-handed.

In the first study we asked users to perform a gesture with their hands in front of a camera and name a meaning for that gesture, repeating this process for as many gestures as they could think of. In this test, four users named 62 distinct gestures and used 26 different hand poses. Several gestures and meaning pairs were repeated by more than one user despite their isolation during testing. 11 gestures were bimanual, but the large majority were performed with only one hand (curiously, one of the left-handed users performed most gestures with the right hand and could not later explain why). All of the bimanual gestures required specific hand poses and movements of one or both hands. Of the gestures with only one hand, however, in this particular context most (35) were static gestures, followed by gestures in which the movement was the most meaningful component (14) and gestures with changes in pose or orientation but no movement (11). Only 2 gestures were used for which both movement and changes in pose of a single hand were considered important for meaning. Only 5 gestures required a specific location for their meaning. In only 1 gesture of this set a hand crossed from one side of the body to another.

The second study was based on two applications presented to the users. We asked them to suggest gestures that they would like to perform to trigger certain actions in each one. The first application is a virtual environment to explore the subject of lighting in computer graphics courses. User actions include turning on and off several light sources of different kinds, including closing a window to "turn off" a directional source.

They also include picking the rendering algorithm used in the scene (flat, Gouraud or Phong). The second application was City of Heroes[1], a massively multiplayer 3D game in which the user's avatar has a number of "super-powers". Users were shown these super-powers for a specific character in the game and were asked to pick gestures to activate these powers. In the study with the lighting environment, a tendency was noticed to attach symbolic representations (in this case, numbers represented by simple hand poses) not only to abstract elements such as the rendering algorithms, but also to objects present in the scene, such as the light sources. In City of Heroes, because the use of each power triggers a very distinct character animation, all users immediately chose to mimic the hand movements performed by the character during these animations in their gesture suggestions. When animations were too similar, the use of different hand poses was suggested. Here the second and last gesture in which hands crossed from one side of the body to the other was suggested. In both of these contexts, bimanual gestures and fixed initial locations for the gestures were much more common than in the first study, but while all gestures suggested for City of Heroes involved movement, in the illumination environment only a few did (particularly for closing the window).

Gestures for a third application were also studied in this phase. VIDA is a virtual didactic atlas of anatomy that allows the visualization of anatomical structures with augmented reality and their manipulation with gestures. This was the only application in which gestures should be used primarily for object manipulation instead of for issuing commands. The study [30] showed how important it is, for this kind of interaction, to allow the use of continuous gesture parameters, such as hand position and orientation, not only after a gesture has been recognized but even during the recognition process. This observation, however, has little effect on the model itself and more on the details of how the recognition of its gestures is implemented. The model does affect which parameters are readily available, though. Since hand orientation and the movement of its center are explicitly used in the model, hand position and orientation can be readily obtained, as can some measure of time, while other parameters, such as velocities or accelerations, must be derived from them. This was also the only study in which a camera was never shown as reference to the users while they suggested gestures.

Finally, users were asked to perform a series of movements, tracing letters, numerals and simple geometric shapes in the air with their hands, as well as simply moving the hand linearly up and down, left and right etc. All users considered the linear movements easy to perform, classified some movements composed of two linear ones (such as the numerals 1 and 7 and the letters L and V) as well as circular movements as being of medium difficulty and complained about all other movements being too difficult to perform with their hand in the air, forcing them to make ample movements with the arm extended away from the body for long periods of time. These more complex movements were also unanimously criticized as being "too slow". In this study, users were also asked to rotate their hands around the wrist while maintaining the hand in some configuration and comment on it.

In the particular context of these preliminary studies alone, including only three applications, users suggested or performed a total of 164 distinct gestures. These tests indicated how important it is to have access to a large number of gestures, and to different kinds of them, showing how different applications can make more or less use of each kind. They also served to determine which components should be part of the model or, in other words, what should the terminal symbols be in the simple grammar shown with extended BNF in the previous section.

3.2 Components

Fig. 1 shows the basic components chosen for the model. All basic movements and locations, but only some of the main poses (with standard orientation), are depicted. It also illustrates the only 12 basic movements that were chosen as part of the model: back, forward, up, down, left, right, four diagonal movements and clockwise and counter-clockwise circular movements. Even though a larger number of movements could easily be handled, the preliminary user studies performed earlier showed that more complex movements were more tiring, slower and displeased users. This result was somewhat surprising (despite works recommending, for ergonomic reasons, to minimize arm extension and maintenance away from the body in gestural interfaces [12]), since these more complex gestures are commonly used in touch or pen-based interfaces. Performing them with one hand in the air, however, proved to differ considerably from using one or more fingertips or a pen over a well-defined surface near the user. Even in touch-based interfaces, simpler movements are preferred and more complex movements can be made up of combinations of these basic ones [31].

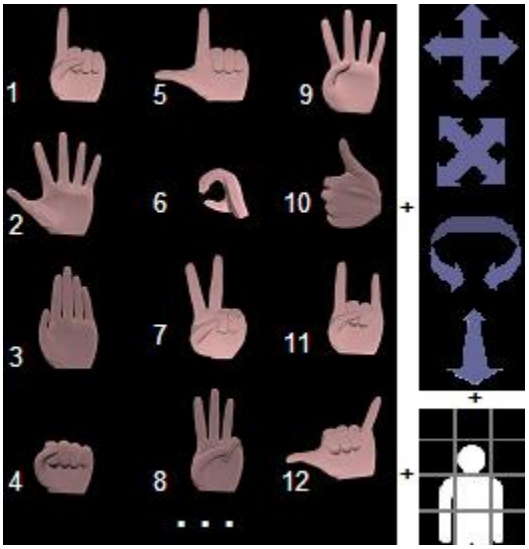

Fig. 1. Gesture components with pose samples.

