A Fast and Robust Gesture Recognition System for Exhibit Gaming Scenarios

Marco Roccetti
Computer Science Department
University of Bologna
Mura Anteo Zamboni 7
40127 Bologna, Italy
roccetti@cs.unibo.it

Gustavo Marfia
Computer Science Department
University of Bologna
Mura Anteo Zamboni 7
40127 Bologna, Italy
marfia@cs.unibo.it

Angelo Semeraro
Computer Science Department
University of Bologna
Mura Anteo Zamboni 7
40127 Bologna, Italy
semeraro@cs.unibo.it

ABSTRACT

With no doubt, advanced human-computer interaction technologies represent one of the key selling points of all the most popular new gaming consoles. Players, in fact, can now interact realistically with their gaming environments, not suffering anymore from that lack of authenticity that was caused by the use of a simple joysticks or mice. Such incredible result has been achieved by means of new advanced hardware specialized in the recognition of players' gestures. In this work, we show that the same objective can be attained utilizing an off-the-shelf webcam and a robust gesture recognition software system. In particular, our novel gesture recognition system, which is based on artificial vision algorithms, leads to a fast and robust interpretation of hand gestures. This type of technology has proven to be particularly efficient for immersive gaming experiences. The efficacy was demonstrated through a game, which has been enjoyed by hundreds of visitors at the 2010 Shanghai World Expo.

Categories and Subject Descriptors
K.8.0 [**Personal Computing**]: Games.

General Terms
Design, Experimentation.

Keywords
Artificial Vision, Interactive Game, Hand-Free Gesture Recognition, Shanghai World Expo 2010.

1. INTRODUCTION

All the major producers of computer games and computer game consoles permanently engage each other at finding new and innovative ways of entertaining their customers. Perhaps the most important challenge of the past few years has been that of providing players with the most exciting and realistic human-computer interface. In fact, recent trends show that the higher the realism a player enjoys when interacting with a game, the better will be the game's chances of beating its competitors in the market share arena. This means that customers no longer feel amused when gaming with traditional hardware as joysticks, keyboards and mice, but rather wish to be able to perform, while gaming, the same natural actions that would be performed for real.

The technology that first has raised the bar of customer expectations from new gaming consoles has been the major reason for success of the Nintendo Wii. While moving, player movements are recognized in real-time by a combination of infrared and accelerometer sensors and echoed in the virtual gaming world. Although such technology does not yet enable hands-free gaming interactions, as it still relies on the use of a controller (i.e., the Wiimote), the Nintendo Wii has been the first console to successfully support natural body movements while playing a computer game. Very recently, an important step forward has been performed with the introduction of the Microsoft Kinect sensor, an advanced video camera that, combined with an advanced software system, supports human body recognition on the Xbox console. While playing with Kinect, players can freely move without touching or holding any controller, as their bodies are the controllers that provide commands through natural body movements. All this has required the use of an advanced video camera, which, combined with a depth sensor, can estimate the distance from the objects that move in front of it in real-time and hence aid the real-time recognition of the position of each of the parts of a player's body.

The main contribution of this work is to provide a description of our new gesture recognition system that, with the sole use of one cheap hardware component (an off-the-shelf webcam), supports playing computer games without the aid of any controller. The main novelty of this system lies in the use of contextual information that can be inferred by the game setting layout and the devise of three artificial vision algorithms capable of recognizing a player's inputs: (a) a hands recognition algorithm; (b) a hands following algorithm, and; (c) an action recognition algorithm. In brief, our gesture recognition system has been devised assuming that a player faces a screen where a virtual world is displayed, while a webcam captures his or her movements from above. The hands recognition algorithm receives the captured video stream and analyzes it, frame by frame, identifying a sub-set of key points where a player's hands and forearms have been detected. After this step, the hands following algorithm determines the exact position of both hands. Finally, the correctness of each of a player's actions is checked in terms of timing and trajectory by our action recognition algorithm. A player, in the meantime, while moving, can appreciate the effects of his or her actions on the screen. This type of technology has proven to be particularly efficient for immersive gaming experiences displayed in exhibit scenarios.

In fact, all the mentioned algorithms have been tested within a computer game, the Tortellino X-Perience, recently displayed at the 2010 Shanghai World Expo [1]-[4]. The underlying objective of the game was that of teaching players how to prepare Tortellini Pasta starting from its main ingredients (flour, water and eggs), while challenging them through a sequence of stages given by its recipe. Clearly, the preparation of a Tortellino requires stepping through a very well defined number of stages where our gesture recognition system played a very important role: it supported the detection of correct movements from incorrect ones in real-time.

The rest of this paper is organized as follows. In Section II we survey some of the approaches that have been, to this date, used in implementing advanced human-computer interaction schemes in commercial gaming consoles. In Section III we provide a discussion on the use of an immersive gaming system within a public exhibit. In Section IV we provide a succinct description of the algorithms we developed, while Section V describes our Shanghai test-bed and reports on its performances. We finally conclude with Section VI.

2. RELATED WORK

A wealth of research has investigated how new and more natural means of interaction between humans and computers could be developed from many different standpoints [5-17], we will here focus on the prominent approaches that have appeared in commercial game consoles.

