Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2015)
F. Jaillet, G. Zachmann, and F. Zara (Editors)

Using Personalized Finger Gestures for Navigating Virtual Characters

Christos Ouzounis[†1] Christos Mousas[‡2,3] Christos-Nikolaos Anagnostopoulos[§3] Paul Newbury[¶4]

[1]Dept. of Media Production, Ostwestfalen-Lippe University of Applied Sciences, Germany
[2]Dept. of Computer Science, Dartmouth College, USA
[3]Dept. of Cultural Technology and Communication, University of the Aegean, Greece
[4]Dept. of Informatics, University of Sussex, UK

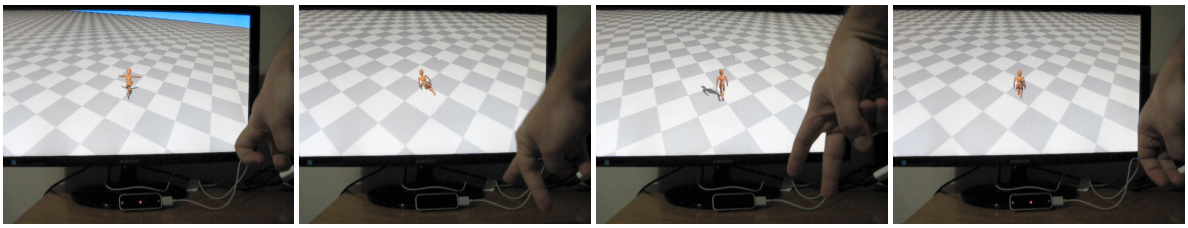


Figure 1: A simple example of finger gestures that animated a virtual character.

Abstract

In this paper, we present and evaluate a method to navigate a character into a virtual environment based on personalized finger gestures. The methodology that has been developed allows a user to generate his/her own finger gestures that are associated with the actions of a character. Specifically, in a pre-processing stage, the user wishes to perform specific gestures for specific actions of a character creating a dataset of gestures. During the runtime of the application, Dynamic Time Warping (DTW) and template matching methods were used to compute the similarity of the input and examples of gestures. The system recognizes the input gesture of a user and generates the motion required to navigate a character into the virtual environment. To demonstrate the efficiency and possible use of such a character navigation method, a number of users participated in an evaluation process. The results of the evaluation process indicate the possible use of personalized finger gestures for navigating a character into a virtual environment.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Giving life to a virtual character by means of various motion capture devices has been examined in detail in recent years [RTIK*14] [CEMTT98] [NWB*10]. Various techniques and solutions have been presented to control the motion of a virtual character in real-time. In addition, recent devices that allow robust motion tracking for compelling real-time interactions have been developed. Examples of these include the Nintendo Wii [Nit15], the Kinect [Mic15] system of Microsoft and the Leap Motion [Lea15b] controller. These devices make it possible to generate a variety of new methods to control virtual characters. Generally, the methodologies that use motion capture technologies to animate virtual characters are called “computer puppetry" [Stu98] [SLSG01].

† christos.ouzounis@stud.hs-owl.de
‡ christos@cs.dartmouth.edu
§ canag@ct.aegean.gr
¶ p.newbury@sussex.ac.uk

DOI: 10.2312/vriphys.20151329

However, a thorough examination is required to design computer puppetry interfaces from the perspectives of handling the information provided by the motion capture devices as well as from the perspective of understanding how users can interact with these interfaces during the human-character interaction process.

The method presented in this paper allows a user to personalize finger gestures to enable a character to navigate within a virtual environment. In the first step of this method, each user must perform a finger gesture for a specified motion of a character. The different gestures are captured using the Leap Motion interface. Then, by introducing a finger gesture recognition process based on Dynamic Time Warping (DTW) and template-based matching techniques, the system can recognize the user's input gesture efficiently. This procedure enables each user to animate the character based on his/her own gestures. Thus, the required personalization is achieved. Figure 1 provides examples of a character's synthesized motions based on finger gestures that a user has provided.

The efficiency of a system in which a virtual character is navigated by an interface while using finger gestures requires evaluation by users. Two different scenarios and three different approaches for navigating a virtual character were used in the evaluation process. In the first scenario, users sought to drive a character to a specified position within a virtual environment. In the second scenario, the users were called to navigate the character within an environment in which the basic task was to avoid an enemy that pursues it. Both of these scenarios used three different approaches to navigate the virtual character. In the first approach the users were asked to complete the tasks using their own gestures. In the second approach the users were asked to complete the tasks using a requested set of finger gestures. Finally, in the third approach, the users were asked to complete the tasks using a controller that is similar to these used in video game consoles.

The proposed methodology contributes mainly in two ways. First, it provides a finger gesture recognition process that generates the requested motion necessary to navigate a character into the virtual environments. Second, a scope behind the proposed implementation is the evaluation of such a character controller. Based on the two different evaluation scenarios, as well as on the different methods of navigating the character, the methodology demonstrates the possible use of finger gestures as an alternative mean to navigate characters located within virtual environments.

The remainder of the paper is organized as follows. Section 2 presents related work on computer puppetry, as well as on character animation methodologies based on finger gestures. Section 3 provides the methodology of the presented solution, by introducing a finger gesture recognition process. Section 4 contains the implementation of the methodology presented and the results obtained while evaluating the presented finger gesture recognition process. The experiments and the results obtained from a user study are shown in Section 5. Finally, conclusions are presented in Section 6.

2. Related Work

The character navigation into a virtual environment can be represented by a perspective of either the first person or third person. In the first person perspective, the player moves around the environment like a person who is exploring a new city. In the third person perspective, the player has a map view of the environment. Generally, each different perspective provides a different type of immersion, different types of possible information for a user [RI99] and different navigational abilities for the player [BC07].

