Gesture Recognition Using Leap Motion: A Machine Learning-based Controller Interface

Ivo A. Stinghen Filho, Bernardo B. Gatto, José Luiz de S. Pio,
Federal University of Amazonas
Manaus, Amazonas
{*iasf,bernardo,josepio*}*@icomp.ufam.edu.br*

Estevam N. Chen, Jucimar M. Junior, Ricardo Barboza
Amazonas State University
Manaus, Amazonas
enc.eng@uea.edu.br, {*jucimar.jr,rsbarboza*}*@gmail.com*

***Abstract*—There is a growing tendency of making use of real-time interactions and converting the gestures to the virtual game scenario. In this paper, we present a gesture interfacing controller for real-time communication between the Leap Motion sensor and games. We also we compare the effectiveness of various methods of real-time machine learning algorithms to find the most optimal way to identify static hand gestures, as well as the most optimal sample size for use during the training step Moreover, we introduce a novel static hand gesture dataset containing** 1200 **samples for** 10 **static gesture classes. This dataset may encourage to develop innovative gesture recognition methods.**

***Keywords*-Virtual Reality, Leap Motion, Motion Capture, Machine Learning**

I. Introduction

Humans interact with machines in a variety of ways. As such, many forms of HCI (Human-Computer Interaction) has been developed [6], [7]. Although the use of the mouse and keyboard is widespread, new methods of HCI are also developed. An example is gesture recognition, a topic that received significant attention in the field of HCI due to the development of Virtual Reality (VR) technology, as well as a method to control robots [17].

A gesture is a form of nonverbal expression. It includes hands, face and other parts of the body [4]. Hand gestures can be divided into two categories: static and dynamic gestures [3]. In static gestures, the gestures usually do not change its shape over time. This paper will focus on such hand gestures.

Depending on the type of the input data, the hand gesture recognition can also be divided into two categories: appearance-based and 3D model-based algorithms. Appearance-based algorithms use the data acquired from the silhouette or contour of the input images. Meanwhile 3D model-based algorithms use volumetric or skeletal data, or even a combination of the two [3].

In this work, we present a comparison of three different machine learning algorithms used in gesture recognition literature, with a game interface control technology providing improved gameplay for games that use gesture recognition. Moreover, we provide a dataset containing 1200 samples for 10 static gesture classes (https://drive.google.com/open?id=1lXKnAlNdJ0I1tbPNnvXbi0VOCyUEPaMF).

II. Literature Review

Leap Motion [2] is a computer hardware sensor device that supports hand and finger motions as input, similar to a keyboard or a mouse, requiring no hand contact or touching [12].

In this work, we used a variety of classifiers, including KNN (K-Nearest Neighbors algorithm), Decision Trees [16] and SVM (Support Vector Machines) [9].

A. Review on Gesture Recognition Methods

In the work of Ameur et al. [1], the authors introduced an application with gestural hand control using leap motion for medical visualization. Their experimental results demonstrated a high accuracy rate of about 81%. In the paper of Mapari and Kharat [11], a novel method for recognition of American Sign Language (ASL) is proposed using Leap Motion. The proposed feature scheme, combined with the MLP achieved 90% of accuracy.

In the work of D. Yao et al. [16], the authors proposed a decision-tree-based algorithm to recognize 3D gestures. The provided experimental results show that the recognition rate reached 95.8%, with a response time of 5.4 seconds. Similarly, In the paper of C.-H. Chuan et al. [4] the authors present an American Sign Language recognition system with Leap Motion, employing KNN and SVM, with a average classification rate of 72.78% and 79.83%, respectively.

III. Methodology

In order to train the machine learning algorithms, it is necessary to acquire the data regarding the fingers. To do so, we recorded a series of hand signs using Leap Motion [12], and the Leap Motion API (Application Programming Interface) in Unity was used [13]. With this, it was possible to get many different combinations of variables regarding the fingers. In this work the main variables chosen were the combination of the normalized spatial positions of the tip of the 5 fingers and the 4 angles between adjacent fingers [3], as represented in the figure 1.

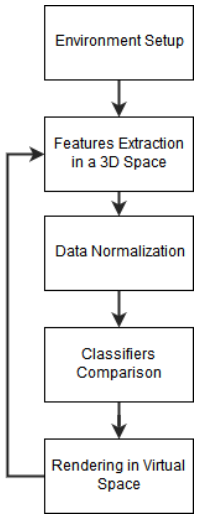

Figure 1. Methodology flowchart

A. *Features Extraction in a 3D Space*

The data is acquired through the Leap Motion camera, which has infrared sensors. The camera is to be tied to the user's forehead, as shown in figure 2.