The most revolutionary and widespread controller that has been to this date created is, beyond any doubt, the Nintendo Wiimote controller. The Wiimote is equipped with an infrared camera sensor, which combined with two light emitting sources placed within the sensor bar positioned above or below the TV set, is used to locate its position in real-time. Such controller, in addition, also carries an accelerometer, used to register any sudden acceleration that is experienced by a player's hand while engaged in a game. Clearly, using such technology requires players holding a Wiimote, hence demands the use of a hardware device that acts as a broker between the gestures a player performs and a Wii console. This hardly suites a scenario where hundreds or even thousands of different people can play a game using the same console during a single day, as was the scenario we dealt with in Shanghai.

Very recently, computer game players have been able to enjoy body free gaming with the Microsoft Kinect sensor and the Xbox. Kinect is a horizontal bar that is placed either above or below the video screen, containing a depth sensor in addition to an RGB camera, capable of recording the distance of all objects that lie in front of it. Depth information is then processed by a software engine that extracts, in real-time, the key human body features of players, thus enabling the interaction between the physical world and the virtual one. We will show that similar results can also be achieved simply using a normal video camera.

Also Sony has supported mixed reality experiences with the EyeToy and the Playstation Eye. Both of such products are based on a digital camera, using a very similar hardware approach to ours. However, our two approaches differ on the software side, as the Playstation Eye focuses on the detecting motion patterns above certain given areas, while our algorithms also recognize the position and the gestures performed by both hands.

In summary, the main experiences that may be found in commercial computer gaming consoles for implementing advanced human-computer interaction schemes either adopt complex hardware interfaces needed to detect and track movements, or reduce the accuracy of the tracking and gesture recognition processes, hence, reducing the realism with which players are supposed to play.

3. PLAYING GESTURAL GAMES IN EXHIBITIONS

Our new gesture recognition system is based on a set of artificial vision algorithms tailored to suite exhibit gaming scenarios, where a computer game is displayed and can be played in a public area within a museum or a fair, for example.

Such types of scenarios introduce one advantage and one disadvantage, in terms of human-computer interaction design. On one side a clear advantage is given by the possibility of controlling the entire setting layout, and, hence, the context within which a player interacts with a game (e.g., the game arena area, position, illumination level, etc.). On the other side, instead, such scenarios pose new challenges, as no a-priori assumptions can be made on the players that, eventually, will play a game (e.g., skill level, physical features, etc.).

We devised a very traditional layout where a visitor plays waving his or her hands within a restricted arena, while facing a video screen that displays the game graphical environment (Figure 1). The only difference with other well-known commercial gaming consoles is given by the video camera, which in our case is placed above the game arena, while, in Microsoft Kinect or Playstation Eye, for example, lies in front of the player. Such choice gives us the possibility to accurately track the horizontal movements that are performed by a player's hands. The gaming layout, hence, represents here a contextual piece of information that can be tuned and put to good use to optimize the efficiency of the gesture recognition system in identifying and interpreting a player's movements for a given game.

Such type of scenario, however, also represents an obstacle to the use of off-the-shelf technologies for gesture recognition. Just as an example, consider the case where we adopted the Playstation Eye gesture recognition system to support our game: before every time a new player started a game a new calibration phase would have been needed. In fact, the Playstation Eye requires an initial calibration phase where the body dimensions of a player are estimated. Another popular technique in legacy gesture recognition systems is the use of color recognizers to track hands (also requiring an initial calibration phase). Clearly, any of such solutions that require an initial calibration phase would result to be unfeasible in a context where a new player can join every few minutes.