To navigate virtual characters into virtual environments, the most common inputs are the joysticks that are located on game console controllers, and a combination of a mouse and a keyboard. By comparing the use of a mouse, keyboard, and joystick, it was found [LSV11] that the mouse outperforms the other inputs devices, although it offers only two degrees of freedom (one translation and one rotation).

In recent years, a variety of different methodologies depending on the input devices have been developed that enable users to control virtual characters. The required parameters for animating virtual characters could be retrieved by using accelerometers [MBT99] [IWZL09] [KSL13] that recognize the motion of a performer, microphones that recognize prosodic features of speech [LTK09] [Bra99], a variety of low cost interfaces [RTIK*14] [SH08a] [HBL11] that could be characterized as puppet interfaces or not interfaces [NATH11] [SH08b] [JPG*14] that are developed for this purpose.

To synthesize real-time motions using a database of motion capture sequences, a key issue for the scientific community is the development of policies for synthesizing the desired motions, while reducing the number of sensors or amount of input information required. Many different techniques and a variety of interfaces have been used in an attempt to solve this problem. In [SH08a] a solution is proposed to control virtual characters that use two accelerometers: the Wiimote controllers. In that method, physically simulated characters are used to perform different actions, such as walking, running, and jumping, based on an input control signal. Another solution that extracts information from internal sensors was proposed in [LWC*11]. The key idea behind this approach is to pre-record motions in a motion capture database, and then to construct a series of online local dynamic models that are needed to construct full-body human motion in a maximum a posteriori framework. There is another solution [NATH11] that uses a reduced number of sensors to synthesize a virtual character's motion. In this approach, after the puppet interface has been designed and the motion approximated, the system recognizes which action is

performed and, based on a collection of motion sequences, attempts to determine which action best matches the input signals.

Like the presented methodology, a variety of approaches have been proposed that permit a user to animate a character using his fingers. The use of image moments and optical flow to detect the movements of a user, allows the motions to be directly mapped to the character's motions [FAB*98]. By using data-gloves in [LZK04] [SZ93], the finger joint angles are mapped to the character's skeleton, resulting in the synthesis of simple motions. Also, by using a data-glove, the hand and fingers in [IIO09] are mapped to the skeleton of the character. Then, by performing finger gestures, the system recognizes which movements of the character should be performed. [Oka03] introduced physical constraints related to gravity and foot contact with the ground to generate motions that were as natural as possible. The basic disadvantage of this approach, however, is that the hands, feet, and neck of the character can move only in one direction. In [WP09], the fingertip position was used as a constraint on a character's foot position. In another recent solution [LK12], information about a user's fingers positions is captured using a touch-based surface. The system determines which motion best matches the user's input and then attempts to generate a desirable result. However, the basic disadvantage of [LK12] compared to the aforementioned methodologies is its inability to synthesize the required motion in real-time.

Generally, when dealing with motion capture devices for animating a virtual character, the basic disadvantage is the synthesis of inconsistent animation. This is due to the jittering resulting from the input signal's noise. In addition, the majority of the aforementioned methods indicate the generated motions are based only on simple walking actions of a character, instead of a continuous locomotion sequence in which the character performs a variety of actions. This is a general problem, since the finger joint angles are mapped to body parts of a character. However, in the proposed approach, this limitation is overcome by using finger gestures instead of finger joint angles to animate a virtual character. Moreover, it is believed that every user should be able to animate a virtual character using his/her own variations of finger gestures. Thus, the presented system provides a personalization process for each user's finger gestures. This personalization process of input gestures can be an advantage. The methodologies presented previously have given less attention to evaluating the use of finger gestures to animate virtual characters. It is assumed that such an evaluation would be quite beneficial. Therefore, it should be noted that the presented finger gesture recognition process, the personalization process of the finger gestures and the evaluations that were conducted in order to understand the possible use of such a method for navigating virtual characters are the main contributions that are presented in this paper.

3. Methodology

In this section, the development of a finger gesture recognition process that was used in our methodology is presented. Specifically, the finger gesture recognition process is separated into the following four steps: (i) finger gesture motion capture, (ii) finger posture representation, (iii) gesture clustering, and finally (iv) gesture recognition. The process of the presented system developed for the finger gesture recognition methodology is illustrated in Figure 2.

3.1. Capturing Finger Gestures

Using the Leap Motion controller in a preprocessing stage, a user's input gestures for each separate action of a character are captured. In the methodology presented, a captured finger gesture is represented by a series of frames, where each captured frame represents the three-dimensional coordinates of the finger joints at a certain time. A captured finger gesture is represented as $G = \{g_t^j | t = 1,...,T; j = 1,...,J\}$ where t denotes the number of frames and j denotes the joint index of a hand. During the capturing process, each user performs repetitions of a specific gesture for each action of a character for ten seconds. Thus, a motion dataset that contains the representations of a single gesture is generated. It should be noted that, with the Leap Motion, 19 degrees of freedom (DOF) of the user's hand are captured.

3.2. Posture Representation of Gestures

In the methodology that is presented, each captured gesture is parameterized in an exponential form. This step is quite important because it enables the system to avoid "gimbal" lock, discontinuities and ball-and-socket joints complications that are related to the captured motion data. The process is accomplished by parameterizing the three-dimensional angle of each captured DOF of the user's hand in an Exponential Map as shown in [Gra98]. The Exponential Map is generated from the corresponding Quaternion representation of each joint rotation. This parameterization can be expressed as:

$$EMP(g_t^j) \rightarrow \bar{g}_t^j \quad (1)$$

Therefore, a set of transformed finger gestures is denoted as $\bar{G}$.