Figure 2. Leap Motion usage

B. *Data Normalization*

Data normalization is a critical step in computer vision related-works [8]. The main variables chosen were the combination of the normalized spatial positions of the 5 fingertips, each one containing a 3D position of each detected finger.

$$\Gamma_i = (P - \rho_i) \tag{1}$$

Γ is the normalized position based on the center of the hand palm (P), and ρ is the tip position of a finger.

Combined to this data, we also collect the 4 angles between adjacent fingers, each one calculated according to law of cosines.

$$\theta_i = \frac{|\rho_{i+1}, P|^2 + |\rho_i, P|^2 - |\rho_i, \rho_{i+1}|}{2 * |\rho_{i+1}, P|^2 * |\rho_i, P|^2} \tag{2}$$

We made use of three world points to calculate the angle between fingers, θ. ρ is the tip position of a finger, and P is the center of the hand palm and "i" varies from 1 to n - 1, where n is the number of fingers. And

$$|\rho_A, \rho_B| \tag{3}$$

calculate the distance between two points.

C. *Classification Models*

In this work, we evaluate the use of different types of classifiers to use in our gesture recognition control system. We used three classifiers: K-Nearest Neighbors(KNN), Support Vector Machines(SVM) and Decision Trees. To do so, a python library named numPy was used, similar to its use in Solem's work [14].

IV. Data set

In this database 6 different volunteers use their hands to feed the database. Each one of them put the leap motion on their heads and training the gestures while having variations of said gestures.

In this work ten classes were used, each representing a hand gesture: "OPEN", "CLOSE", "THUMB", "TWO", "THREE", "FOUR", "LOVE", "COOL", "FIRE" and "SHAKA". The hand gestures can be seen in figure 3.

Figure 3. 10 classes used in this work.

V. Experimental Results

To generate the necessary data, it is required to do at least one record session for each machine learning class, each representing a hand sign. For each record session, while positioning the hand, a script in Unity records the data to a .csv file, along with the class name. A line is recorded every 0.05s, varying in total recording time with necessity. 30% of the lines were used to train, while the remaining 70% lines were used to evaluate accuracy. The algorithm then returns the percentage of correct predictions.

The .csv files are then analyzed. Overfitting started occurring when over than 12000 samples were utilized (1200

per hand gesture), as such, this particular sample size was used. Figure 6 shows the accuracy of gesture predictions using different classifier algorithms utilizing a file with 1200 learning samples per gesture. It shows a comparison based on the best algorithms results. The worst results were that of the "COOL" class, with $96,57\%$ hit rate. Table 1 shows the confusion matrix of the hand gesture recognition of the selected classes, In order to check for false positives and consider the complexity of each gesture of the Decision tree algorithm. The values are presented in percentage based on the number of gestures of each class. Noticeably, the "SHAKA" class has the smaller false positive rate.

A. Application

Given the results, the decision tree classifier was used for offline [5] base classification and training with real-time sample input. By having offline training, it's possible to make real-time predictions.

Our system have the following specifications: Windows 10 version 10.0.17, 64 bit, Intel® Core™ i5-5200U CPU @ 2.20GHz, 2201 Mhz, 2 Core(s), 8GB RAM DDR3 3x, USB 3.0 port, NVIDIA GTX 920M. We also make use of Leap Motion, which uses two monochromatic IR (infrared) cameras and three infrared LEDs (Light Emitting Diode), the sensor device observes a roughly hemispherical area, approximately up to 1 meter of distance. The LEDs are also able to generate pattern-less IR light [15]. The software tools used were Python 3.6.2 and Unity Engine 2018.1.0f2.

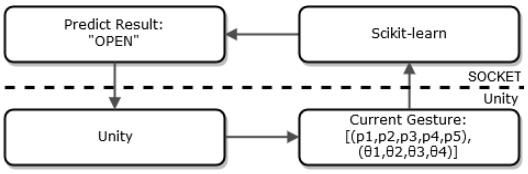

Figure 4. Data loop between Python and Unity.

A loop was created between Unity and a socket created by Python, running on localhost, as shown in figure 4. The current state is updated every 0.02 seconds in Unity's interface, Unity then sends a vector as input through the socket to the decision tree classifier in scikit-learn.

