Real-Time Gesture Prediction Using Mobile Sensor Data for VR Applications

Vipula Dissanayake, Sachini Herath, Sanka Rasnayaka, Sachith Seneviratne, Rajith Vidanaarachchi and Chandana Gamage

***Abstract*—The objective of this work is to develop a real-time gesture prediction system for navigation in a Virtual Reality Environment. Though earlier work presents situations where the level of activity is high, this research work refers to slight gestures, where the distinction is low. The paper further discusses the use of several machine learning methods to solve this prediction problem, including Support Vector Machines, Random Forests, and Artificial Neural Networks. After considering performance variation with respect to different test configurations, a conclusion is drawn on which configuration is to be used in the prediction engine. The final system was tested on a significant user base with results indicating sufficient accuracy and performance for real-time use.**

***Index Terms*—Gesture input, virtual reality, supervised learning.**

I. Introduction

Virtual Reality is a rapidly growing field with innovative empirical applications. The latest trend in virtual reality applications is to make the virtual reality (VR) experience more realistic by providing new ways to interact with the virtual environment. VR headsets with mounted mobile devices is one of the popular and affordable methods for experiencing Virtual Reality.

However the commonly used means of input prompts for navigation, such as button clicks and touch sense, hinders the immersive VR experience. The objective of this research work is to investigate and develop a unique gesture based input system to navigate seamlessly in a virtual environment while maintaining the immersiveness. Two main gesture input requirements for navigating in a virtual reality environment, have been identified as follows:

- VR gestures: for changing the view angle around a single viewpoint (i.e. looking around the virtual environment)
- Navigation gestures: for changing the viewpoint (i.e. moving from one position to another)

Formally, this research work is focused on classifying a sequence of 3D points captured by a mobile sensor, into two separate classes - Virtual Reality gestures or Navigation gestures. These 3D data points contain various sensory information which are described in detail in the coming sections.

Manuscript received December 4, 2015; revised June 5, 2016.

V. Dissanayake, S. Herath, S. Rasnayaka, S. Seneviratne, R. Vidanaarachchi, and C. Gamage are with the Department of Computer Science & Engineering, University of Moratuwa, Moratuwa, SriLanka. (e-mail: vipula.11@cse.mrt.ac.lk, sachini.11@cse.mrt.ac.lk, sanka.11@cse.mrt.ac.lk, sachith.11@cse.mrt.ac.lk, raji.11@cse.mrt.ac.lk, chandag@cse.mrt.ac.lk).

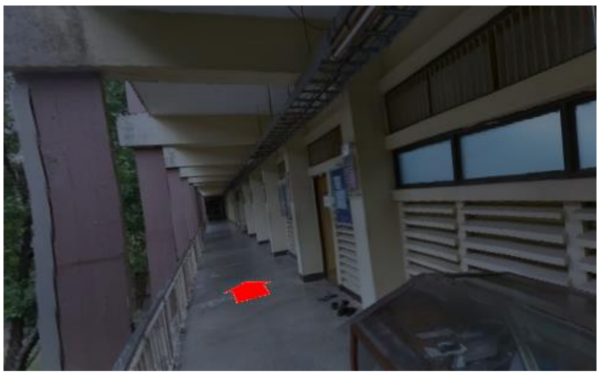
Fig. 1. Directional cues as seen by user.

In application context, when the user in a virtual environment encounters a visual cue suggesting possible movement (as shown in Fig. 1), they will be able to use the navigation gesture to initiate movement. In addition, the user will always be able to use the VR gesture to look around the virtual environment from a stationary position.

II. Related Work

Gesture Recognition forms a key part of the intuitive navigation framework suggested by our research. To obtain foresight into solving this problem, previous research work suggesting various solutions in varying settings were reviewed [1]-[4].

Kim *et al.* have used a dynamic time warping technique with gyroscope data for the problem of 3D handwriting recognition [5]. In this work, one major difference is the decision to avoid accelerometer data due to the gestures showing significant variation between different users

In the research by Ravi *et al.*, a tri-axial accelerometer is used to measure acceleration along the x, y and z axes from which velocity and displacement is also estimated [4]. Using this data, the authors propose a method of classifying users into several categories based on the activities they are performing.

Kwapisz *et al.* proposes a framework for classifying user activity into six separate classes using accelerometer data [1]. In this research, the primary difference from most prior work is the use of a commercial mass-marketed device rather than a research-only device.

Most of the previous works are concentrated on recognizing activities like walking, jogging, climbing up stairs, climbing down stairs, sitting and standing.

Our work extends concepts from feature extraction processes of previous work described above. In addition, the relatively high accuracy obtained by the methods indicate that, it would be sufficiently precise for use in a real world system.

III. Methodology

The mobile device used for VR content streaming comes with an in-built gyroscope and accelerometer. Thus the aim of this project is to identify the movements of the device as Navigation or VR gestures, using the sensor data.

The two types of gestures defined are shown in Fig. 2 and Fig. 3. The gestures shown in Fig. 2 are for the VR motions, which explore the 360°×360° view from a single viewpoint. The gesture shown in Fig. 3 is for navigation, which is moving to the next available viewpoint in the direction faced by the user. This gesture is primarily vertical.

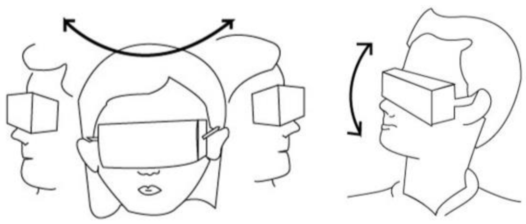
Fig. 2. VR motions.

