Representation of Intractable Objects and Action Sequences in VR Using Hand Gesture Recognition

Denis Savosin[1], Simant Prakoonwit[1], Feng Tian[1], Jingui Liang[2], Zhigeng Pan[3]

[1] Bournemouth University, Dorset, United Kingdom
[2] Shihezi University, Xinjiang, China
[3] Hangzhou Normal University, Hangzhou, China
{i7620684,sprakoonwit,ftian}@bournemouth.ac.uk

Abstract. We propose a novel approach on using static and dynamic gesture recognition in VR games to represent interactive objects in games, such as equipment system, weapons and handy-tools. We examine various applications of gesture recognition in games, learning, medicine and VR, including how developers currently use the bundles of HDM devices paired with hand tracking sensors. The proposed approach provides game developers with a control over recording gestures and binding them to in-game intractable objects and equipment.

Keywords: Games, Gesture Recognition, Virtual Reality.

1 Introduction

In recent years, many innovative Human-Computer-Interaction (HIC) controllers have emerged into a process of creation the new gameplay experiences in video games [6, 13, 15]. Each of the 7th generation of game consoles – Xbox 360, PlayStation 3 and Wii has introduced their visions of the interaction with games. The most known example of those devices was a Kinect controller, which brought the gesture and human pose recognition into mass games products. Since that times, the gesture recognition field has been extensively studied, particularly the hand gesture recognition and facial recognition [6, 10]. The gesture recognition has been applied to solve various problems in human-computer interaction field, including 'serious games' and rehabilitation applications, handwriting, numeral gesture recognition and Sign Language [12, 14, 16].

The 8th generation of gaming systems has not only brought the overall increase in graphics fidelity, the complexity of a gameplay and AI but also become a first ever generation to adapt a Virtual Reality Head Mounted Displays (HMD) to mass use and introduced the players to a new interactive entertainment experiences. The introduction of a new generation of HMD devices raised a question of adapting input controllers to act as complement (supplement) to VR devices to expose the potential of interaction with games [8, 13].

Several companies, such as Leap Motion Inc. and Oculus have made significant progress in combining the HMD devices and hand-recognition controllers [2, 3, 4]

Although the combination of the VR HMD device and a hand tracking controller allowed game developers start exploring the new opportunities to interact with the games, most of the existing games and prototypes use the simple 'mimic' of the hands in the virtual worlds [1]. Those games are mostly utilizing the hands as grasping and handling tools to interact with some interactive objects, like normally human do in the reality [1] or as a to give gesture commands to a game [13]. From the overview of a portfolio of VR games with gesture recognition we conclude that game developers have not been fully implementing games where the hands of the players should mimic weapons, gear and inventory items.

To address this limitation, we propose a novel approach for game developers, which will allow them to fully the expose the potential of the hand-tracking controllers in virtual-reality games, disregarding the genre of the game. Developers would have an API to create the custom database of hand gestures and bind each of the gesture to an interactive object or action sequence. In a genre, as first-person shooters played in a VR, the usage of a player's own hands as a weapon will act a natural way to interact with a game, completely immersing players into virtual reality. In Section II of the paper, we go through the explanation of the approach to implement this tool, including the justification of existing methodologies in this area. In Section III we present evaluation of the effectiveness of various gesture recognitions during a gameplay, followed by the conclusion in Section V.

2 Methodology

We use a Leap Motion controller to capture the image of the hand, and extract the important data, such as fingertips positions, palm normal, and their directions and then passing that data to a Support Vector Machine classification learner [7] to train the system which must respond to a stream of data in real time and give the correct gesture recognition.

To train the recognition system, game developers must record all proposed gestures and build the database of the gestures. In our recognition system, we have proposed the "trigger-class" association approach. Game developers will have an option to create their class [7] and nominate it, for example: "shotgun". Then they can select that class response as active and start recording gestures. All data, extracted from a Leap controller will be marked accordingly to that class and feed into database table. Developers have an option of recording one gesture at a time, or record all gestures in one run, by switching active classes using keyboard or using timers.

Once the database is filled with data and class responders, the data are fed into a Multi-SVM [7] classifier to recognize the performed gestures. After the classifier has been trained, the database can be disposed optionally.