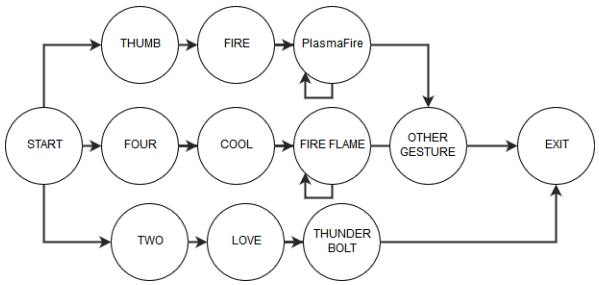

Figure 5. The sequences used in the tests.

When receiving the data through the socket, the predict() function is run, (using the decision tree classifiers) and returns the result. Ex: "Open". Finally, Unity receives the result from the socket and creates effects as needed.

B. Qualitative Analysis

In our analysis the algorithms were labeled as A and B so that the user could not know the name of the classifier he was testing or which had better results in our comparison. Algorithm A represents SVM and B the Decision Tree.

For this, we used the Case Study method [10] to obtain a detailed examination of this comparison in relation to the ease of gesture recognition. In order to obtain significant analysis, the case study had 12 volunteers involved, who tested 6 of the total of 10 gestures of that work.

The volunteers were instructed on how Leap Motion works, including to only use the right hand and to always remain in the field of view of the device, as well as images of each gestures, as shown in figure 3.

Then it was instructed to follow the sequence of gestures shown in figure 5. After that, an Unity scene is initialized and the user is instructed to attempt to destroy all targets on the screen. A test ends when destroying all targets or when the sequence is finished. In the analysis to the test using the A algorithm, the following results were verified:

In the gesture sequence 1 (tested with all users), the system often outputs the "Thumb" gesture when the inputs were "Fire". Everyone managed to perform the "Plasma Fire" magic. In the gesture sequence 2 (tested with 9 users), the system often outputs the "Open" gesture when the inputs were "Four". Everyone managed to perform the "Fire Flame" magic. In the gesture sequence 3 (tested with all users), the system often outputs the "Fire" gesture when the inputs were "Two", meaning summoning the Thunderbolt magic was impossible in all cases. That was due to the fact the thumb finger blocked the camera view of the index finger.

Meanwhile using the B algorithm, in all 3 gesture sequences (tested with all users) the system output was mostly correct gestures, with a small number of false positives, although the same problem with the "Fire" gesture remained.

Overall, through observation and verbal reports of each user, we came to the conclusion that the Decision Tree approach was superior when comparing with the SVM.

VI. Conclusion and Future Work

In this paper, we used 1200 samples per class and normalized data regarding the fingers and the angles between them. With this, we achieved a hit rate of over 99.7% using the decision tree classifier, while showing that the Unity engine and the Leap Motion API can be used in real-time applications of arbitrary gestures.

Everything considered, further study of how dynamic gestures behave through normalization might be interesting, as both dynamic and static gestures can be used for various applications, such as VR games.

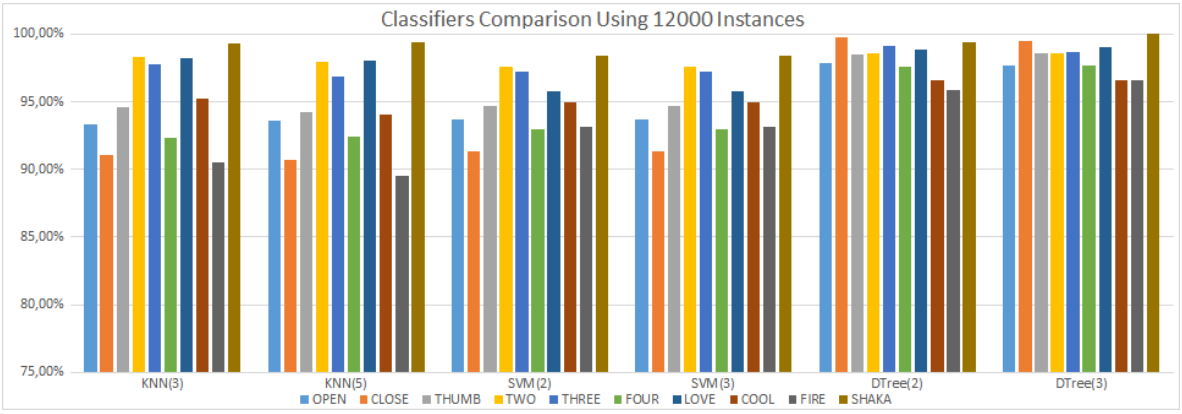

Figure 6. Comparison between KNN (with 3 and 5 neighbors), SVM and Decision Tree classifiers, respectively, for each of the 10 classes.