Locations relative to the user's body also number 12 in the model. They are obtained simply by discretizing the space around the user's body in four regions based on hand height (above the head, head high, chest high and below the chest) and further in three regions based on the hand's lateral position (right, left or in front of the body), totaling 12 regions which are

[1] City of Heroes and its game content and materials are trademarks and copyrights of NCsoft Corporation and its licensors and were used with permission. All rights reserved.

also illustrated in fig. 1. More lateral divisions (for instance, to the right and close to the body or right and away) were also considered, but after the user studies and recommendations from other works [12], we decided to discourage gestures with the arms extended far from the body, so only these three lateral divisions remained. Given the fact that less than 2% of all gestures performed during the user studies crossed from one side of the body to another, even the need for any lateral divisions at all may be questioned, but the decision was to keep at least this degree of flexibility in the model at this stage, especially considering the small number of users evaluated in the preliminary tests.

Unlike all other gesture components, which were defined in a fixed number in the model, hand poses were not. During the user studies, a relatively high number of poses was used (26) and even afterwards users were coming up with new ones, often particular to a specific domain, such as "Vulcan live long and prosper" and "Spiderman's web shooting" poses, not to mention obscene gestures. We felt that attempting to determine a closed set of poses that would be adequate for a large number of domains, applications and cultures would be overcomplicated and unnecessary, or even unwise. Instead, the model allows the addition of new elements to the pose set and the configuration of which subset of poses should be used at each moment. That is represented in Fig. 1 by the ellipsis punctuation mark below the poses.

A standard set of poses was defined, however, comprising those numbered between 1 and 9 in Fig. 1. This choice was based on the following criteria. In a formal study with 20 users choosing gestures freely to perform 36 actions in a 2D tabletop desktop environment [32], poses 1 to 7 where the ones most often adopted by users (the picture actually depicts a variation for a pinching pose in number 6). These poses were also the most commonly used during the preliminary user studies described earlier. While in the desktop environment pose 1 was used much more frequently than the others, possibly due to an association with the mouse pointer so common in the desktop metaphor, in the preliminary studies with free gesture associations and their use in 3D environments poses 1 to 4 were all used most often and with approximately the same frequency. In these studies poses 8 and 9 were also used frequently and associated with numerals. Poses 10 to 12 shown in Fig. 1 are simply examples of additional domain-specific poses that may be added to the set. While pose 10 is commonly used with a positive or confirmation meaning in many cultural contexts, 11 and 12 are only common in particular domains, such as rock-and-roll and surfing.

The orientation portion of the poses is simply expressed as angular values relative to an arbitrarily chosen standard orientation. The tests with users, however, indicated that they had difficulty in differentiating between orientations in intervals smaller than $\pm 15°$. Orientations were, then, discretized in $30°$ intervals, which lead to 12 possible orientations around each direction. It should be noted, though, that only three or four of these orientations could be reached comfortably and were used spontaneously during the preliminary user studies. The exception to this were the tests with VIDA, in which orientations were used as a continuous parameter for object manipulation.

All these standard poses and the 12 basic movements are also commonly used in sign languages. Location relative to the body is also very relevant in this case, but the space is usually discretized in a different manner, based on features of the user's body [25].

3.3 Implementation

Recognition of gestures based on this model can be performed by a relatively simple automaton or finite state machine. This is suggested by the simple grammar rules shown previously using extended BNF, and the fact that specific poses, maintained for a small interval of time, are used to segment each gesture in time. This temporal segmentation suggests a few states, such as searching for the initial pose (in the initial location, if it has been defined), maintaining it, searching for the final pose once the initial has been confirmed and, finally, maintaining it.

After the confirmation of the final pose and once movement stops, the movement of the center of the hand for this gesture can be considered complete and may be classified.

If the initial pose (found comparing the one performed by the user with all the possible initial poses for the gestures currently defined), the final pose (similarly found), the initial location and the classified movement match one of the subset of possible gestures defined as recognizable at the moment, that gesture is recognized. If, at any point, a pose or movement is found that does not correspond to any of the currently defined gestures, the system returns to the initial state. Fig. 2 illustrates these rules.

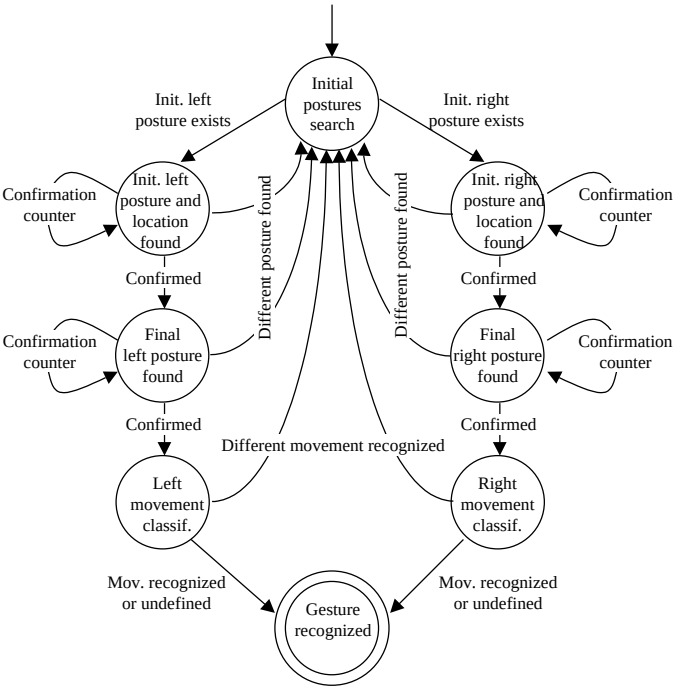

Fig. 2. State machine for recognition of gestures defined by the model.

