Towards Universal Kinect Interface for Fighting Games

Pujana Paliyawan[a,b], Kingkarn Sookhanaphibarn[a], Worawat Choensawat[a], and Ruck Thawonmas[b]

[a]Multimedia Intelligent Technology Lab, School of Science and Technology, Bangkok University, Thailand
[b]Intelligent Computer Entertainment Lab, Graduate School of Information Science and Engineering, Ritsumeikan University, Shiga, Japan

Email: pujana.p@gmail.com, kingkarn.s@bu.ac.th, worawat.c@bu.ac.th, ruck@is.ritsumei.ac.jp

***Abstract*—This paper presents an approach towards developing a universal Kinect interface supporting game-playing. The proposed interface is designed to be used with existing fighting games of any sort. Players can define their own postures and map them to keystrokes for sending input to a targeted game without the need of accessing game source code. We describe the overview of the interface system; including motion segmentation, posture feature extraction, and posture-to-key mapping.**

Keywords—Interface Design; Kinect; Posture Recognition; Frame Extraction; Key Mapping;

I. INTRODUCTION

Today, game controllers with a natural user interface (NUI) are increasingly grabbing attention from game players. However, most games for playing with NUI are individually developed and limited in number. Some players may also have favorite games they want to play with NUI, but such games are available to be played with keyboard only.

In 2012, Suma et al. [1] presented a middleware named FAAST for mapping gestures into keyboard commands and sending outputs to a targeted application. Our work shares the basic concept to them, but differs in that the posture recognition module based on an AI approach is proposed. This allows a player to add a new gesture to the system by just recording data while performing a movement; there is no need of writing detection rules manually. Another main difference is that we optimize our system to work with fighting games, where the "facing direction" of the player's character can lead to different expected key presses and overlap postures can easily exist.

The contributions of this paper are twofold. The first is that of universalizing the interface so that it is able to work with any fighting games; by provisioning a format of posture-to-key data and its processing algorithm. The second is that of allowing players to define new postures by introducing approaches for frame extraction and posture recognition. The proposed interface system has been tested against typical existing games in the genre of fighting games: "FightingICE[1] [2]", "Kickboxing[2]", "Kung-Fu Master[3]" and "Ultra Street Fighter IV[4]".

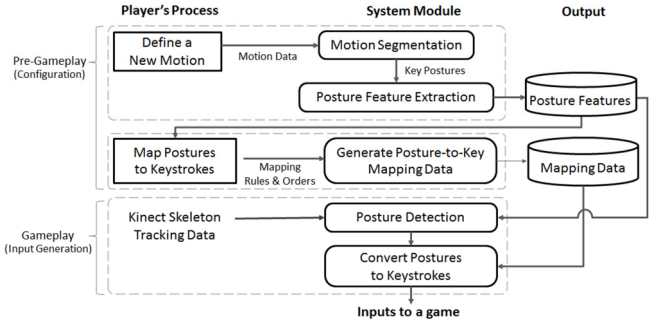

Fig. 1. Overview of the Proposed Interface System.

II. PROPOSED INTERFACE SYSTEM

The overview of our system is shown in Fig. 1. There are two Major modules are as follow;

A. Defining a New Motion

A player can introduce a new posture or a new motion (sequence of postures) to the system by performing a desired movement. The obtained motion data are in the form of skeleton tracking data that represent a sequence of human body joint positions in 3D coordinate space. The system will segment the given motion into key postures by using a technique adapted from Choensawat et al.'s work [3].

The process consists of (1) calculating the velocity of body movement by summing up of the Euclidian distances calculated by relative changes in joint positions in a unit of time, (2) smoothing up the velocity data by using the moving average and then marking up local minima. Each local minima represents a key posture of the defined motion as shown in Fig. 2, and its features are extracted and stored into the database.

For example, the player can add a new motion called "Shakunetsu Hadoken[1]" to the system. This motion consists of three key postures: *SH_Charge*, *SH_Ready* and *SH_Release*. However, after analyzing the given motion data, four postures are extracted by the system as shown in Fig. 2. The first extracted posture is a normal standing posture, not an intended step for executing the Shakunetsu Hadoken; however, the player can tell the system to exclude this posture in the motion detection process.

[1] ICE Lab., "FightingICE," [Java Game], ver.1.22 (May 12, 2015), Available: http://www.ice.ci.ritsumei.ac.jp/~ftgaic/
[2] Miniclip, "Kickboxing," [Web Game], (Access on June 12, 2015), Available: http://www.miniclip.com/games/kickboxing/en/
[3] Zapak, "Kung-Fu Master," [Web Game], (Access on June 12, 2015), Available: http://feed.zapak.com/kung-fu-master-games-play-1183-1
[4] CAPCOM U.S.A., INC., "Ultra Street Fighter IV," [Video Game], PC ver. (Aug 8, 2014), Official Website: http://www.streetfighter.com/us/usfiv

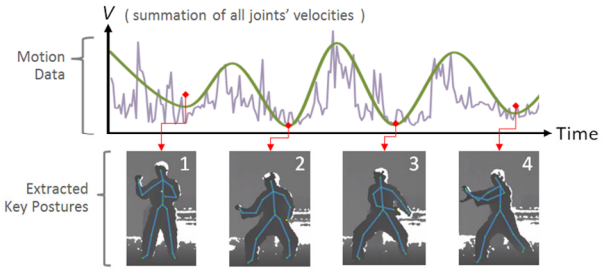

Fig. 2. Posture Extraction for Shakunetsu Hadoken.