Developers then can assign each of the static gesture [5] variable to an item in a game and the dynamic gesture [9] variables to trigger action sequences, such as firing a gun or throwing an object. Here we introduce a concept of "gesture blending" – the smooth transition from a static object to an action sequence with that object. The example is a "stone" object represented by player holding his hand in a fist and the sequence action "throw stone" where the player repeats a throwing move with his hand. This problem is discussed in the testing section.

3 Experiment setup

To do tests and evaluate the viability of the proposed concept, the experimental setup has been implemented. The setup consists of the C++ console application, written using SDL2 library and Leap Motion SDK. The application is written and built using XCode 8 IDE under Mac OS X 10.12 operational system. The program kicks-out with adding Sample Listener – defines the list of call-back functions, who respond to events from a Controller instance (the interface for physical Leap controller). As the purpose of the program is to read data continuously from controller and write it to a file, alongside with response for keypresses, it implements basic events like controller connection and disconnection and the frame events – each frame is used to get a data from it.

The program reads the data outputs from a Leap Motion controller: the fingers positions, velocity and validity, alongside with a palm normal and velocity vectors. Values are written sequentially into a Comma Separated Values (.CVS) table.

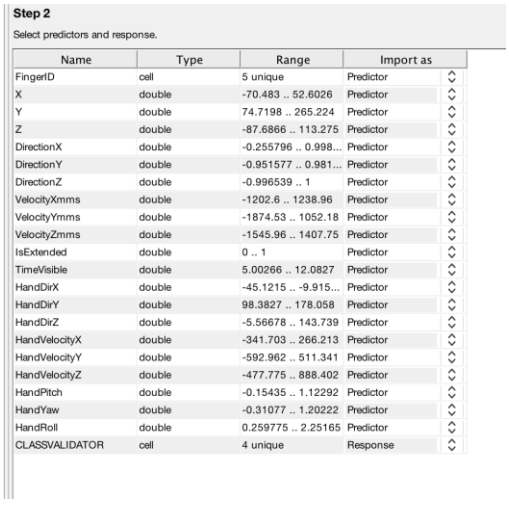

Fig. 1. Selecting the features from recorded database to be imported into MATLAB for testing. 4 unique CLASSVALIDATOR values are: 'unrecognized', 'pistol', 'fist', 'shooting action'

	A	B	C	D	E	F	G	S	T	U	V
1	Finger ID	X	Y	Z	DirectionX	DirectionY	DirectionZ	HandPitch	HandYaw	HandRoll	CLASS(VALIDATOR)
2	Thumb	-32.3269	220.184	37.7749	-0.005464	0.288962	-0.957325	0.18102	0.218757	-0.776146	unrecognised
3	Index	-30.416	202.135	-13.336	-0.319262	-0.001198	-0.947666	0.18102	0.218757	-0.776146	unrecognised
4	Middle	-42.6243	143.922	42.4189	-0.42981	-0.333249	0.839171	0.18102	0.218757	-0.776146	unrecognised
5	Ring	-40.7802	130.212	36.164	-0.842755	-0.528025	-0.104658	0.18102	0.218757	-0.776146	unrecognised
6	Pinky	-20.3064	130.04	39.7539	-0.890779	-0.387827	-0.236859	0.18102	0.218757	-0.776146	unrecognised
7	Index	-30.2807	202.095	-14.0295	-0.322641	0.002268	-0.946519	0.18102	0.218757	-0.776146	unrecognised
8	Thumb	-29.5654	219.252	37.4543	0.02227	0.262272	-0.964737	0.170692	0.214012	-0.791823	unrecognised
9	Middle	-42.3845	144.259	43.5251	-0.41175	-0.30869	0.857422	0.170692	0.214012	-0.791823	unrecognised
10	Ring	-41.0053	130.111	36.4187	-0.851775	-0.515393	-0.09407	0.170692	0.214012	-0.791823	unrecognised
11	Pinky	-20.5628	129.86	40.0201	-0.898501	-0.375358	-0.227602	0.170692	0.214012	-0.791823	unrecognised
12	Index	-30.1583	202.041	-14.6522	-0.325342	0.006577	-0.945573	0.170692	0.214012	-0.791823	unrecognised
13	Thumb	-27.7294	218.642	37.2126	0.036508	0.245267	-0.968768	0.162209	0.209807	-0.806229	unrecognised
14	Middle	-42.1514	144.583	44.5558	-0.396618	-0.288652	0.871421	0.162209	0.209807	-0.806229	unrecognised
15	Ring	-41.335	130.013	36.6707	-0.859883	-0.503498	-0.08421	0.162209	0.209807	-0.806229	unrecognised
16	Pinky	-20.9309	129.64	40.2562	-0.90545	-0.363473	-0.219196	0.162209	0.209807	-0.806229	unrecognised
17	Index	-30.0368	201.983	-15.2345	-0.326164	0.011228	-0.945247	0.162209	0.209807	-0.806229	unrecognised
18	Thumb	-26.5462	218.206	36.9911	0.044112	0.23443	-0.971132	0.162209	0.209807	-0.806229	unrecognised