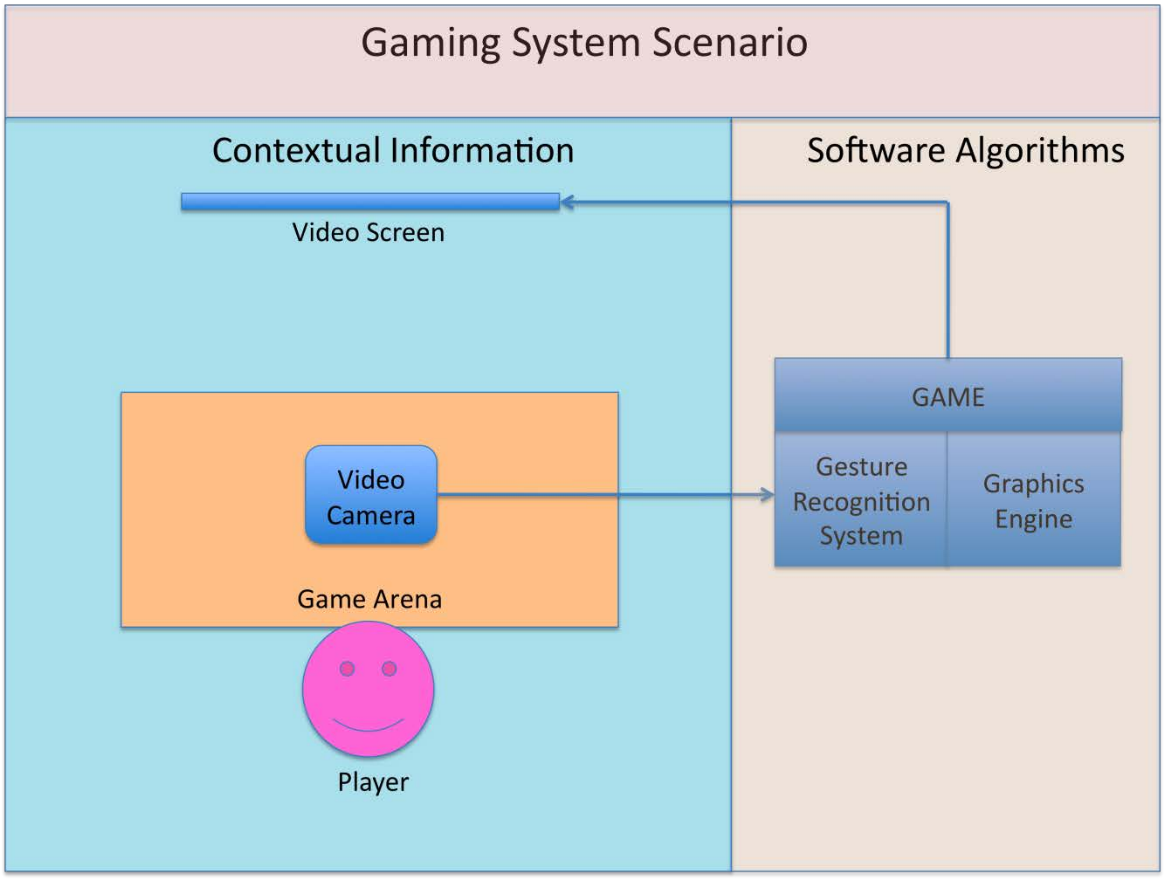

Figure 1. Gaming system at an exhibit.

In summary, a gesture recognition system that can efficiently cope with such scenario should be able to put to good use the a-priori assumptions that can be made on its setting layout, while being robust to its variety of users.

4. ALGORITHMS

Our gesture recognition system is based on three algorithms, which, working in cascade, recognize the actions that are performed by a player at each given game stage. Anticipating here how they work, the first one recognizes any luminance change over the game arena while someone moving his or her hands, compared to the static scene captured before anyone plays. In this way our algorithm distinguishes those areas that a player crosses with his or her hands and forearms while challenged in a game. Once those areas have been identified, the second algorithm processes them searching for the farthest positions reached by a player's hands, as for the game under consideration, a hand can be associated with such points. Finally, a third algorithm recognizes the gestures that a player performs tracking those farthest points, given that each movement a player performs involves moving along a trajectory which starts from a zone and ends in another where the hand completes its gesture. We now follow describing each algorithm with more detail.

The first algorithm worries about efficiently tracking the hands and the forearms of a player while they wave above the game arena. To fulfill this aim, a player's hands are only searched for at specific locations (e.g., pre-defined pixels), computing the luminance differences between the static scene before the hands and forearms enter the arena and the scene after a player begins playing. This approach is justified by the fact that a coarse-grained check for the movement of an arm or hand is sufficient for its detection, thus the game arena can be divided in an *N* x *M* grid of blocks that are chosen sufficiently wide to provide the granularity necessary to distinguish an average hand, whose width is about 5 centimeters. However, before anyone plays, such procedure requires first the performance of a unique calibration phase, that involves the preliminary division of the underlying surface rectangle into an *N* x *M* grid of blocks, and the saving of the maximum and minimum RGB luminance values for a subset the of *K* pre-defined pixels that have been identified within each block. Clearly *K,* the number of pre-defined pixels that are observed within each block, should be chosen considering that higher values increase the computational effort required to determine the presence of a hand or a forearm, but lower values could cause inaccuracies resulting in a hand waving above the game arena without being detected. Hence, after *K*, *N* and *M* have been chosen and the initial calibration phase has been run returning the minimum and maximum luminance values observed during a sufficient time per each pixel *p* while no one was playing, we can move on to how this algorithm works while the computer game is actively running. While a player is engaged in a game, the algorithm checks for the presence of a hand above any of the *K* pixels of each block within each new frame provided by the video camera. Per each new frame, the algorithm proceeds as follows. First, a bi-dimensional low-pass Gaussian filter is applied to smooth out color peak values given by small object movements and/or brightness variations. Second, within each block, our algorithm checks whether the luminance values of each of the *K* pixels *p* falls within the minimum and maximum luminance interval. If not, the pixel is changed its luminance value. Third, the number of pixels that changed is counted for each block. If this number exceeds a given threshold, then the luminance of the entire block has been altered. However, to confirm that an area composed of changed blocks contains a hand or a forearm, it is important to also check that all those blocks are close to each other (adjacent). To this aim, the fourth and final part of this algorithm iteratively checks whether all the changed blocks represent an area, termed *active area*, area where each block can be reached from any other one moving along a path that lies within it. Only after this final check, and at the end of all this process, we can finally conclude that that active area represents a player's hand and forearm. In conclusion, such approach reveals to be computationally efficient because it avoids checking all the pixels that compose the game arena and its implementation can be done very efficiently utilizing a multi-threaded strategy that checks for luminance changes over different areas of the arena.