As discussed before, initial location, movement and even the orientation of poses may be optionally left undefined and, in this case, do not affect the recognition. Likewise, for bimanual gestures parameters for both hands must be defined but leaving one hand's portion undefined (indicated by an undefined initial pose) allows the description of a gesture performed only by the other. In this case only the corresponding branch in the state

machine shown in Fig. 2 moves forward whereas for bimanual gestures both branches perform the recognition process in parallel.

The model and this strategy for the recognition of its gestures are independent of how the data about the hands is acquired or how each component is classified. They can, therefore, be applied to a broader range of problems. But aside from the requirement of allowing the recognition of gestures chosen from a large number of possibilities, which they fulfill more than adequately, this model and recognition strategy alone are not sufficient to satisfy the requirements discussed in the introduction, and must instead be used within a more complete recognition system.

IV. THE GESTURES2GO RECOGNITION SYSTEM

Based on the requirements discussed earlier, it was decided that this particular implementation of a recognition system using the Gestures2Go model would acquire data about the hands using a simple, low-cost camera without infra-red or depth-sensing capabilities, satisfying the cost and availability requirements. While depth-sensing cameras have recently become widely available, this was not true during most of the development of our work. And while available, these cameras still have a much higher cost than a simple webcam. Furthermore, a single camera is used. While a pair of them obtaining stereographic pairs of images would still satisfy the cost and availability requirements, its use often demands complex calibration procedures, violating another one. This decision, however, left the system with no direct depth measures, and this required a few simplifications to the model.

First, and most importantly, pose orientation is defined only by rotations in the image plane. Small rotations in other planes are simply ignored as small variations in the gesture, while larger changes in orientation in other planes may actually be recognized as different poses. This may also make the use of deitic gestures somewhat awkward, allowing the user to point up, down, left, right or anywhere in that plane, but not forward. The hand position parameter is also given in only two coordinates, and the back and forward basic movements must be inferred indirectly.

As mentioned before, the system allows the definition of which subset of all the possible gestures can be recognized at each moment, in runtime. This may not only speed up the recognition process, but also reduce errors, eliminating a number of similar gestures from the process. Out of the thousands of possible gestures that can be recognized based on the small number of components discussed above, for interaction the user should only need to learn, memorize and use between 5 and 10 (or 7±2 [33]) gestures in each interaction context, or maybe twice as many when the gestures can be easily associated to the actions they command.

Finally, the camera and a single user are assumed to be facing each other and, for this implementation, only the height is used for hand locations, ignoring hand lateral positions. Given the low frequency with which the hands crossed the body in the preliminary user studies, this limitation is not considered so severe. But it allows the simplification of image segmentation and analysis by assuming that each hand will remain on its side of the body. Indeed, while the segmentation of objects of interest in the image was necessary, it was not the focus or the main contribution of our work so far. The algorithms and strategies used for segmentation and for the recognition of each component in this implementation were the simplest possible (which actually, for pose and movement recognition, made them quite fast, a positive consequence of this decision). While some of these strategies and algorithms might actually be considered original due to implementation details, they are mostly derivative work and will be described only briefly. Improving upon some of them is one of the priorities for future work, even though the system performed adequately during testing.

4.1 Image-Based Recognition

To classify hand trajectories as one of the twelve basic movements described above we used a very simple approach. After the initial pose for a hand is confirmed in the state machine shown in Fig. 2, it starts to send the positions of the center of that hand, as well as its area, to the movement classifier. Hand area is used to infer movements along the camera's visualization direction. For this set of gestures, most in 2D, this classifier does not need to store the entire hand trajectory, but only the maximum and minimum values for a pair of coordinates in the image plane and for the areas. Based on these values, a simple decision tree with three levels is used to classify the gestures.

At the first level, the tree classifies the gesture as either in depth, vertical, horizontal or both, based on which coordinate showed a greater variation compared to the others. Gestures with movements in depth must have the same initial and final poses to simplify this classification, and are given priority in it, since they are usually accompanied by large movements in the image plane as well, due to the angle between the camera and the user. Using a low cost depth sensor, instead of a simple webcam, would make this estimation of depth movement based on area unnecessary and would simplify the recognition of this sort of movement, making it no different than the other linear movements. The second level of the tree analyzes the order in which these maxima and minima occurred to determine movement direction for depth, vertical and horizontal movements or to further classify a movement with large variations in both vertical and horizontal coordinates as a diagonal or circular movement. Finally, the third level classifies the diagonal or circular movement according to its direction based on a further analysis of the order of maxima and minima.

Hand location is classified simply based on hand vertical position in relation to the position and size of the user's head, which is also detected by the system, allowing the determination of whether the hand is above the head, within head height, within chest height (up to one and a half head heights below the head) or below that.