Fig. 2. Screenshots demonstrating the resulting .CSV table with data captured from a Leap Controller.

The last field in each table entry is an actual class variable – we use 4 class variables in our experimental setup: ‘fist’, ‘pistol’ – static gestures, ‘pistol shot’ – the dynamic gesture to define the shooting action sequence. The ‘unrecognized’ class is used to mark all fields, which are not intended to be classified as a gesture. By pressing corresponding keys during recording, those class variables have been recorded to a field. One data set was recorded using those class variables, and the second set was recorded with the same gestures, but all ‘class’ fields were left blank – this was due to simulation of the real case scenario, where the data from a controller is going to be fed into trained model in the real time.

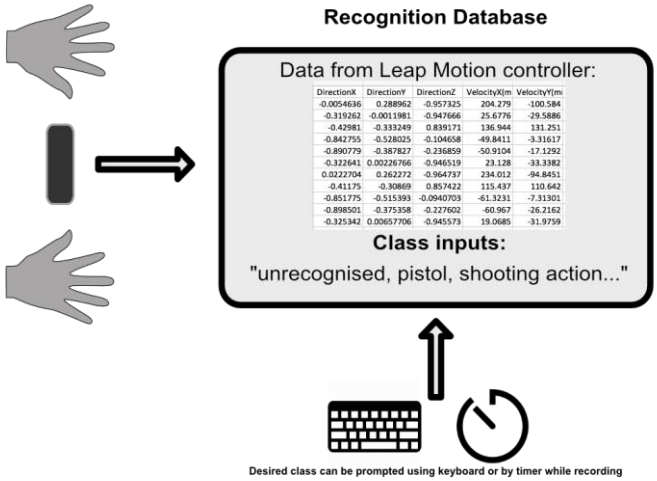

Fig. 3. Image shows the gesture recording pipeline, which is used by game developers to create and record gestures for their VR games.

4 Testing and evaluation

Resulting training set with recorded class variables has been uploaded to a MATLAB R2016b and used to train Multiple-SVMs classifiers. The first obtained performance and accuracy results allowed to make changes into process of the variable selection, as some variables have more effect on a prediction performance than others.

Type of SVM model	True Positive	True Negative	Observations	Misc. Rate
Medium Gaussian SVM	98%	2%	False: (33 + 3) True: (165+180) Total: 381	(33+3)/381 = 0.09
Fine Gaussian SVM	99%	1%	False: (12 + 2) True: (166+201) Total: 381	(12+2)/381 = 0.03
Cubic SVM	97%	3%	False: (15 + 5) True: (163+198) Total: 381	(15+5)/381 = 0.05
Linear SVM	72%	28%	False: (45 + 47) True: (121+168) Total: 381	(45+47)/381 = 0.24

Table 1. The table above shows performance and misclassification rates of Multiple-SVMs trained to recognize 'pistol' class.

After the classification algorithm has been trained, the second testing set with a class variables left blank has been used to evaluate resulting prediction model. The results of the prediction were satisfying enough, and we have obtained the correct predictions on all four trained classes, concluding, that a chosen concept is viable, and after more tweaking can be implemented as a full game engine plug-in.