Once the hands and forearms of a player are identified, it is now the turn of recognizing the gestures that he or she performs. Since it is complex to follow a player's hands with precision in real-time, we opted for following a representative point for each hand. Such hands representative points, in our system, coincide with the extreme points that rest farthest away from a player and closest to the video screen (obviously, such argument follows from the fact that the video camera is set above a player's head). Not only ours, but many different immersive gaming experiences require that a player extends his/her hands in front of him/her, and hence an efficient way for following them and recognizing gestures can be devised identifying those points that reach farthest away from the player's body. Perhaps, the most important component of our gesture recognition system, our hand following algorithm, works as follows. We start by applying a bi-dimensional Cartesian coordinate system to the game arena, the area above which players wave their hands, taking as a unit of length the dimension of a block. The *x* and *y* axes origin at the bottom leftmost corner of the game arena (as seen by a player watching the video screen from the playing position), and point, respectively, to the right and away from the player, in the direction of the video screen. During the first step of this algorithm, the active areas detected by the hands recognition algorithm are mapped into bar charts as follows. Any block correspondent to a point along the *x* axis is mapped into a bar whose height *y* equals the distance of that point from the player's body (if for a given *x*, multiple blocks exist, all the bars of lower heights are replaced by the longest one). In brief, we can summarize this first step saying that at the end of it a chart is returned where each bar represents the highest *y* value, for each *x*, where a hand, or a part of it, was recognized. Now, the second step of the algorithm (that is, determining the position where the first hand lies) is simple, as the first hand simply given by searching for that bar with the maximum *y* on the bar chart. More difficult is finding the position of the second hand, as it cannot be straightforwardly individuated as the previous hand, as other high *y* values may be due to lower points of still the first hand. This does not mean that we are not seeking for an extreme point any more, but we should be careful in not choosing it blindly, risking of not reliably identifying the second hand. Therefore, we are still looking for another maximum *y* value, but on a bar chart where all the bars that are due to the first hand have been excluded. The third step of this algorithm, hence, entails constructing a new bar chart, from the first one, as follows. Starting from the *x* position where the maximum has been found, the algorithm iterates for lower and higher values, separately. At each step of the iteration, the height (*y*) of each bar of the old chart is compared with the value stored in a variable called *minimum* whose value is then updated. The value of *minimum* is initially set to the maximum *y* value of the old chart and is updated at the end of each step of the iteration if the current *y* value is smaller. Now, starting with the next bar at the left (or at the right) of the one where the global maximum was found, we describe what occurs at each step. A new bar is inserted in the new chart, of height equal to the difference between the current bar length and the value of the *minimum* variable, if this value is positive. If the resulting value is negative, instead, the bar chart will contain a bar of length zero (hence, no bar) in that position. After the bar has been inserted in the new chart, the *minimum* variable is updated, taking the current value on the old chart if this results to be smaller than the value it presently stores. At the end of this procedure the result is a bar chart where bars, if present, are due to the second hand waved by a player. The very final step, hence, is to identify the (*x, y*) coordinates of the maximum of this second bar chart. Now, knowing precisely where each hand is at any given time, it is possible to check the correctness of the movements performed by a player.

Finally, the action recognition algorithm kicks in. Differently from many others algorithms designed with similar scopes in mind, ours not only checks whether a hand reaches a certain position, but also checks whether a hand reaches a certain position following a given trajectory, starting from an origin and ending into a destination zone. In this way, for example, it is possible to define valid trajectories that the player should follow in order to behave correctly, and invalid ones that represent mistakes. Therefore, succinctly summarizing how this algorithm works, the trajectory described by a point that identifies a hand is recognized as correct if it lies within a certain distance from the ideal trajectory (this to give a certain degree of tolerance to players). This mechanism was devised to make our algorithm able to consider as correct a wider set of movements with slightly different trajectories that differentiate only for a few geometric differences. This scheme can be applied to trajectories of either linear or circular shape, thus an ending zone can coincide with starting one.