3.3. Gesture Clustering

Certain postures of finger gestures may be semantically similar. However, these postures may not be numerically similar. Therefore, it is necessary to resolve this problem that may affect negatively the finger gesture recognition process. Generally, identifying and grouping similar finger postures manually is a time consuming process. In the methodology presented a grouping of $\bar{G}$ into k cluster is achieved by use of a centroid method, the $k-means$ clustering [Mac67]. It

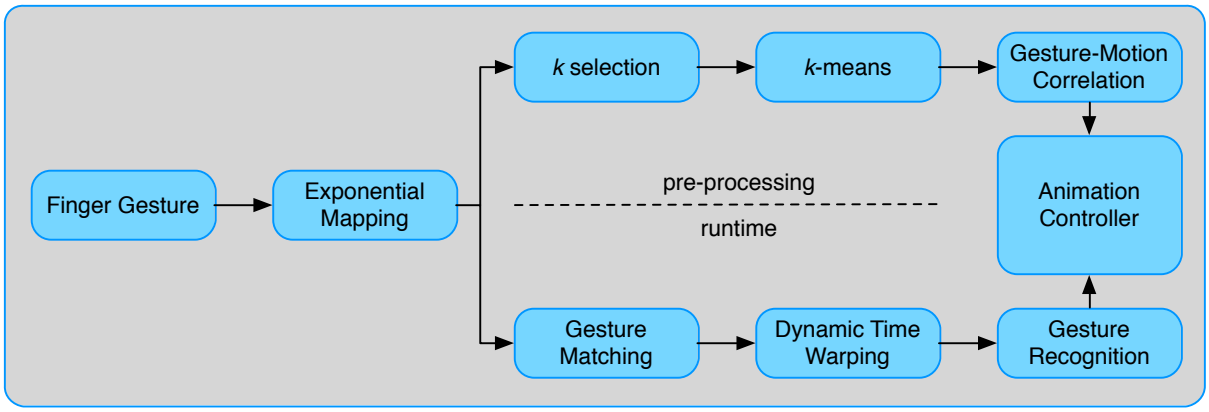


Figure 2: The process of the presented system consists of two stages: the preprocessing, where the finger gestures of a user are analyzed and mapped to the animation controller, and the runtime, where the finger gestures of a user are recognized and are mapped to the animation controller for allowing a user to navigate a virtual character.

should be noted that the $k-means$ clustering method was chosen since the selection criteria of the median element is performed more efficiently and also is less expensive computationally.

In the methodology presented, it is necessary to cluster a finger gesture in to k clusters, where each cluster contains a posture similar to that of a gesture. It should be noted that as a by-product of the $k-means$ process, each cluster characterizes a phase of the finger gesture. Among others, a difficulty with the $k-means$ clustering method is the selection of k. This problem can be overcome by using a method called "Elbow" that is proposed in [KS96]. This method uses the percentage of variance determined as a function to select the appropriate k based on automatic selection of the Elbow criterion.

In this case, in order to select a delegate for each cluster, a ranking scheme is used for each pose of a gesture, according to the Manhattan distance metric [Kra73]. The distance, D, between any two postures of finger gesture $\bar{g}_m$ and $\bar{g}_n$ in a cluster is measured using the total distance among corresponding joints, which is defined as:

$$D(\bar{g}_m, \bar{g}_n) = \sum_{j=1}^{J} \|\bar{p}_m^j - \bar{p}_n^j\| \qquad (2)$$

The registered finger posture, $E_{e,k}$, has the lowest average distance between a cluster grouping of postures. Therefore, it can be said that $k-means$ provides an elegant method to segment a motion sequence that contains a finger gesture. For the method that is presented, a label is assigned to each example finger posture based on where it appears in the captured motion sequence. The same label is used for all the postures of a specific finger gesture. Thus, for each gesture, we retain k number of registered finger postures and a label to represent each gesture type. For simplicity, let $S = \{s_e | e = 1,...,E\}$ be the complete gesture set, where $s_e = \{E_{e,k} | k = 1,...,K_e\}$ is a specific gesture that comprises K number of finger postures for a gesture type e.

3.4. Finger Gesture Recognition

During the application's runtime, the system should be able to recognize the input finger gesture of a user at every time instance. In the presented method, a temporal matching of an input segment of a finger gesture derived from frames $f-1$ and f captured frames was used. Specifically, given an input motion segment $g = \{g_t^j | t = f-1, f; j = 1,...,J\}$, each $t-th$ hand posture is treated individually. It should be noted that it is assumed that the input segment forms a gesture in time order. Moreover, it is also assumed that each input finger gesture is correlated to one of the captured gestures of a user contained in our dataset.

For the finger gesture recognition, each posture of the fingers is first parameterized in exponential form by using Equation 1. Then, by using the distance function represented in Equation 2 the similarity between the input finger gesture and the pre-recorder gestures contained into a database is determined. The minimum of the summation of distances is computed by:

$$L_t := \min\{D(\bar{g}_t, E_{e,k}) | e = 1,...,E; k = 1,..,K\} \qquad (3)$$

Therefore, L_t is the winning label that is assigned to the observed pose. It should be noted that a unique label is assigned to each assigned finger gesture.

At each time instance, template matching is performed to determine the gesture type. Specifically, given an observed label $L_{f-1 \to f}$, we compare the sequence to each action model s_e to find the best pattern match. The winning type L_t is determined by the minimum Dynamic Time Warping (DTW) cost with respect to the Itakura constraint [Ita75],

which is given as:

$$\mathscr{L}_t := \min\{DTW(L_{f-1\to f}, s_e)|e = 1,...,E\} \quad (4)$$

Finally, by having each gesture classified, recognition of the input gesture is defined based on the accumulative voting that is indicated in $\mathscr{L}$.

4. System Implementation and Evaluation

Here, the implementation of the presented system, and the results obtained while evaluating the finger gesture recognition process are presented.