Although the results of the experiments in a MATLAB were satisfying to continue exploring the concept in rather more sophisticated manner, further progress on the concept should be ideally supervised by the gameplay programmers, as problems discovered, such as solving the transition between static 'item' gestures and dynamic 'action' gestures in a runtime. Also, some genres of games are more dependent on the timings and a prediction speed rather than accuracy. The problem can be addressed on both sides: gameplay programmers and designers should adopt the gameplay logic, if an implementation, in other hand will be flexible to fit into games of various genres.

5 Future Work

As we are planning to continue working on the prototype and an approach, implementing the Unity and the Unreal Engine plug-in, which can be embedded into the editor. We prepared mock-ups to demonstrate concepts of how the final product might look like. Figures show two plug-in windows on OS X system, where the first figure shows the ‘Gesture Learner’ window – setup and recording of gesture database. The second figure shows ‘Linker’ window – binding recorded gestures to an equipment or gearing system in game.

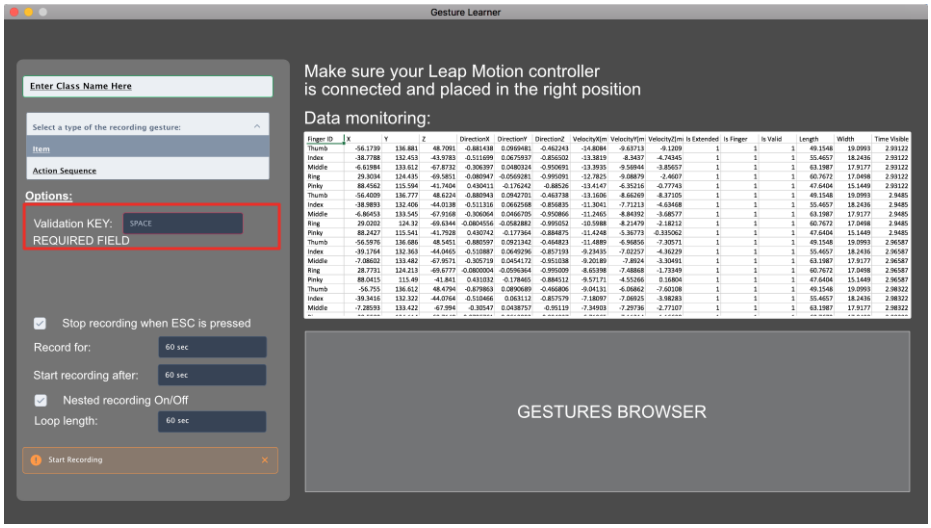

Fig. 4. Image shows a concept of the ‘Gesture Learner’ part and an editor window of the final product.

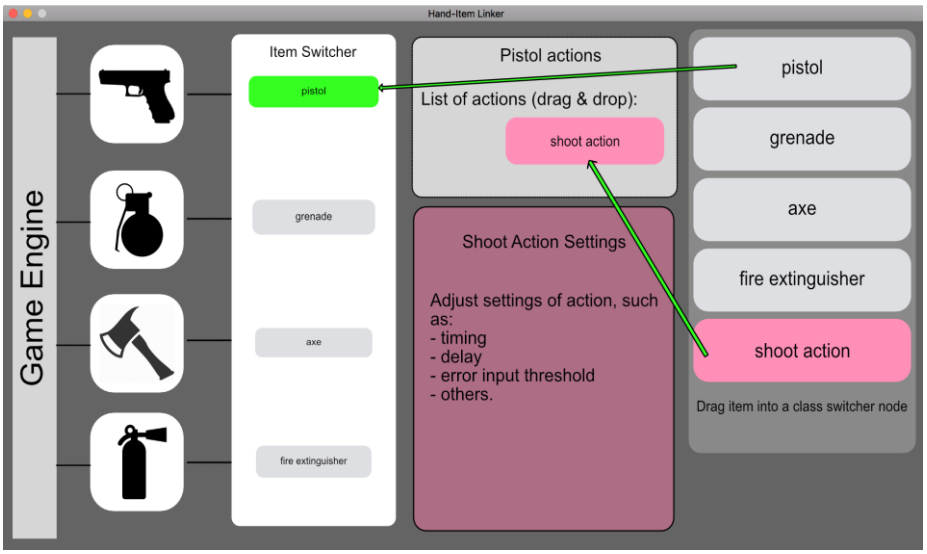

Fig. 5. Image shows a concept of the 'Linker' part and an editor window of the final product.