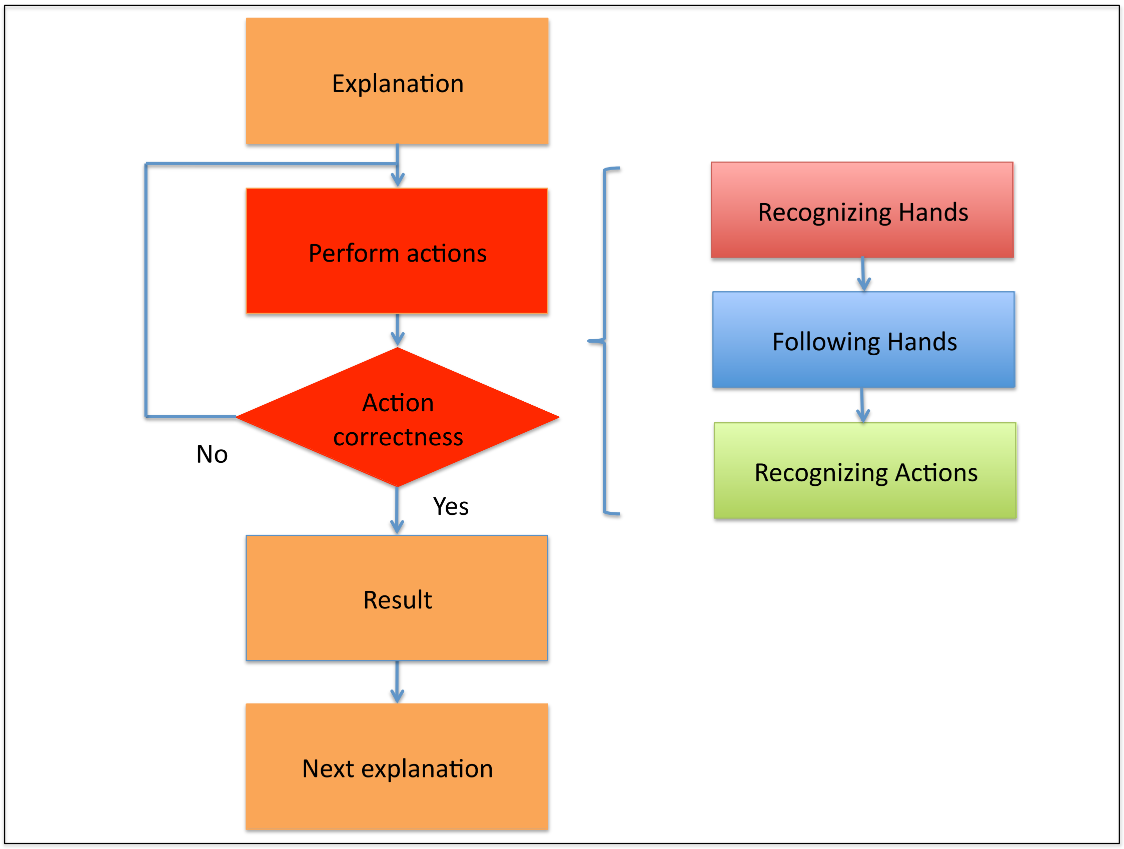

Figure 2. Game stage of Tortellino X-Perience.

5. THE TORTELLINO X-PERIENCE

Our gesture recognition system, more than simply being assessed in a set of experiments performed in a lab, has been tested by several hundreds of people playing our game, the Tortellino X-Perience, in Shanghai, at the 2010 Universal Expo. Spending a few words on the context where this game has been played for the first time, before moving on to its design, we here report that it has been created with the intent of describing an important piece of the culinary cultural heritage of Bologna, Italy, to foreign players. For this reason we chose to challenge players in preparing a Tortellino Pasta, as it represents on of the most famous and traditional dishes of our city.

5.1 A Cooking Gestural Game

When designing our game we had to consider that: (a) not many people are familiar with cooking at all; (b) many players in China may be unaware of what a Tortellino is. Hence we designed a game that teaches as it is played. The game is divided in stages, where within each stage a player is taught the correct movements he or she should perform. Hence, while a player accomplishes a series of movements, mimicking the actions that have been previously displayed, the gesture recognition system checks for their correctness. If incorrect, the player is asked to retry. If correct, instead, the game moves on to the next stage.

Figure 2 summarizes, with a flow diagram, how each stage is organized. At first a player watches a video displaying a *Sfoglina*, a traditional Bologna Tortellini chef, that explains and shows what gestures should be performed. At this point, if, for example, the player needs to mix the flour, water and eggs altogether, he or she will receive points in reward and be able to proceed with the recipe only if performing the correct operation the right number of times.

5.2 Cooking a Tortellino

Cooking Tortellini, as cooking any type of food, follows a very well defined recipe, recipe that can be divided into successive phases, one per each stage of the game. In particular we implemented the following list of phases where a player: (a) is required to move the ingredients at the center of an empty cooking board from the right hand side; (b) kneads the ingredients starting from the yolks and the albumen of the opened eggs float squeezed within the flour, adding some water, and ending up with a smooth ball of dough; (c) rolls out the ball of dough with the rolling pin; (d) cuts the thin foil of dough that is lying on the cooking board into squares; (e) stuffs the squares of dough with the meat and cheese that are sitting on the side of the cooking board; (f) closes the first pair of opposite ends of the Tortellino, and; (g) closes the second pair of opposite ends.

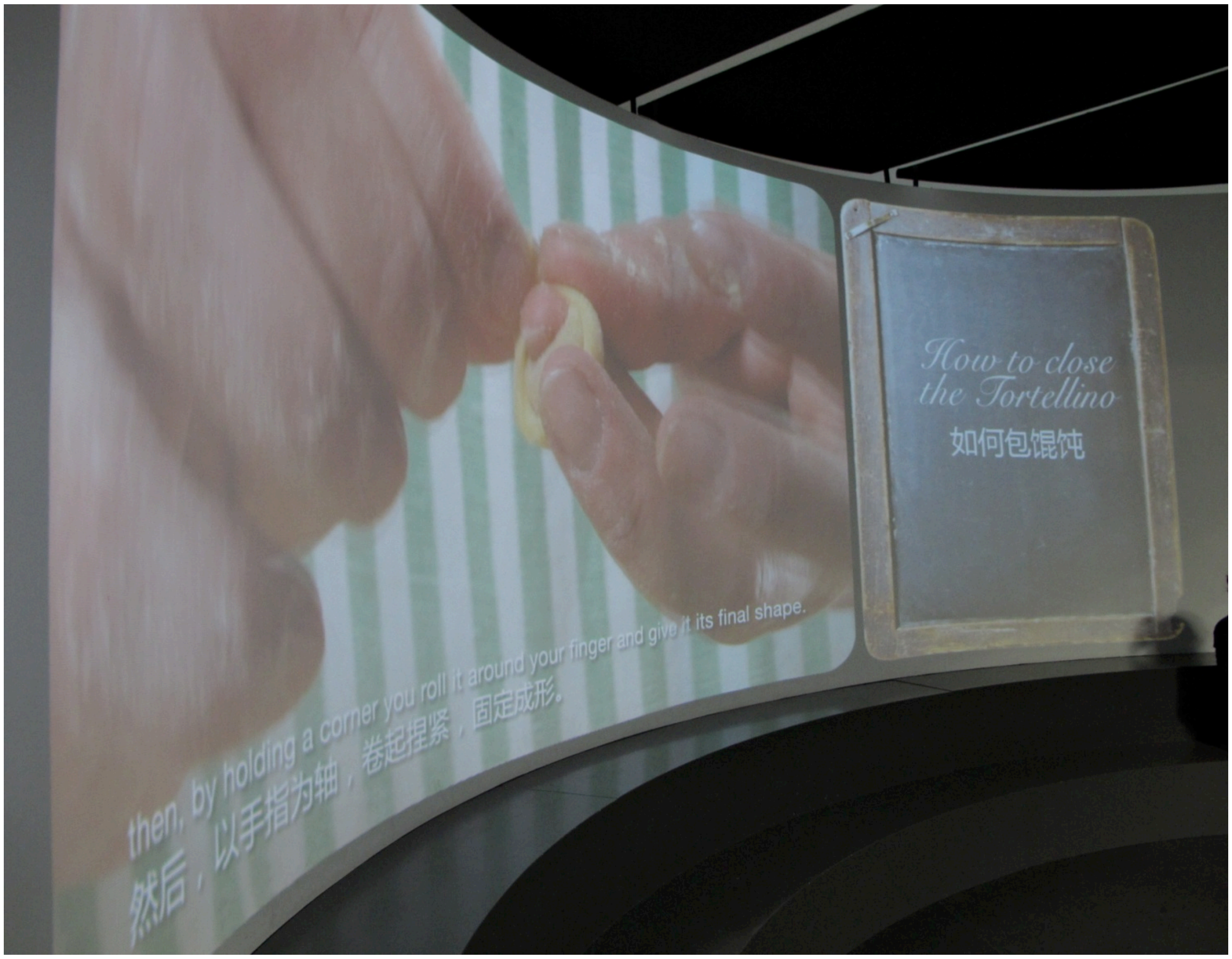
How to close the Tortellino
如何包馄饨
then, by holding a corner you roll it around your finger and give it its final shape.
然后，以手指为轴，卷起捏紧，固定成形。