4.1. Implementation

The methodology that is presented is implemented on an Intel i7 2.2 GHz processor with 8 GB RAM. The user's hand fingers are captured using the Leap Motion interface [Lea15b] with its associated SDK (version 2.2) [Lea15a]. The application was developed in the Unity3D game engine [Uni15a]. The SmartBody animation system for Unity3D provided by [Uni15b] was used for the character animation. The animation system of a character consists of basic motion: walking (forward, backward, right and left), running (forward, backward, right and left), jumping and staying idle. Finally, the Recast/Detour library [Mon15] for collision avoidance with the environment was used. Our system runs in real-time at an average frame rate of 49 frames per second, and the latency of our implementation is 0.022 seconds.

4.2. Evaluating Finger Gesture Recognition

The estimation rate of correct recognition of each finger gesture was computed. This evaluation was conducted by use of the leave-one-out cross validation method. Specifically, the captured finger gestures used for the recognition process were split into small segments. By using the small motion segment as a reference input gesture, and also by using the presented finger gesture recognition process, the estimation rate was computed. This procedure was repeated for each different gesture. Thus, a class confusion matrix that provides the allocation of the estimation rate is shown in Figure 3. As can be seen, the proposed methodology seeks to estimate correctly at an average rate equal to 91.6%.

5. User Study

We designed a study to measure the efficiency of the finger gesture character navigation controller. Participants were asked to navigate a character within a virtual environment by using either their own set of gestures or a predefined set of gestures to achieve the given goal. Moreover, in order to better understand the capability of using such a controller, users were asked to complete the same tasks with use of a controller that is similar to these used in game consoles (a Logitech Dual Action, as illustrated in Figure 4). Then, they were asked to identify their preferred inputs. In our experimental study, 15 individuals (9 males with a median age of 24 and 6 females with a median age of 22) tested the presented methodology. Their game experience varied from novice to experienced user, with an average score of 4.2 on a scale on which 1 signifies a novice and 7 signifies an expert. All participants were novices with finger gesture controllers for character navigation.

5.1. Experimental Methodology

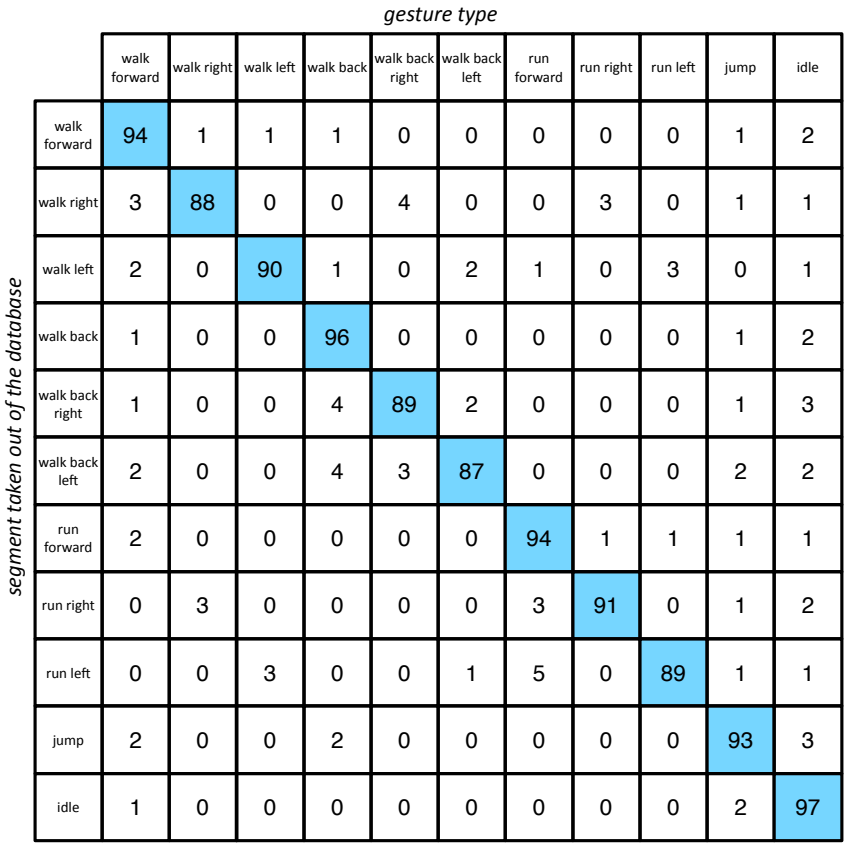
The evaluation process that was conducted is the following. Participants sat in a quiet room in front of a desktop computer with both the application and the motion capture device installed. The experimenter explained to the participants that they would be using gestures involving either the left hand or right hand (depending on each participant's preference) in order to navigate a character. The experimenter also explained that, for each attempt in both scenarios (see Section 5.2), each participant should use his/her own hand gestures in the first attempt, a predefined set of gestures (see Section 5.3) in the second attempt and a controller (see Figure 4) in the third attempt. It should be noted that a personalization process was used for the predefined set of gestures. Thus, even if each user was requested to use specific gestures, this personalization process would allow the system to recognize the user's variations of the finger gestures.

For the first scenario, the experimenter told the participants that they should try their best to avoid hitting the walls. For the second scenario, the experimenter asked the participants to try to avoid hitting the enemy that pursued the character.

Before the start of the evaluation process, participants tested each different navigation approach separately in an environment that was free of objects and obstacles for two minutes. This enabled each participant to practice the different character controller approaches prior to the task. During the evaluation process, the time (in seconds) to complete the first scenario and the number of times that the wall was hit were computed. For the second scenario, the number times that the pursuing enemy hit the character were computed. The results were computed for each of the three different methods of navigating the virtual character. After completing the evaluation process, each participant evaluated the intuitiveness of each different method and the difficulty of controlling a virtual character. Finally, users were asked to select their preferred methodology for navigating a virtual character and to explain their preference.