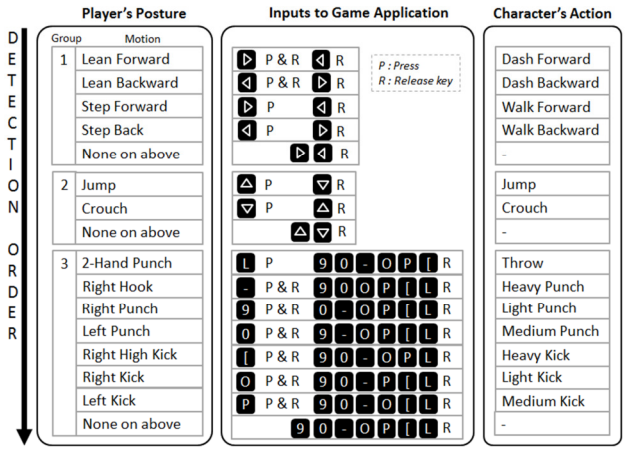

Fig. 3. An example of Posture-to-Key Mapping.

Fig. 4. Key presses for executing Shakunetsu Hadou[1].

B. Mapping Postures to Keystrokes

This process is to map each of the postures available in the database to a desired keystroke output.

1) Key Pressing Commands

The command data stream will be continuously sent to the game whenever Kinect data are captured and analyzed. All commands are classified into three types: *Press*, *Release* and *Press & Release*.

Press is a command to simulate key pressing. When this command is called, the targeted key will be pressed and held until the *Release* key is sent.

Press & Release is a command that sends *Press* and *Release* commands switchingly, resulting in continuously tapping the targeted button until *Release* key is sent.

2) Posture-to-Key Mapping

An example of posture-to-key mapping is shown in Fig. 3. This example is derived from the basic character control in Ultra Street Fighter IV.

The game inputs in this example are categorized into three groups as follows:

G#1 controls the horizontal movements. The detection begins with *Posture#1* (*P#1*), which is "Leaning Forward." When the player leans forward, the system sends a command *Press & Release* on the right arrow key to the game, hence, triggering dashing forward. If leaning forward is detected, the system will skip detection of the rest of the postures in *G#1* and continue to process *G#2 P#1*; otherwise it processes the next posture in the same group (*G#1 P#2*).

G#2 controls the vertical movements. Output keys from different groups can be combined. For example, a combination of "Step Forward" and "Jump" will result in "Jump Forward."

G#3 controls the attacks by game characters. We recommend that the pressing commands for basic attacks, such as "Light Punch", should be sent as *Press & Release;* so while the player is holding the right punching posture, the character will continuously execute "Light Punch". We found this way helps the player keep up with the game speed.

3) Posture-to-Key Mapping for Special Moves

Special moves are the character's moves that require a sequence of button presses for execution. Figure 4 shows an example how Shakunetsu Hadouken is executed in Ultra Street Fighter IV. It is needless to say that it is not practical for the player to execute a sequence of 6-step key presses by using a sequence of 6-step postures; hence, a special group of detection called *G#0* is introduced to solve this issue.

For example, let us consider the mapping of the player's Shakunetsu Hadouken to his or her character's Shakunetsu Hadouken. If the *SH_Release* posture is detected in *G#0*, the system will skip detection of the rest in the group and automatically send the aforementioned sequence of 6 inputs to execute Shakunetsu Hadouken as shown in Fig. 4.

C. Posture Detection during the Gameplay

On the gameplay, the system will analyze postures in each group of posture-to-key mapping data and generate classifiers for posture detection.

III. Results & Conclusions

From the results of testing the proposed interface (cf. the uploaded videos[2]); the system is capable of controlling a fighting game character in an effective manner. In the first-three-mentioned games, only a few configurations are required on keystroke mapping. For Ultra Street Fighter IV, detailed configurations are required for special moves. All of the aforementioned games are standard fighting games, which cover most features in the genre of fighting games.

References

[1] E.A. Suma, D.M. Krum, B. Lange, S. Koenig, A. Rizzo and M. Bolas, "Adapting user interfaces for gestural interaction with the flexible action and articulated skeleton toolkit," Computers & Graphics, 2013, vol. 37, pp. 193-201.

[2] F. Lu, K. Yamamoto, L.H. Nomura, S. Mizuno, Y-M. Lee and R. Thawonmas, "Fighting Game Artificial Intelligence Competition Platform," GCCE 2013, IEEE, Tokyo, Japan, 2013, pp. 320-323.

[3] W. Choensawat, M. Nakamura and K. Hachimura, "GenLaban: A tool for generating Labanotation from motion capture data," Multimedia Tools and Applications, 2014, pp. 1-24.

[1] Street Fighter Wikia, "Shakunetsu Hadoken," Available: http://streetfighter.wikia.com/wiki/Shakunetsu_Hadoken

[2] ICE Lab., "Universal Kinect Interface," Available: https://www.youtube.com/watch?v=PwJkY9vfAU8&list=PLwrbb0kRyPsmhLmi8FN_Dl1KoynMoRE2r