Poses are classified based on a representation of hand contour. This contour is extracted from the image using the algorithm described by Suzuki and Abe [34], and a configurable fixed number of evenly spaced points are sampled along it. The distance between these points and the center of the hand (the position of which is also used for movement

classification) is stored in a feature vector which is normalized, so the maximum distance becomes 1. This vector is also aligned so that the point of maximum distance is always in the first position. Finally, the pose is classified by a direct search within the set of possible gestures, the feature vectors of which have been previously calculated, for the one with maximum verisimilitude with the pose performed by the user. To calculate the feature vectors for the known possible hand poses, a set of hand images was used. These images were user-independent and obtained from the manipulation of a 3D hand model, with a single image required for each pose. The sampling of a fixed number of evenly spaced contour points and the vector normalization handles differences in user hand shapes and sizes well, including those caused by the distance or angle between the hand and the camera, while the vector realignment makes the classification independent to rotations in the image plane. Maximum verisimilitude is estimated by the minimum quadratic error q between the observed feature vector and the known vector for a pose, given by equation (1), where q_i is the error between the observed vector and the known vector for pose i; n is the number of points in the feature vectors (the number of evenly spaced points in the hand contour); d_j is the j-th element of the observed feature vector (distances are already squared when stored in the vectors) and d_{ij} is the j-th element in the feature vector for known pose i.

$$q_i = \frac{1}{n}\sum_{j=1}^{n}\left(d_j^2 - d_{ij}^2\right)^2 \quad (1)$$

If the minimum q_i is still greater than a configurable tolerance, no pose is recognized, otherwise the pose i that produces the minimum error q_i is. This strategy can lead to confusion between similar hand poses, but that is actually considered a positive feature of the system, since it accepts some variability in gesture execution by the users (which increases ease of use, an important requirement for education and entertainment applications) and only a small number of distinct hand poses is necessary to define a much larger number of gestures through the combination of all the gesture components. Besides, not even all of these poses need to be recognized at each moment, since in each interaction context the users should only need to use and remember a subset of gestures much smaller than the total number of possibilities.

Simpler alternatives to this strategy, such as comparing invariant Hu moments, were also evaluated, but yielded less satisfactory results for similar poses and larger sets of them.

The orientation is determined by the angle between the center of the hand and the point with maximum distance to it, already used for normalization and alignment.

Finally, the segmentation of the objects of interest in the image, user hands and head, is done using a simple pixel-by-pixel skin color classification using band-pass filters for hue and saturation and discarding the smaller objects left after an image processing pass. The filters use parameters for the skin color obtained from a quick calibration step in which the user shows a skin region for sampling. Another calibration step records a background model to subtract from the image. Tests showed that these calibration steps were performed, on average, in approximately 9±2s, which was considered acceptable for the requirement of demanding little calibration and preparation. Currently, the system does not segment hands from arms and users must wear long sleeves, but this is a priority change for future works. It does not deal well with changes in lighting (they may require a new calibration, which currently is not done automatically).

4.2 Architecture and Integration

This recognition system is an instantiation of a more general component-based framework for image-based gesture recognition, also developed here but independent of the model, which facilitates the replacement or even the combination of algorithms for specific tasks. In this implementation, separate modules organized in a pipeline pattern are responsible, in this order, for image capture, segmentation, analysis and gesture recognition. The image capture model feeds an image object to segmentation, which in turn feeds the segmented image to the analysis. The analysis module produces a set of numerical features as output, which is then used by the recognition module. The gesture model and the state machine that implements it are inside this module, as are the algorithms used to classify movements, locations and poses. Finally, the recognition model outputs the recognized gesture or partial information about it, such as the last pose recognized and gesture parameters such as hand position, orientation and area. During this entire process, several images (for instance showing the background or not and showing only the segmented objects or not) are also made available so that applications can show visual feedback to the users.

A final and optional module is part of the system, although it is not involved in gesture recognition per se. This module, Desc2Input, translates recognized gestures to other sorts of input, for instance operating system input events such as keyboard strokes or mouse movements and clicks. This option greatly facilitates the integration of the gesture recognition system with other applications. In fact, the input to this module may be any descriptor, not only for a gesture. It can thus also be used as a simple way to integrate multiple modes of input, such as gesture and speech recognition or several different input devices. This integration with other modes of interaction is considered important for the system, since gestural interaction is better suited to some tasks than to others, for which its combination with other modes can be advantageous [1].

An earlier version of the Gestures2Go model and recognition system was discussed in a previous work [35], presented in the Brazilian symposium on games and digital entertainment, with a greater focus in implementation details and applications in gaming and less test results. Several of those implementation details are different in the current version of the system (such as separate orientation in gesture definition and parameters or contour vector alignment).

V. TESTS AND RESULTS

Up to this point, the goals for the evaluation of Gesture2Go were mostly related to determining its functional and performance characteristics and whether they satisfy the system requirements. As such, most of the tests were performed by a

reduced number of expert users with knowledge of 3D interaction and at least some knowledge of the system.

To determine whether the gesture model was indeed broad enough to allow the definition and use of gestures appropriate to different domains and interaction contexts in 3D environments applied to entertainment and education, the three applications previously discussed and used during the user studies were used once again. In all three cases, gestures and parameters coherent with the applications and actions could be defined using our gesture model. VIDA and the illumination environment, however, were developed in parallel with Gestures2Go, and by different teams. Because of differences in the timing of the recognition system tests and the development of these applications, the gestures defined for them were tested separately from the applications themselves. But the gestures for City of Heroes could be integrated to the game and tested in that context. We also measured execution times for the system and for its components and developed a simple interactive application that facilitated testing several features of the system as they were added to it.

5.1 Tests with City of Heroes

In City of Heroes, a commercial game completely unmodified by us, many movement-based gestures (with a high percentage of bimanual gestures) should mimic the animations performed by the avatar when activating certain actions to trigger those same actions. Using Gestures2Go it was possible, in these tests, to choose 9 gestures that closely resembled 9 different actions for a character and 4 more for system control, more than are usually necessary during much of the game. While more actions could have been defined and tested, we judged that this number was already somewhat higher than most gesture-based interfaces should demand of their users.