6 Conclusion

In this paper, we have proposed an approach for recording and predicting static gestures and actions, aiming to give game developers an opportunity to deeper explore the possible use cases for VR headsets bundled with tracking devices. Developers might also be able to adapt the gameplay experiences to immerse players into virtual reality. By creating this prototype, we will investigate more about the opportunities for gesture recognition in Virtual Reality games. We also plan to further improve the design of the existing prototype and embed the tool into game engines, such as Unreal Engine and Unity as a plug-in. The concept presented here, and our discussion on how this concept can be adapted for VR games, certainly can further promote the research in this field and grab game developers' attention to this method, which can be used to create immersive VR experiences.

7 Acknowledgement

This paper is partially supported by the project – Virtual Visualization System for Culture Communications (2015BAK04B05) funded under the theme – National Science and Technology Supporting Project, China.

References

1. (2017) VR – Leap Motion Gallery. In: Gallery.leapmotion.com. https://gallery.leapmotion.com/category/vr/. Accessed 6 Apr 2017
2. (2017) VR Setup. In: Leap Motion Developer. https://developer.leapmotion.com/vr-setup/. Accessed 11 Apr 2017
3. (2017) Unreal. In: Leap Motion Developer. https://developer.leapmotion.com/unreal#103. Accessed 11 Apr 2017
4. (2017) Unity. In: Leap Motion Developer. https://developer.leapmotion.com/unity. Accessed 11 Apr 2017
5. Chen Y, Ding Z, Chen Y, Wu X (2015) Rapid recognition of dynamic hand gestures using leap motion. 2015 IEEE International Conference on Information and Automation. doi: 10.1109/icinfa.2015.7279509
6. Cheng H, Yang L, Liu Z (2016) Survey on 3D Hand Gesture Recognition. IEEE Transactions on Circuits and Systems for Video Technology 26:1659-1673. doi: 10.1109/tcsvt.2015.2469551
7. Cristianini N, Shawe Taylor J (2000) An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, Cambridge
8. Ling H, Rui L (2016) VR glasses and leap motion trends in education. 2016 11th International Conference on Computer Science & Education (ICCSE). doi: 10.1109/iccse.2016.7581705
9. Lu W, Tong Z, Chu J (2016) Dynamic Hand Gesture Recognition With Leap Motion Controller. IEEE Signal Processing Letters 23:1188-1192. doi: 10.1109/lsp.2016.2590470
10. Marin G, Dominio F, Zanuttigh P (2014) Hand gesture recognition with leap motion and kinect devices. 2014 IEEE International Conference on Image Processing (ICIP). doi: 10.1109/icip.2014.7025313
11. Mentzelopoulos M, Tarpini F, Emanuele A, Protopsaltis A (2015) Hardware Interfaces for VR Applications: Evaluation on Prototypes. 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing. doi: 10.1109/cit/iucc/dasc/picom.2015.237
12. Naglot D, Kulkarni M (2016) Real time sign language recognition using the leap motion controller. 2016 International Conference on Inventive Computation Technologies (ICICT). doi: 10.1109/inventive.2016.7830097
13. Rautaray S, Agrawal A (2011) Interaction with virtual game through hand gesture recognition. 2011 International Conference on Multimedia, Signal Processing and Communication Technologies. doi: 10.1109/mspct.2011.6150485
14. Sharma J, Gupta R, Pathak V (2015) Numeral Gesture Recognition Using Leap Motion Sensor. 2015 International Conference on Computational Intelligence and Communication Networks (CICN). doi: 10.1109/cicn.2015.86
15. Sonkusare J, Chopade N, Sor R, Tade S (2015) A Review on Hand Gesture Recognition System. 2015 International Conference on Computing Communication Control and Automation. doi: 10.1109/iccubea.2015.158
16. Spanogianopoulos S, Sirlantzis K, Mentzelopoulos M, Protopsaltis A (2014) Human computer interaction using gestures for mobile devices and serious games: A review. 2014 International Conference on Interactive Mobile Communication Technologies and Learning (IMCL2014). doi: 10.1109/imctl.2014.7011154