5.2. Evaluation Scenarios

We developed two scenarios to evaluate the efficiency of using the presented methodology. In the first scenario, which is illustrated in Figure 5a, the user is asked to navigate the character to the goal position. In the second scenario, which



segment taken out of the database \ gesture type	walk forward	walk right	walk left	walk back	walk back right	walk back left	run forward	run right	run left	jump	idle
walk forward	94	1	1	1	0	0	0	0	0	1	2
walk right	3	88	0	0	4	0	0	3	0	1	1
walk left	2	0	90	1	0	2	1	0	3	0	1
walk back	1	0	0	96	0	0	0	0	0	1	2
walk back right	1	0	0	4	89	2	0	0	0	1	3
walk back left	2	0	0	4	3	87	0	0	0	2	2
run forward	2	0	0	0	0	0	94	1	1	1	1
run right	0	3	0	0	0	0	3	91	0	1	2
run left	0	0	3	0	0	1	5	0	89	1	1
jump	2	0	0	2	0	0	0	0	0	93	3
idle	1	0	0	0	0	0	0	0	0	2	97

Figure 3: The class confusion matrix shows the allocations of the presented finger gesture recognition process.

is illustrated in Figure 5b, the user is asked to avoid an enemy that pursues it for a time period of two minutes. In both scenarios, the environments consisted of walls that the character was not permitted to pass over or through, and obstacles that the character was meant to jump over.

5.3. Predefined Set of Gestures

The predefined set of gestures that were used for the evaluation process is presented in this section. This predefined set consists of gestures that mimic each of the character's basic actions by using only two fingers of a hand: the middle finger and the index finger. Figure 6 shows four of the gestures that were used for walking, running, jumping and idle motions.

5.4. Results

Having presented the experimental methodology to evaluate such a character navigation mechanism, our findings are presented in the following subsections. It should be noted that in the following subsection the subscripts u, r and c denote the user-defined gestures, the requested gestures and the game consol controller respectively.

5.4.1. Time to Completion and Wall Hits

For the first scenario, as well as using the two different gesture types and the game console controller, the time that participants needed to complete the given tasks was recorded.

Although participants used their own gestures to navigate a virtual character, a mean time of $m_u^{time} = 67.1$ seconds was required. The standard deviation was estimated at $\sigma_u^{time} = 9.3$. When participants used the requested set of gestures, the mean time was $m_r^{time} = 77.8$ seconds with a standard deviation of $\sigma_r^{time} = 10.5$. Finally, when using the game console controller, the mean time was $m_c^{time} = 63.5$ seconds and the standard deviation is $\sigma_c^{time} = 7.4$.

Generally, in observing the aforementioned results, one can state that participants navigated a virtual character to a goal position more quickly when using the game console controller than with the two different finger gesture sets. Moreover, one can say that the participants navigated a character more quickly to a goal position when using their own set of gestures, instead of the requested set of gestures. However, by using a pairwise $t-test$ for the time to completion, the following should be noted. Participants did not navigate the character to a goal position noticeably faster

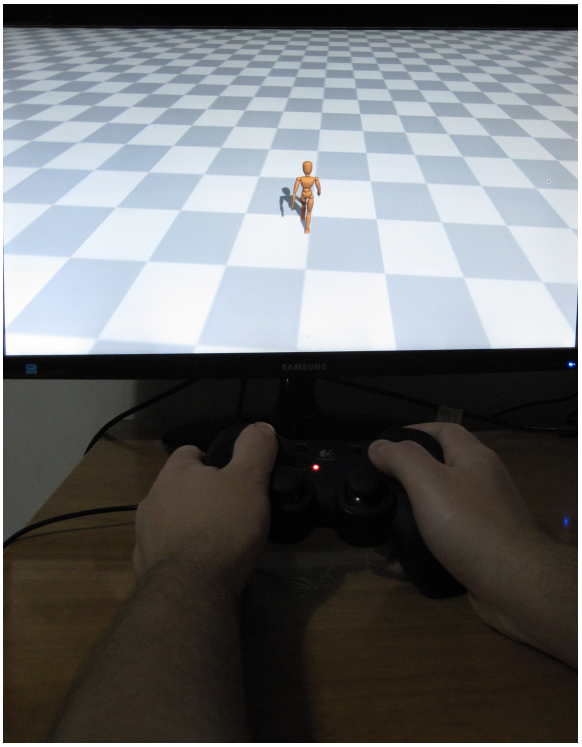


Figure 4: The controller used in the presented evaluation process.

$(t(14) = 2.23,\ p = 0.07)$ when comparing the game console controller and their own set of preferred gestures. When the participants used the predefined set of gestures, the time for completion was higher than when they used their own gestures $(t(14) = -1.77,\ p = 0.003)$ and the game console controller $(t(14) = -2.11,\ p = 0.001)$.

5.4.2. Wall Hits

For the first scenario, it was also recorded the wall hits while using the two different gesture types and the game console controller. Specifically, while measuring the wall hits, the following results were obtained. When users employed their own gestures, the participants hit the walls $m_u^{hits} = 9.4$ times with a standard deviation of $\sigma_u^{hits} = 4.2$. When participants used the predefined set of gestures, the wall hits were $m_r^{hits} = 11$ with a standard deviation of $\sigma_r^{hits} = 6.5$, and when using the game console controller, the wall hits were $m_c^{hits} = 2.3$ times with a standard deviation of $\sigma_c^{hits} = 1.2$.