Figure 3. ***Sfoglina*** **explaining actions.**

The pictures taken from our testbed, and shown in Figures 3, 4 and 5, represent an example of the final stage of our game. In particular, Figure 3 shows the *Sfoglina* explaining how to close a Tortellino. Figure 4, instead, depicts a digital representation of two hands trying to close a Tortellino. Once the player succeeds performing the correct movements, a stop motion video shows the final result (Figure 5).

5.3 Results: a Summary

Having many players assess our game at a public event has given us the opportunity of evaluating our algorithms in terms of recognition speed, accuracy and player satisfaction. In terms of speed, we observed that the overhead due to the gesture recognition algorithms never exceeds the limit of 30 milliseconds, thus providing a convincing result considering also that our reference implementation was written using the Java programming language. In terms of accuracy, we observed that false negatives (right actions not recognized as valid) were below 6%. Finally, in terms of player satisfaction, over hundreds of interviewed players, we have found that our game has been enjoyed by people of all ages, which gave it a score of 4.1 over a maximum of 5 when answering to the question “Did you enjoy the game?”. A more extensive report on all these performance factor can be found in [18].

6. CONCLUSION

In this work, we have shown that it is possible to support a wide class of gestural games devised for immersive environments and displayed in public spaces utilizing an off-the-shelf webcam and a robust gesture recognition software system. To witness the efficiency of our approach we developed a novel gestural game, termed the Tortellino X-Perience, that has been successfully demonstrated at the Shanghai World Expo, where hundreds of visitors have been challenged in preparing the best traditional Tortellini dish.

7. ACKNOWLEDGMENTS

We express our gratitude to all our undergraduate students (CS Class: Multimedia Systems and Applications, University of