The gestures chosen to mimic and trigger character actions were close enough to the movements performed by the character that users who were inexperienced with the game declared it was considerably easier to remember the gestures, after watching and practicing them briefly, than to remember the keyboard or mouse commands for the same actions after a similar period of learning. The user gestures and character animations were not similar enough, however, to force the user to crouch or bend forward or backward as the character sometimes did. Users could possibly do it if they wished, as long as their hands and head remained in the camera field of view since they are the only elements segmented from the image. This possibility, however, was not tested at this point. As mentioned before, cases in which different actions used the same or similar animations in the game were easily worked around by using different hand poses. City of Heroes characters are, for technical reasons, currently limited to using only a small number of hand poses. Initial location was also an important parameter to differentiate between several of these gestures and to make them more similar to the character animations.

The four additional gestures for system control that were also suggested, defined and tested for this application were: selecting the next target (an arbitrary gesture similar to the one used for hitch-hiking); moving the mouse pointer (users picked an open hand with fingers spread, because the game uses a representation of a hand in this pose as a mouse pointer over certain objects of interest); interacting with an object under the mouse pointer (closing the hand, as if grabbing the object); and manipulating the camera (mimicking holding a camera near the head). This last gesture was the only one not included in either of the two test scenarios described below.

For both City of Heroes and the illumination environment, ground travel within the 3D environment could also have been performed with gestures. A simple technique to do so was tested, using a pointing up gesture to move forward and pointing to the sides to turn. Whereas this technique could be used without issues in the illumination environment, it was more problematic for the game. It is not always necessary (and sometimes not even possible) to move the character and trigger its animation-based actions at the same time in City of Heroes, but when it can be done it can be advantageous within the game. Using gestures for both triggering actions and traveling made doing both simultaneously complicated or impossible. The game also allows moving away from the ground with jumps or flying, which would require a more complex gesture-based technique, added on top of the already numerous gestures defined for the game. Instead, it was decided that this was a good opportunity to test Gesture2Go's integration with other modes of interaction. A dancing mat (where the user can step on one or two out of 8 buttons arranged around a central space, behind and in front of him, on each side and on the diagonals) was used to control travel in the game, including jumping on the mat to make the avatar jump. The output from this mat was transformed into descriptors which were associated to and translated as operating system events in the same way and by the same module that translated the gestures. This travel technique was so well received by the users (who even commented more than once on how natural they found it to act with their hands but to move around with their feet) that it was also adopted as the default option in the illumination environment.

Two scenarios test scenarios were used with this setup. In the first, the user selected a single computer-controlled opponent with the mouse pointer and used a series of mostly single target actions to defeat it. In the second scenario, the user selected the second closest opponent in a group using the command to select the next opponent, closed up with the group and defeated them all at once using area-of-effect actions. All gestures performed during these tests were correctly recognized even though the tests were performed before a cluttered background, but some care was taken when positioning the camera and the mat (which served as a boundary for the user's position) to avoid certain illumination effects that degrade the simple segmentation strategy used, such as intense specular reflections on the skin or bloom effects between the fingers.

Fig. 3 shows the user view of one moment of this test. In the lower left corner, the feedback window can be seen, showing the segmented hands and face. While the current character action animates the user in Fig. 3 is already performing the gesture to trigger the next action. Several users, not only those participating in the test but also those who watched videos of it, actually commented about this fact, which they felt was a delay in the gesture recognition system that only showed up in City of Heroes and not in the other applications. This was a matter of perception only, since the gesture recognition delay was as imperceptible in the test with the game (lower than 15ms) as it

was for the other applications. In this case, however, both the user input (the gesture) and the interface's response (the avatar's animation) took a perceptible amount of time to execute. This was the only case in which users had this perception of lag in the recognition. It was not perceived when using an apparently immediate form of input (such as the keyboard, mouse or dancing mat) to interact in City of Heroes, even though the character animation only began after the appropriate key press had finished. When using gestures with a perceptible duration to trigger seemingly instant actions, such as all interactions in the illumination environment and in VIDA, or selecting targets or objects in City of Heroes, no delay was perceived either. When both input and response had perceptible duration, though, it was. In this game, this was possibly aggravated by the very similarity between gestures and animations, but even when this similarity was purposefully eliminated, the sense of delay remained. This unexpected test result points that, for this sort of interaction, we may need to recognize incomplete gestures to reduce this perceived lag.

Fig. 3. Test with a digital game (Color Plate 4).

5.2 Tests for educational 3D environments

In the virtual environment that allows the exploration of illumination concepts in a computer graphics course, gestures should allow users to attribute symbolic values (in the form of numerals) to objects and algorithms, to turn the former on and off and switch between the latter. Two different scenarios and sets of gestures were suggested in this case, using primarily either bimanual static gestures or gestures made up of a sequence of two poses to indicate object and action pairs. Both sets of gestures suggested for the illumination environment could be defined and tested successfully, totaling 20 gestures, 10 for each scenario.

In VIDA several gestures with one or two hands were suggested to select specific features in anatomical structures and to allow the 3D manipulation of those structures, making extensive use of gesture parameters. In this case the definition of gestures was not so simple. Perhaps because this was the only application for which users were not shown a camera when suggesting gestures or perhaps because of the nature of its main task (3D manipulation) some of the gesture sets suggested for this application required rotations in two perpendicular planes and the use of the orientation as a parameter in both. This could not be resolved by the current implementation of Gestures2Go with a single camera.