Based on these results, one can state that participants when using the game console controller hit the walls fewer times than with the two different finger gesture methods. A pairwise $t-test$ strengthens the aforementioned observation since when users employed their own gestures $t(14) = -2.17,\ p < 0.006$, and when participants used the predefined set of gestures $t(14) = -2.93,\ p < 0.004$ while both approaches that used gestures when compared to the game console controller.

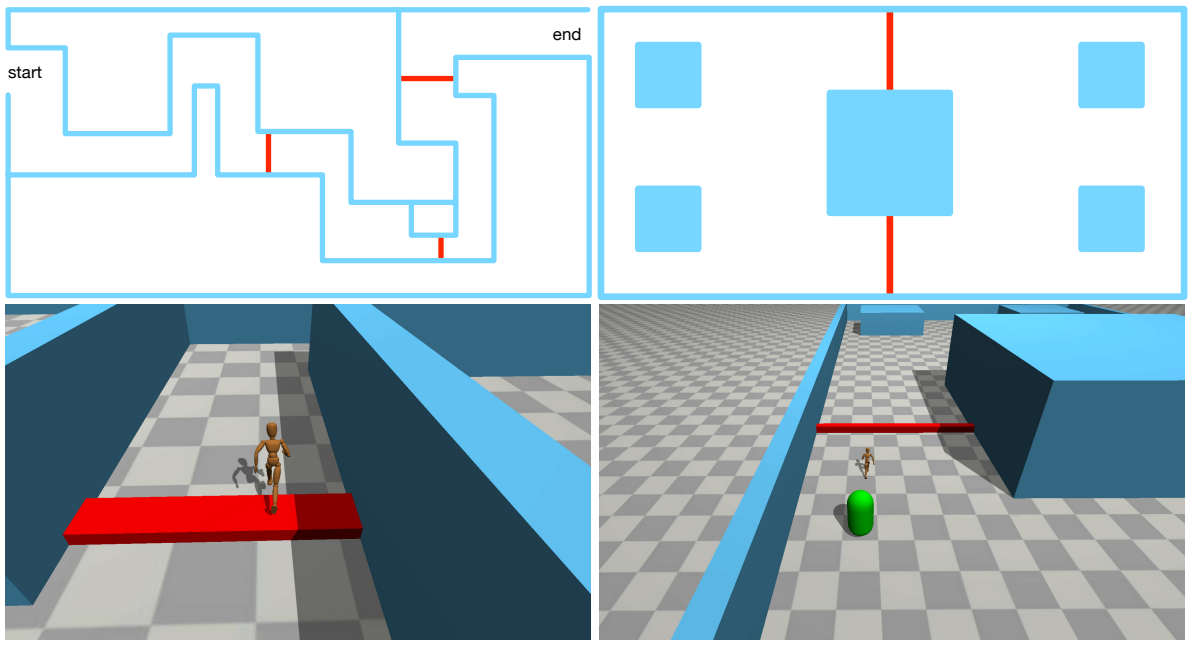
5.4.3. Avoiding the Enemy

Here, we present our findings after evaluating the second scenario. Specifically, in this scenario, the number of hits that occurred between a character and an enemy was computed using the two different finger gesture methods and the game console controller. The following results were obtained for a time of two minutes. When participants used their own gestures, the mean number of hits computed was $m_u^{enemy} = 16.2$ with a standard deviation of $\sigma_u^{enemy} = 7.5$. When participants used the requested set of gestures, the mean number of hits computed was $m_r^{enemy} = 21.3$ with a standard deviation of $\sigma_r^{enemy} = 11.1$. Finally, when participants used the game console controller, the mean number of hits was $m_c^{enemy} = 4.1$ with a standard deviation of $\sigma_c^{enemy} = 2.7$.

After examining the aforementioned results, one can state the following. Generally, when participants used the game console controller they avoided the enemy that pursued the character more easily. This observation is strengthened by a $t-test$ that showed that there is no similarity between the game console controller and the users' gestures $(t(14) = -3.94,\ p = 0.002)$, as well as with the requested gestures $(t(14) = -4.55,\ p < 0.001)$. However, by comparing the two different gesture types that the participants used, it is also clear (it is also confirmed by a $t-test$: $t(14) = -3.16$, $p = 0.003$) that the participants are more capable of avoiding an enemy when using their own gestures, instead of the requested gestures.

5.4.4. Intuitiveness and Difficulty

After the participants had completed all of the requested tasks, they were invited to evaluate the intuitiveness and the difficulty of each of the three different methodologies for navigating a virtual character. They did so by completing a questionnaire that contained a $7-point$ Likert scale (1 is difficult, 7 is easy). The results obtained from this evaluation are the following. When using their own gestures, the mean was $m_u^{difficulty} = 4.1$ with a standard deviation of $\sigma_u^{difficulty} = 1.1$. When using the predefined set of gestures, the average difficulty measured was $m_r^{difficulty} = 5.6$ with a standard deviation $\sigma_r^{difficulty} = 0.9$. Finally, when using the game console controller, the mean was $m_c^{difficulty} = 3.6$ with a standard deviation of $\sigma_c^{difficulty} = 0.7$. A pairwise $t-test$ found that there it is quite easy to navigate a character by using both their own gestures and the game console controller $(t(14) = 4.21,\ p = 0.08)$. However, there is a distinction between the aforementioned two methods with the predefined set of gestures: $t(14) = -3.04,\ p < 0.02$ between requested and user defined gestures, and $t(14) = -2.88,\ p < 0.03$ between the requested gestures and the game console controller.



(a) (b)

Figure 5: The two different scenarios implemented for the evaluation process proposed. In the first scenario (a) users sought to drive a character to a specified position (end) within a virtual environment while avoiding wall hits. In the second scenario (b) the users were called to navigate the character within an environment in which the basic task was to avoid an enemy (green capsule) that pursues it.