Table 1: Cross-validated confusion matrix for Decision Tree classifier.

	OPEN	CLOSE	THUMB	TWO	THREE	FOUR	LOVE	COOL	FIRE	SHAKA
OPEN	97.69	0.48				1.05		0.11	0.57	
CLOSE	0.12	99.52							0.34	
THUMB			98.56				0.47		0.8	0.12
TWO	0.24			98.57	0.49	0.47			0.23	
THREE				1.07	98.65	0.12		0.11		
FOUR	1.58				0.12	97.66		0.69		
LOVE	0.24					0.35	99.06	0.23		0.12
COOL	0.85	1.43				0.35	0.12	96.57	0.8	
FIRE	0.85	0.59	0.36	0.48			0.94	0.34	96.58	
SHAKA										100

Acknowledgment

The authors would like to thank IComp UFAM, UEA Ludus Lab, and FAPEAM for supporting the development of this work.

References

[1] S. Ameur, A. B. Khalifa, and M. S. Bouhlel. A comprehensive leap motion database for hand gesture recognition. In *Sciences of Electronics, Technologies of Information and Telecommunications (SETIT), 2016 7th International Conference on*, pages 514–519. IEEE, 2016.

[2] M. Buckwald. Leap motion. https://www.leapmotion.com (accessed: 2018.08.04).

[3] F. Chen, J. Deng, Z. Pang, M. Baghaei Nejad, H. Yang, and G. Yang. Finger angle-based hand gesture recognition for smart infrastructure using wearable wrist-worn camera. *Applied Sciences*, 8(3):369, 2018.

[4] C.-H. Chuan, E. Regina, and C. Guardino. American sign language recognition using leap motion sensor. In *Machine Learning and Applications (ICMLA), 2014 13th International Conference on*, pages 541–544. IEEE, 2014.

[5] E. M. Y. David, Shai; Kushilevitz. Online learning versus offline learning. 1997.

[6] B. B. Gatto, A. Bogdanova, L. S. Souza, and E. M. dos Santos. Hankel subspace method for efficient gesture representation. In *Machine Learning for Signal Processing (MLSP), 2017 IEEE 27th International Workshop on*, pages 1–6. IEEE, 2017.

[7] B. B. Gatto, E. M. dos Santos, and W. S. Da Silva. Orthogonal hankel subspaces for applications in gesture recognition. In *Graphics, Patterns and Images (SIBGRAPI), 2017 30th SIBGRAPI Conference on*, pages 429–435. IEEE, 2017.

[8] B. B. Gatto, S. Waldir, M. Eulanda, and D. Santos. Kernel two dimensional subspace for image set classification. In *Tools with Artificial Intelligence (ICTAI), 2016 IEEE 28th International Conference on*, pages 1004–1011. IEEE, 2016.

[9] S. R. Gunn et al. Support vector machines for classification and regression. *ISIS technical report*, 14(1):5–16, 1998.

[10] S. Krug. Não me faça pensar. *Tradução de Roger Maioli dos Santos, São Paulo: Market Books*, pages 123–137, 2001.

[11] R. B. Mapari and G. Kharat. American static signs recognition using leap motion sensor. In *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies*, page 67. ACM, 2016.

[12] R. Ribeiro et al. *Framework for registration and recognition of free-hand gestures in digital games.* SBGames, 2016.

[13] L. Shao. *Hand movement and gesture recognition using Leap Motion Controller*. Stanford University, Stanford, CA, 2016.

[14] J. E. Solem. Programming computer vision with python. 2012.

[15] D. R. B. F. D. Weichert, F; Bachmann. Analysis of the accuracy and robustness of the leap motion controller. 2013.

[16] D. Yao, M. Jiang, A. Abulizi, and X. You. Decision-tree-based algorithm for 3d sign classification. In *Signal Processing (ICSP), 2014 12th International Conference on*, pages 1200–1204. IEEE, 2014.

[17] J. Youngkyoon, S.-T. Noh, H. J. Chang, and T.-K. Kim. 3d finger cape: Clicking action and position estimation under self-occlusions in egocentric viewpoint. 2015.