It was possible, however, to run two instances of the gesture recognition system, even in the same computer, with different gesture definitions and receiving input from different cameras, the image planes of which were approximately (but did not need to be exactly) perpendicular. These two instances of Gestures2Go were independent, each with its own set of possible gestures, their definition and association with actions using Desc2Input. When one instance of the system recognizes poses and rotations in its image plane, these same poses and rotations are perpendicular to the image plane used by the other instance and are not associated with any gestures or actions, being simply ignored by that other instance. In this way, the gestures suggested by the users, requiring rotations in two perpendicular planes, could be addressed. The only extra calibration this setup required was running twice through the two quick calibration steps described previously, once for each instance of the system (which was necessary because the cameras had different color parameters, illumination conditions and known backgrounds).

5.3 Execution times and feature tests

We also measured execution times for the recognition system, for all modules (except image capture and the translation of descriptors into events) and the main activities within these modules. Table I shows approximate average times for 375 test frames (5 tests of 5 seconds at 15 frames per second) in a single core of a 3GHz Intel Core 2 Duo processor with 2GB of available RAM (of which only a small portion was used during the tests). Poses were classified within 10 possibilities (classification time increases linearly with the amount of possible poses).

TABLE 1: EXECUTION TIMES.

Activity		Time (ms)
Segmentation		13.600
Analysis	Connected components	0.650
	Areas and center	0.013
	Features	0.003
Recognition	10 poses	0.003
	Movement	<0.001
	Gesture	<0.001

These results show that the entire process takes little more than 14ms in this platform, far below the requirement of 100ms derived from the study of interaction in MMOs [16] and leaving considerable processing time for other activities common in 3D environments (gesture recognition was performed every 67ms in these tests). Over 95% of this time is taken up by the segmentation, which is an obvious target for improvement in future works.

Several small and simple applications were also created to test particular features of the system as they were developed, involving potential users in testing the system all along the development process. A MS Windows application that allows a gesture with the right hand to move the mouse pointer and another to perform left clicks was created to test the use of

gesture parameters as well as the translation of gesture descriptions as operating system events. A simpler test program showing approximate pose orientation interactively as the user rotates the hand while maintaining a particular hand configuration was used to test the use of orientation as a continuous parameter (inspired by the preliminary user studies with VIDA). Another application is used to interactively test the recognition of all the different kinds of gestures as well as the calibration process. It shows the segmented image and the recognized poses at all times in a window, and prints the name of recognized gestures in another. Finally, a static comparison between pre-segmented poses (96 images of 16 different poses) was one of the first evaluations applied to the system during its development, to compare different strategies for pose classification, trying to isolate it from the segmentation problem. Currently, this test recognizes 95.8% of these poses correctly. Besides being used to involve and get feedback from potential users during the development, these tests were also continuously updated and performed after each project interaction as a form of regression testing.

The interactive tests were particularly useful for noticing that many false negatives in pose recognition were simply imperceptible to the users in the context of recognizing the gestures themselves and using them for issuing commands or manipulating objects. The next correct recognition made these errors irrelevant. False positives in pose recognition were rarer and more critical, since they often provoked a false negative in the gesture recognition or, worse still, a false positive. These tests also showed at least two relevant problems. The first is that the simple segmentation strategy used in this implementation, while behaving well enough in cluttered backgrounds, even those containing skin-toned objects, can be quite sensitive to certain illumination effects that change the skin color of the objects of interest, such as intense specular reflections and bloom effects. The second problem regards poses showing a side of the hand (instead of its palm, back or a fist) to the camera. Users reported some small difficulty and the need to pay closer attention to the feedback in some of these cases (such as in a "karate chop" gesture) to align the hand properly with the camera angle, since relatively small variations in the wrist angle could end up showing large variations in the captured image, which does not happen so intensely for frontal or back hand poses. In other cases, however, performing the gesture for the camera in a recognizable way was difficult enough to make users give it up. Simply taking the "karate chop" pose and bending all fingers forward together by 90 ° generates one of these "impossible" poses. The variation of a pinching pose numbered as 6 in Fig. 1 is another example. In this case, to avoid large variations in the captured images, users must align not one but two axes with the camera, which proved too complicated for use in interaction. In the latter case at least, using a pinching pose with only the index finger and thumb extended away from the hand while the others stay in a fist somewhat alleviates this problem. While these two problems must certainly be addressed in future works, presently they do not represent a serious source of concern, since neither the segmentation algorithm nor the pose recognition strategy are among the intended and major contributions of this work.

VI. CONCLUSIONS

The test results show that our system currently has its problems and limitations. The segmentation strategy used, in particular, takes up over 95% of the processing time and is the most common source of errors in the recognition, particularly under unfavorable lighting conditions. It also currently imposes the limitation that users must use long sleeves to segment the hand from the arm. Eliminating this limitation is a priority for future works, but since the entire segmentation strategy is a strong candidate for a complete redesign in the future (in fact at least one work towards this end is currently under way) the hand segmentation will probably be included in that redesign, instead of simply being added up on top of the current system. Despite these limitations, we can construe that, based on our test results, Gestures2Go already satisfies all of the requirements discussed in the introduction. These requirements for interaction in entertainment and education applications using virtual and augmented environments were based on an analysis of the relevant literature as well as on preliminary user studies performed during this work.