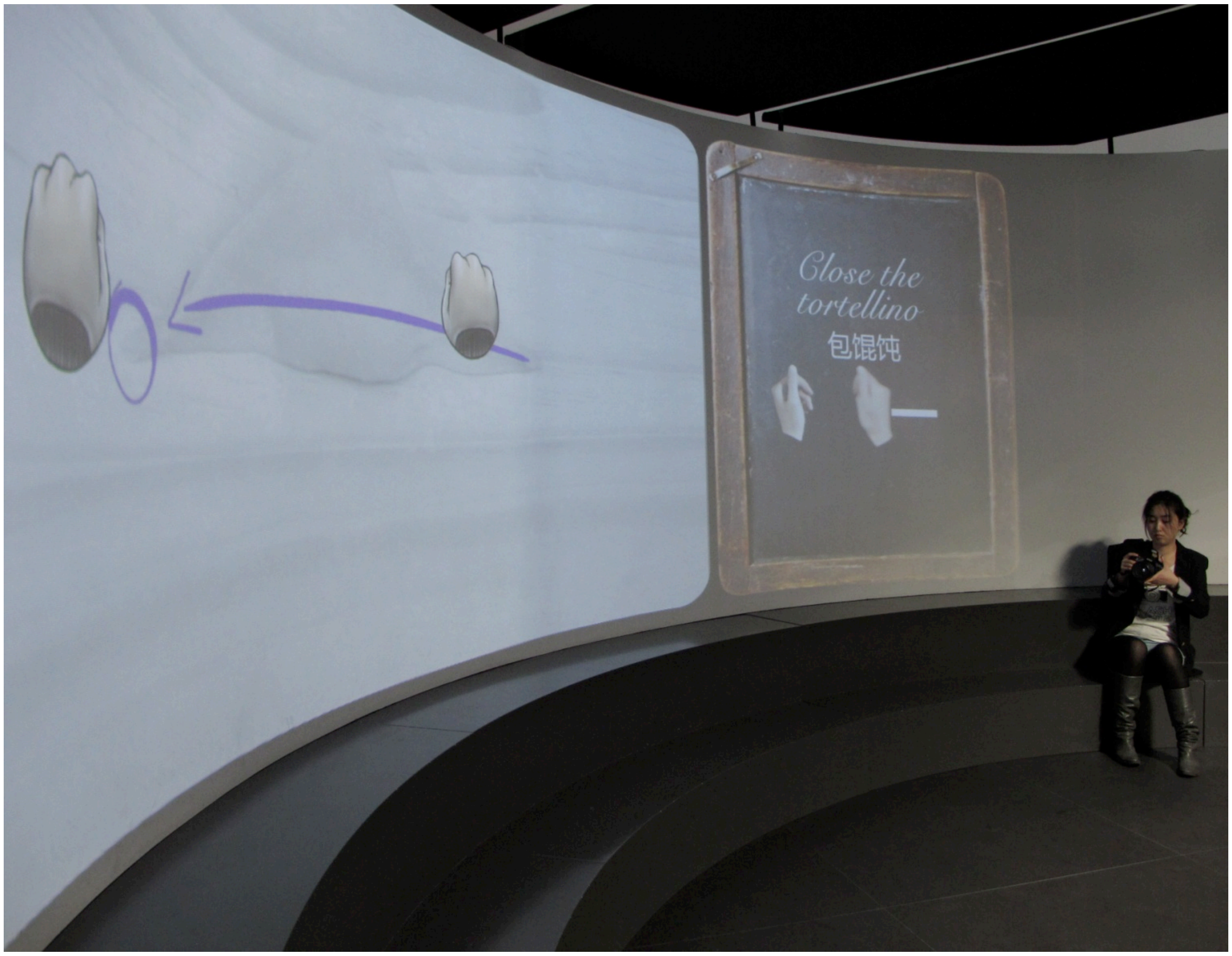

Figure 4. Actions performed by a player.

Bologna). Their names follow: Beatrice Bacelli, Cristian Bertuccioli, Carlo Brualdi, Antonio Casamassima, Giovanni De Marco, Andrea Di Toro, Luca Leoni, Andrea Marcomini, Giacomo Giorgi, Mirko Pedrini, Pierluigi Rocca and Diego Rodriguez. They took an important part in the implementation of our hand gesture recognition system. Special thanks are devoted to Marco Zanichelli and to Articulture, a Bologna-based media agency, that engineered and organized the Tortellino X-Perience event in Shanghai.

8. REFERENCES

[1] M. Roccetti, G. Marfia, "Recognizing Intuitive Pre-defined Gestures for Cultural Specific Interactions: An Image-based Approach," in *Proc. IEEE DENVECT'11,* Las Vegas (USA), 2011.

[2] M. Roccetti, G. Marfia, M. Zanichelli, The Art and Craft of Making the Tortellino: Playing with a Digital Gesture Recognizer for Preparing Pasta Culinary Recipes", *ACM Comput. Entertain.*, vol. 8, n. 4, 2010.

[3] Available online, accessed on the 2nd of February 2011: http://www.cs.unibo.it/~marfia/tortellinox-perience.mov

[4] Available online, accessed on the 2nd of February 2011: http://www.newscientist.com/blogs/onepercent/2011/01/computer-game-that-teaches-you.html

[5] T.B. Moeslund, A. Hilton, V. Krüger, "A Survey of Advances in Vision-based Human Motion Capture and Analysis," *Comput. Vis. Image Underst.*, Elsevier, New York, pp. 90-126, November 2006.

[6] S. Mitra, T. Acharaya, "Gesture Recognition: a Survey," *Trans. On Sys., Man and Cyb.*, IEEE, New York, pp. 311-324, May 2007.

[7] M. Roccetti, P. Salomoni, "A Web-based Synchronized Multimedia System for Distance Education," in *Proc. of the 2001 ACM symposium on Applied computing (SAC '01)*, New York, NY, USA, pp. 94-98, 2001.

[8] D. Campbell, "Physical gaming: Out of the Lap and into the Living Room," in *Proc. of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality,* Cambridge, 2008.