Figure 6: The predefined set of gestures used in the presented evaluation process. From right to left, they are walking, running, jumping and idle.

5.4.5. Navigation Preference

In the questionnaire that was distributed to participants after they had finished the evaluation process, they were asked what their preferred controller was. Specifically, they were asked first to choose one of the three different methods (their own gestures, the requested set of gestures or the game console controller). Then, they were asked to choose between their own gestures and the requested set of gestures. Next, they were asked to choose between their own gestures and the game console controller and, finally, to choose between the requested set of gestures and the game console controller.

The results of this questionnaire are as follows. Of the three different methods, the use of their own gestures received 39% of responses, the use of requested gestures received 13% and the use of the controller received 48% of responses. Of the user defined gestures and the requested gestures, user defined gestures received 77% of responses and requested gestures received 23%. When asked to choose between their own gestures and the game console controller, their own gestures received 49% of responses and the use of game console controller received 51%. Finally, when the respondents were asked to choose between the requested set of gestures and the game console controller, the requested set of gestures received 18% of responses and the use of game console controller received 82% of responses.

Based on the aforementioned results the following should be noted. Participants found that the use of the game console controller is “easier" or “more intuitive". Generally, since the use of game console controllers is closer to their own previous experiences, less time as well as less effort is required to learn how to use it in order to interact by navigating a character into a virtual environment.

Besides the aforementioned evaluation, the additional questions provided information that is useful to understand the use of finger gestures as an alternative method for navigating a virtual character. Specifically, when participants were asked to choose between their own and the requested

gestures, the vast majority of participants indicated that they prefer to user their own gestures. This means that the willingness of a participant to control a character by using his/her own personalized finger gestures makes him/her more comfortable. This is because it is easier for a participant to remember, as well as to perform his/her own gesture without much effort. When comparing the use of participant's own gestures and the game console controller, the gap between the two methodologies decreases. These results showed that in the first question, respondents choose the use of requested gestures, whereas when they must choose between their own gestures and the game console controller they preferred their own gestures. This means that there is a potential willingness of the participants to continue using a finger gesture methodology, instead of a game console controller. In the final question, participants expressed their preference to use the game console controller, instead of the requested set of finger gestures. In this case, contrary to the results of the previous question, participants who had chosen to use their own set of gestures now changed and chose the game console controller. Therefore, this result showed that there is a possible willingness of participants to choose an easier methodology, the game console controller, where they have prior knowledge of such a controller, instead of a methodology that requires effort to learn how to use it.

6. Conclusions

The use of computer puppetry techniques to handle the motions of virtual characters with novel interfaces is a field that requires further research, especially in cases where the interfaces are used to generate motions employ alternative approaches, such as the finger gesture approach examined in this study. The development of novel interfaces requires the completion of evaluation scenarios by a number of users. In addition, as motion capture technologies are of interest in the development of interfaces for gaming applications, these interfaces should be compared to commonly used controllers that the majority of applications use. This was a major goal of our paper. Therefore, the evaluation process used in this study sought to assess the efficiency of finger gesture recognition for virtual character navigation.

The results presented in this paper show that, even if finger gestures can be used for navigating a virtual character within an environment, the participants are more familiar with commonly used controllers. This is especially true in cases where participants are requested to use a predefined set of gestures instead of their own. However, as the divergences between performance using a game console controller and personalized finger gestures are not great, we believe that such a methodology for navigating a virtual character may be employed efficiently in various virtual reality navigation and gaming scenarios.

Although our study demonstrates the feasibility of using novel controllers in two different scenarios, their application in more complex scenarios has yet to be examined. In future implementations, we would like to extend the proposed approach by not only generating simple motions of a character that are related to locomotion, but also by generating a large number of different actions, such as actions encountered in shooting and fighting games. In situations in which a character performs a variety of combined motions, such as simultaneously evolving jumping and kicking actions, it would be interesting in a future implementation to use finger gestures of both hands to generate more complex actions. Moreover, additional evaluations to indicate the potential use of such complex approaches should be examined in depth.

References

[BC07] BURIGAT S., CHITTARO L.: Navigation in 3d virtual environments: Effects of user experience and location-pointing navigation aids. *International Journal of Human-Computer Studies 65*, 11 (2007), 945–958. 2

[Bra99] BRAND M.: Voice puppetry. In *Annual Conference on Computer Graphics and Interactive Techniques* (1999), ACM SIGGRAPH, pp. 21–28. 2

[CEMTT98] CAVAZZA M., EARNSHAW R., MAGNENAT-THALMANN N., THALMANN D.: Motion control of virtual humans. *IEEE Computer Graphics and Applications 18*, 5 (1998), 24–31. 1

[FAB*98] FREEMAN W. T., ANDERSON D. B., BEARDSLEY P. A., DODGE C. N., ROTH M., WEISSMAN C. D., YERAZUNIS W. S., KAGE H., KYUMA K., MIYAKE Y., ICHITANAKA K.: Computer vision for interactive computer graphics. *IEEE Computer Graphics and Applications 18*, 3 (1998), 42–53. 3

[Gra98] GRASSIA F. S.: Practical parameterization of rotations using the exponential map. *Journal of graphics tools 3*, 3 (1998), 29–48. 3

[HBL11] HA S., BAI Y., LIU C. K.: Human motion reconstruction from force sensors. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (August 2011), ACM SIGGRAPH/Eurographics, pp. 129–138. 2

[IIO09] ISMAIL N. I. N., ISHIGURO K., OSHITA M.: Real-time character motion control using data gloves. In *International Conference on Computer Games, Multimedia and Allied Technology* (2009), -, pp. 307–314. 3

[Ita75] ITAKURA F.: Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing 23*, 1 (1975), 67–72. 4