The most important requirement, allowing the definition of a large number of gestures to increase the likelihood of finding the most adequate ones for each interface action, was plainly satisfied by the model, which allows the definition of many thousands of gestures. Even the simpler implementation discussed here, with no 3D information and only 4 locations instead of 12, allows the definition of almost 40 thousand gestures if one considers only the 9 standard poses (1 to 9 in Fig. 1, with a modified pinching pose to cause less problems) and the 3 orientations most comfortable to users, out of the 12 possible. The gesture model which allows this by defining gestures as combinations of simpler components, along with the simple strategy used to recognize these gestures, are the two major contributions in this work.

Because all these gestures are defined based on so few components of each kind and the components are recognized separately, all of them could be recognized at once, but since this is not necessary for use in interaction, much smaller subsets of gestures to be recognized at each moment may be defined, previously or in runtime, which may reduce recognition errors and processing time.

This gesture variety was not simply considered as numbers in a vacuum, but was also tested in the context of three 3D applications: a commercial massively multiplayer 3D game and two educational 3D environments in the domains of computer graphics and anatomy. We could define adequate gestures, suggested in preliminary user studies, for all three applications. Different kinds of gestures were used to do so. All these gestures could be correctly recognized by the system during testing, although some of the gestures suggested for 3D manipulation in VIDA required two instances of the recognition system running at once.

The tests also proved that our recognition system satisfied its other requirements. It does accept variations, both in user hand shapes and in gesture performance; a single, low-cost and widely available kind of camera is used as the only sensor for the system; calibration time takes up only about 10 seconds; various forms of visual feedback may be shown to the user; the

system runs in real-time taking up little processing time and, finally, its use by programmers requires little to no specialized knowledge. One only needs to define a gesture by choosing its components and associate it with the desired action.

A promising direction for future work pointed out during testing is the recognition of incomplete gestures. Taking advantage of depth-sensing equipment, now that this hardware is more widely available, to implement a recognition system using the Gestures2Go model is another important future work which is actually already underway. This should not only allow the use of an even greater variety of gestures, particularly less awkward deitic gestures, but can also be a great advantage for segmentation. Finally, another priority for future work, now that tests have revealed the functional strong and weak points of the system, is to complete the integration of Gestures2Go with other entertainment and education applications and perform formal usability tests with a larger pool of users of more varied background, using applications such as, but not limited to, VIDA, the illumination environment and, of course, City of Heroes. This is another work that is already underway.

REFERENCES

[1] D.A. Bowman, E. Kruijff, J.J. LaViola and I. Poupyrev. 3D User Interfaces: Theory and Practice. Addison Wesley Longman Publishing Co.; 2004.

[2] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle and X. Twombly. A review on vision-based full dof hand motion estimation. In: *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society; 2005*.

[3] OpenNI User Guide. OpenNI; 2010. Available: http://www.openni.org/Documentation.aspx

[4] J. Wobbrock, A. Wilson, Y. Li. Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In: *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology. ACM; 2007*, p. 159–168.

[5] T. Starner, B. Leibe, B. Singletary and J. Pair. Mind-warping: Towards creating a compelling collaborative augmented reality game. In: *Proceedings of the 5th International Conference on Intelligent User Interfaces. ACM; 2000*, p. 256–259.

[6] W. Swartout and M. van Lent. Making a game of system design. *Communications of the ACM 2003*;46:32–39.

[7] B. Kane. Beyond the gamepad. 2005. Available: http://www.gamasutra.com/view/feature/2378/postcard_from_siggraph_2005_.php

[8] J. Payne et al. Gameplay issues in the design of spatial 3D gestures for videogames. In: *CHI '06 Extended Abstracts on Human Factors in Computing Systems. ACM; 2006*, p. 1217–1222.

[9] B. Shneiderman and C. Plaisant. Designing the User Interface: Strategies for Effective Human-Computer Interaction; 4 ed. Pearson Addison Wesley; 2004.

[10] C. Youngblut. Educational uses of virtual reality technology. Tech. Rep. D-2128; *Institute for Defense Analysis*; 1998.

[11] R. Azuma. A survey of augmented reality. *Teleoperators and Virtual Environments 1997*;6:355–385.

[12] M. Nielsen, M. Storring, T. B. Moeslund and E. Granum. A procedure for developing intuitive and ergonomic gesture interfaces for man-machine interaction. In: *Proceedings of the 5th International Gesture Workshop. 2003*.

[13] K. G. Derpanis, R. P. Wildes and J. K. Tsotsos. Hand gesture recognition within a linguistics-based framework. In: T. Pajdla and J. Matas (ed.), Computer Vision: *ECCV 2004*. Lecture Notes in Computer Science; Springer; 2004, p. 282–296.

[14] T. Hong. Shoot to thrill. Game Developers Magazine 2008;15(9):21–28.

[15] F. Tsuda, P. Hokama, T. Rodrigues and J. Bernardes. Integration of JARToolkit and enJine: Extending with AR the potential use of a didactic game engine. In: *Proceedings of the 9th Symposium on Virtual and Augmented Reality. 2007*, p. 51–59.

[16] T. Fritsch, H. Ritter and J. Schiller. The effect of latency and network limitations on MMORPGs: A field study of Everquest2. In: *Proceedings of 4th ACM SIGCOMM Workshop on Network and System Support for Games. ACM; 2005.*

[17] H. Kang, C. W. Lee and K. Jung. Recognition-based gesture spotting in video games. 2004.

[18] S. Mitra and T. Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man and Cybernetics 2007;* 37(3):311–324.

[19] V. I. Pavlovic, R. Sharma and T. S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence 1997*; 19:677–695.

[20] X. Liu and K. Fujimura. Hand gesture recognition using depth data. In: *Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition. IEEE Computer Society;* 2004, p. 529–534.