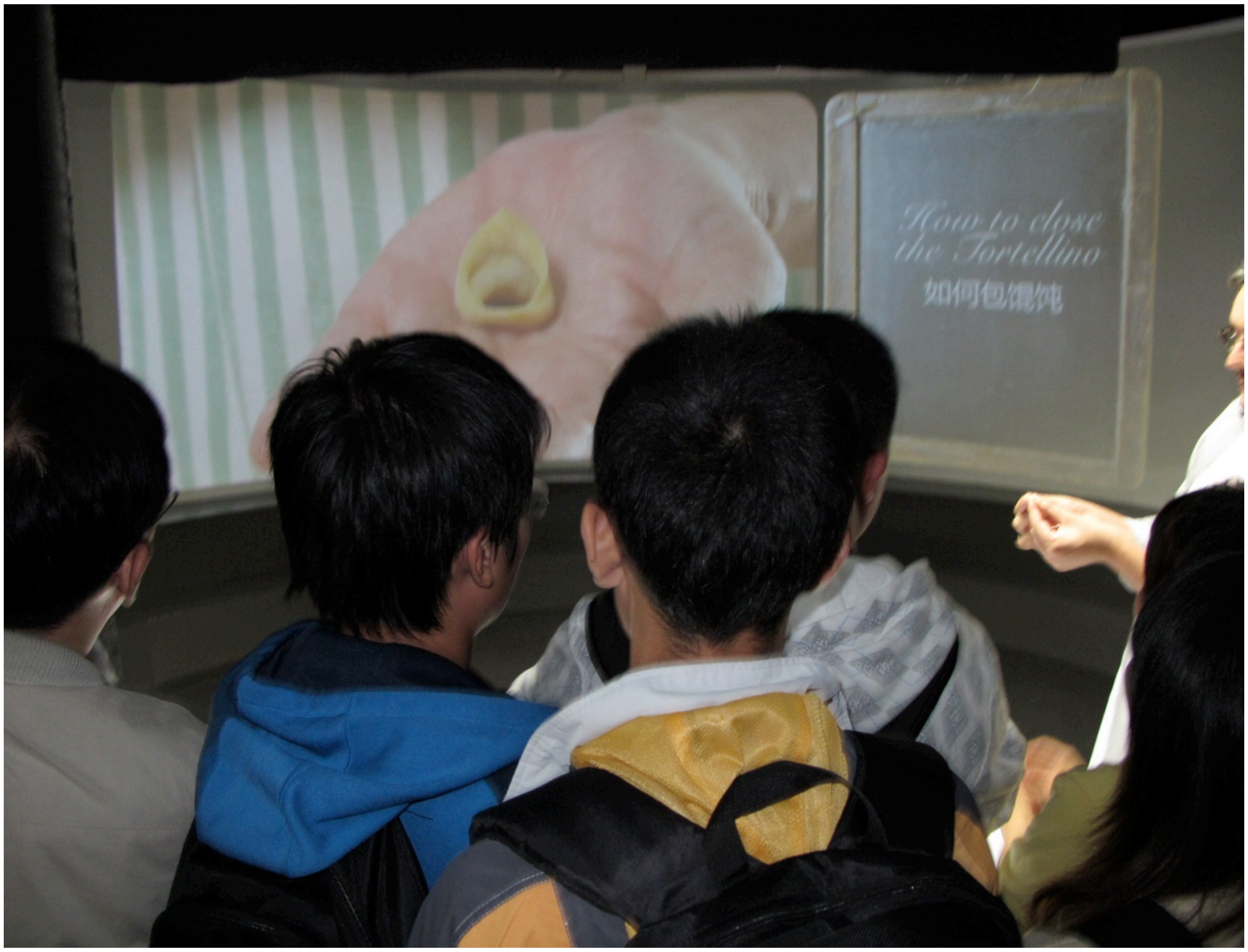

Figure 5. Result of a player's actions.

[9] M. Pasch, N. Bianchi-Berthouze, B. Van Dijk, A. Nijholt, "Movement-based Sports Video Games: Investigating Motivation and Gaming Experience," *Elsevier Entertainment Computing*, vol. 1, n. 2, pp. 49-61, 2009.

[10] S. Ferretti, M. Roccetti, "Fast Delivery of Game Events with an Optimistic Synchronization Mechanism in Massive Multiplayer Online Games," in *Proc. of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, Valencia, pp. 405-412, 2005.

[11] T. Selker, W. Burleson, "Context-aware Design and Interaction in Computer Systems," *IBM Systems Journal*, vol. 39, no. 3.4, pp. 880-891, 2000.

[12] R. Y. Wang, J. Popovic, "Real-time Hand-tracking with a Color Glove," *ACM Trans. Graph.,* vol. 28, n. 3, pp. 1-8, 2009.

[13] A. Rhalibi, M. Merabti, P. Fergus, S. Yuanyuan, "Perceptual User Interface as Games Controller," in *Proc. 5th IEEE Consumer Communications and Networking Conference,* Las Vegas, pp. 1059-1064, 2008.

[14] C. Harrison, A. K. Dey, "Lean and Zoom: Proximity-Aware User Interface and Content Magnification," in *Proc. of the twenty-sixth annual ACM SIGCHI conference on Human factors in computing systems,* Florence, pp. 507-510, 2008.

[15] D. Bannach, O. Amft, K.S. Kunze, E.A. Heinz, G. Troster, P. Lukowicz, "Waving Real Hand Gestures Recorded by Wearable Motion Sensors to a Virtual Car and Driver in a Mixed-Reality Parking Game," in *Proc. IEEE Symposium on Computational Intelligence and Games*, Hololulu, Hawaii, USA, pp. 32-39, 2007.

[16] L. Kratz, M. Smith, F.J. Lee, "Wiizards: 3D Gesture Recognition for Game Play Input," in *Proc. of the ACM 2007 conference on Future Play* (Future Play '07), Toronto, Ontario, Canada, pp. 209-212, 2007.

[17] E. Tse, S. Greenberg, C. Shen, C. Forlines, "Multimodal Multiplayer Tabletop Gaming," *ACM Comput. Entertain.,* vol. 5, n. 2, 2007.

[18] M. Roccetti, G. Marfia, A. Semeraro, "An All-In-Software Hand Gesture Recognition System for Immersive Gaming Experiences," submitted for publication to an international journal, 2011.