[IWZL09] ISHIGAKI S., WHITE T., ZORDAN V. B., LIU C. K.: Performance-based control interface for character animation. *ACM Transactions on Graphics 28*, 3 (2009), Article No. 61. 2

[JPG*14] JACOBSON A., PANOZZO D., GLAUSER O., PRADALIER C., HILLIGES O., SORKINE-HORNUNG O.: Tangible and modular input device for character articulation. *ACM Transactions on Graphics 33*, 4 (2014), Article No. 82. 2

[Kra73] KRAUSE E. F.: Taxicab geometry. *The Mathematics Teacher* (1973), 695–706. 4

[KS96] KETCHEN D. J., SHOOK C. L.: The application of cluster analysis in strategic management research: an analysis and critique. *Strategic management journal 441-458*, 17 (1996), 6. 4

[KSL13] KIM J., SEOL Y., LEE J.: Computer animation and virtual worlds. *Human motion reconstruction from sparse 3D motion sensors using kernel CCA-based regression 24*, 6 (2013), 565–576. 2

[Lea15a] LEAP MOTION: Developer sdk version 2.2. https://developer.leapmotion.com/, accessed 25/01/2015. 5

[Lea15b] LEAP MOTION: Finger motion capture system. https://www.leapmotion.com/, accessed 25/01/2015. 1, 5

[LK12] LOCKWOOD N., KARAN S.: Finger walking: Motion editing with contact-based hand performance. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation Symposium on Computer Animation* (2012), ACM SIGGRAPH/Eurographics, pp. 43–52. 3

[LSV11] LAPOINTE J. F., SAVARD P., VINSON N. G.: A comparative study of four input devices for desktop virtual walkthroughs. *Computers in Human Behavior 27*, 6 (2011), 2186–2191. 2

[LTK09] LEVINE S., THEOBALT C., KOLTUN V.: Real-time prosody-driven synthesis of body language. *ACM Transactions on Graphics 28*, 5 (2009), Article No. 172. 2

[LWC*11] LIU H., WEI X. K., CHAI J., HA I., RHEE T.: Real-time human motion control with a small number of inertial sensors. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2011), ACM SIGGRAPH, pp. 133–140. 2

[LZK04] LAM W.-C., ZOU F., KOMURA T.: Motion editing with data glove. In *ACM SIGCHI International Conference on Advances in computer entertainment technology* (2004), ACM SIGCHI, pp. 337–42. 3

[Mac67] MACQUEEN J.: Some methods for classification and analysis of multivariate observations. In *Berkeley symposium on mathematical statistics and probability* (1967), University of California Berkeley, pp. 281–297. 3

[MBT99] MOLET T., BOULIC R., THALMANN D.: Human motion capture driven by orientation measurements. *Presence 8*, 2 (1999), 187–203. 2

[Mic15] MICROSOFT: Kinect for windows. http://www.microsoft.com/en-us/kinectforwindows/, accessed 25/01/2015. 1

[Mon15] MONONEN M.: Recast/detour navigation library. http://code.google.com/p/recastnavigation/, accessed 25/01/2015. 5

[NATH11] NUMAGUCHI N., ATSUSHI N., TAKAAKI S., HODGINS K. J.: A puppet interface for retrieval of motion capture data. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2011), ACM SIGGRAPH/Eurographics, pp. 157–166. 2

[Nit15] NITENTDO: Wii remote. http://www.nintendo.com/consumer/wiiplay.jsp, accessed 25/01/2015. 1

[NWB*10] NGUYEN N., WHEATLAND N., BROWN D., PARISE B., LIU C. K., ZORDAN V.: Performance capture with physical interaction. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2010), ACM SIGGRAPH/Eurographics, pp. 189–195. 1

[Oka03] OKADA Y.: Real-time motion generation of articulated figures using puppet/marionette metaphor for interactive animation systems. In *International Conference on Visualization, Imaging, and Image Processing* (2003), -, pp. 13–18. 3

[RI99] ROUSE III R.: What's your perspective? *ACM SIGGRAPH Computer Graphics 33*, 3 (1999), 9–12. 2

[RTIK*14] RHODIN H., TOMPKIN J., IN KIM K., VARANASI K., SEIDEL H. P., THEOBALT C.: Interactive motion mapping for realâĂŘtime character control. *Computer Graphics Forum 33*, 2 (2014), 273–282. 1, 2

[SH08a] SHIRATORI T., HODGINS J. K.: Accelerometer-based user interfaces for the control of a physically simulated character. *ACM Transactions on Graphics 27*, 5 (2008), Article No. 123. 2

[SH08b] SLYPER R., HODGINS J. K.: Action capture with accelerometers. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2008), ACM SIGGRAPH/Eurographics, pp. 193–199. 2

[SLSG01] SHIN H. J., LEE J., SHIN S. Y., GLEICHER M.: Computer puppetry: An importance-based approach. *ACM Transactions on Graphics 20*, 2 (2001), 67–94. 1

[Stu98] STURMAN D.: Computer puppetry. *IEEE Computer Graphics and Applications 18*, 1 (1998), 38–45. 1

[SZ93] STURMAN D. J., ZELTZER D.: A design method for "whole-hand" human-computer interaction. *ACM Transactions on Information Systems 11*, 3 (1993), 219–238. 3

[Uni15a] UNITY TECHNOLOGIES: Unity3d game engine. http://unity3d.com/, accessed 25/01/2015. 5

[Uni15b] UNIVERSITY OF SOUTHERN CALIFORNIA, INSTITUTE FOR CREATIVE TECHNOLOGIES: Smartbody. http://smartbody.ict.usc.edu/, accessed 25/01/2015. 5

[WP09] WANG R. Y., POPOVIĆ J.: Real-time hand-tracking with a color glove. *ACM Transactions on Graphics 28*, 3 (2009), Article No. 63. 3