[21] J. M. S. Dias, P. Nande, N. Barata and A. Correia. O.G.R.E.: Open gestures recognition engine. In: Proceedings of *the XVII Brazilian Symposium on Computer Graphics and Image Processing. IEEE Computer Society*; 2004, p. 33–40.

[22] Z. Mo. Gesture interface engine. Ph.D. thesis; University of Southern California; 2007.

[23] K. Lyons, H. Brashear, T. Westeyn, J. S. Kim and T. Starner. GART: The gesture and activity recognition toolkit. In: Proceedings of the 12th International Conference on Human-Computer Interaction: *Intelligent Multimodal Interaction Environments. Springer-Verlag; 2007*, p. 718–727.

[24] M. Kolsch, M. Turk and T. Hollerer. Vision-based interfaces for mobility. In: *Proceedings of the International Conference on Mobile and Ubiquitous Systems.* 2004.

[25] W. C. Stokoe. Sign language structure: An outline of the visual communication systems of the american deaf. Studies in Linguistics: Occasional Papers 1960; 8.

[26] S. K. Liddell and R. E. Johnson. American sign language: *the phonological base. Sign Language Studies 1989*; 64:195–278.

[27] F. Quek. Toward a vision-based hand gesture interface. In: *Proceedings of the Conference on Virtual Reality Software and Technology. World Scientific Publishing Co.* 1994, p. 17–31.

[28] F. Quek et al. Multimodal human discourse: Gesture and speech. *ACM Transactions on Computer-Human Interaction 2002*;9:171–193.

[29] ISO-IEC. International standard 14977: *Information technology: Syntactic metalanguage: Extended BNF*. 1996.

[30] R. Tori, F. Marques, R. Nakamura, J. Bernardes, C. Correa, D. Tokunaga. Design de interacao para um atlas virtual de anatomia usando realidade aumentada e gestos. In: *Proceedings of Interaction South America '09.* 2009.

[31] J. G. Elias, W. Westerman and M. Haggerty. *Multi-touch gesture dictionary. 2010.* U.S. Patent 7840912.

[32] J. Epps, S. Lichman and M. Wu. A study of hand shape use in tabletop gesture interaction. In: *CHI '06 Extended Abstracts on Human Factors in Computing Systems. ACM, 2006*, p. 748–753.

[33] G. A. Miller. The magic number seven plus or minus two. Psychological Review 1956;63:81–97.

[34] S. Suzuki and K. Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics and Image Processing 1985*;30(1):32–46.

[35] J. Bernardes, R. Nakamura and R. Tori. Design and implementation of a flexible hand gesture command interface for games based on computer vision. In: *Proceedings of the 8th Brazilian Symposium on Games and Digital Entertainment*. *SBC*, 2009, p. 89–98.

J. Bernardes was born in Rio de Janeiro, Brazil, in October 11th, 1976. He graduated as a mechanical engineer, with a concentration in systems and automation, from Escola Politecnica da Universidade de Sao Paulo in 1999 and obtained the degree of Master of Engineering from the same institution in 2004 with research chiefly in virtual reality and scientific visualization. Bernardes achieved the degree of Doctor of Science in 2008 in the area of digital systems, also from Escola Politecnica. His major field of study is human-computer interaction, particularly the use of image processing, analysis and synthesis in interaction and applications in entertainment and education.

He is a Professor in Escola de Artes, Ciencias e Humanidades da Universidade de Sao Paulo and has participated in several research and development software projects since he was an undergraduate student. Before his current position he was a professor in ISES, SENAC and UNIFIEO and also worked as a professional engineer for Svedala Faco and WEG Motors for a few years after graduating, before dedicating himself to teaching and research.
Prof. Bernardes is a member of the Brazilian Computing Society (SBC).

R. Nakamura was born in Sao Paulo, Brazil, in June 4th, 1976. He graduated as a mechanical engineer, with a concentration in systems and automation, from Escola Politecnica da Universidade de Sao Paulo in 1999 and obtained the degree of Master of Engineering in the area of digital systems from the same institution in 2002, with multiuser virtual environments as his main topic. Nakamura obtained the degree of Doctor of Science in 2008, also from Escola Politécnica and in the area of digital systems, with augmented reality and digital games as his main subjects of study. His major field of study is human-computer interaction, particularly in digital games.
He is a Professor in Escola Politecnica, part of the managing committee for the special group in games and digital entertainment of the Brazilian Computing Society (SBC) and has participated in several interactive technology projects in the areas of virtual and augmented reality and digital games.
Prof. Nakamura is a member of the Brazilian Computing Society (SBC) and of the Association for Computing Machinery (ACM).

R. Tori was born in Sao Paulo, Brazil, in December 1st, 1959. He graduated as an electric engineer, with a concentration in digital systems, from Escola Politecnica da Universidade de Sao Paulo in 1982 and obtained the degree of Doctor of Science in the same institution in 1994, with a thesis in the area of hypermedia and multimedia. In 2003 he achieved his Habilitation from Universidade de Sao Paulo, in the area of interactive technologies. His major field of study is human-computer interaction, especially the use of virtual and augmented reality and digital games applied to education, health and entertainment.
He is a Professor in Escola Politecnica and in Centro Universitario SENAC, has coordinated several research and development projects in the area of interactive technologies as well as several scientific events and committees in the areas of computing and design. He is a member of the deliberative council of the research laboratory called "Escola do Futuro" and author or co-author of several books.
Prof. Tori is a member of the Brazilian Computing Society (SBC) and of the Association for Computing Machinery (ACM).