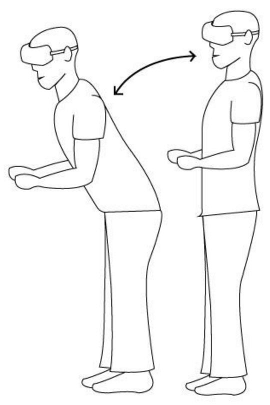
Fig. 3. Navigation motion.

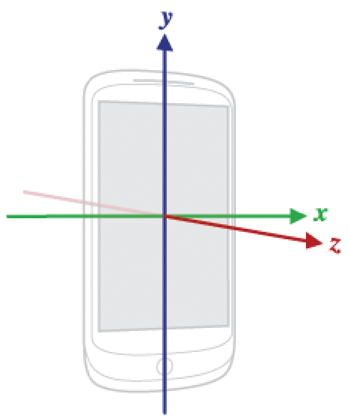

Fig. 1. Axial convention in android [6].

The task of distinguishing the gestures was modeled as a classification problem, using a sliding window over time to extract features. Many sliding window widths and learning methods were tested to find the best suited model for the scenario. In order to compare the machine learning models, it was required to identify a suitable set of evaluation parameters. Among the customary parameters such as accuracy, recall, latency, model size etc. we opted to use accuracy and latency as they were the most contributing factors for the successful completion of the required task.

A. Data Collection

In order to collect data for this task, a data collection Android application was developed. In the application, user is asked to specify the gesture he is about to do, and then perform the gesture, which will be recorded along with the sensor readings.

The motion sensors of the mobile device will generate events when a change is detected. These events were aggregated within a sampling frequency and persisted accordingly. A sampling frequency of 0.2 seconds was used considering that the average time for a quick gesture is over one second.

The motion sensors used for capturing data were the accelerometer and the gyro-sensor. The Android convention uses the axial system shown in Fig. 4 as directions for the sensor data.

Nine attributes, which are discussed in detail below, were captured for deviations and aggregated over an interval of 0.2 seconds. The aggregation was performed to ensure that the data points are evenly distributed along the time axis and to dampen the effect of noise and anomalies. The values obtained are accelerometer reading on x, y and z axis, gyro sensor readings around x, y, and z axis, and the computed attribute values for linear acceleration excluding gravity in x, y, and z directions.

The collected data were saved in the csv format along with additional information such as a unique identification for the user, height etc.

B. Data Pre-processing

Given the nature of data collected (accelerometer and gyro-sensor readings), the data pre-processing needs were minimal.

First and last few seconds' data were removed as a cleaning mechanism, as this usually corresponded to the user wearing and removing the virtual reality headset. Although both raw accelerometer readings and linear acceleration data (gravitational forces excluded) were collected, since the orientation of device changes as the user moves, correcting raw accelerometer data for gravity was error prone. Linear acceleration, on the other hand, is a virtual sensor and implementation may differ from device to device. However, we opted to drop accelerometer data and directly use the readings for linear acceleration without any gravitational correction.

C. Feature Engineering

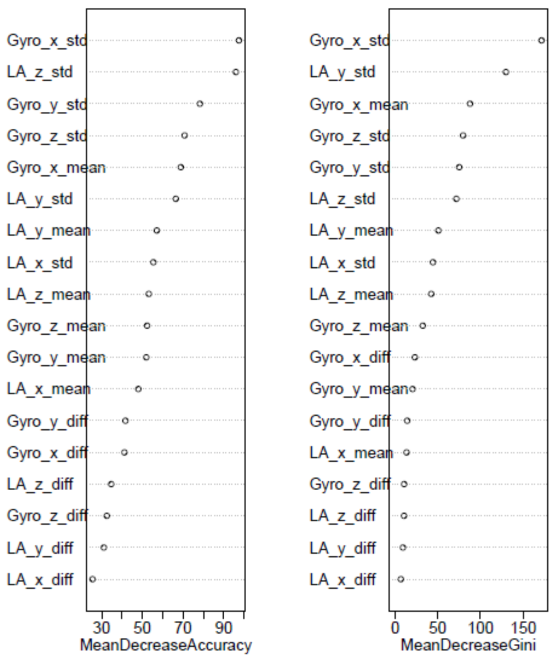

Fig. 5. Importance of features from the random forest model.

Feature extraction was done over a sliding window. Within the window, various metrics for the distribution of data were calculated and used as features.

The following features were calculated within the sliding window for each attribute:

- Mean
- Standard Deviation
- Difference (Final - Initial)

These metrics were calculated for both Gyro sensor readings and linear acceleration for all 3 axes for a total of 2 (data) × 3 (axes) × 3 (metrics) = 18 features.

Importance of the features out of these 18 were identified using random forest feature importance. Fig. 5 shows the importance of features from the random forest model.

The most important feature is given as the Gyro_x standard deviation. This can be considered accurate when considering the physical meaning of the reading with respect to the two gestures. The VR gestures will show a high variance in rotation around x axis, while the navigation gesture shows very little variance in the rotation around x axis. This has been captured by both methods as discussed above.

IV. Results

A. Finding the Optimal Window Size

When extracting features as discussed above, using an optimal window size was critical. To get a better understanding on a good window size, the measures of goodness for the current context was identified.
The measures used were:

- Accuracy: Value taken by doing cross validation on the random forest.
- Latency (delay): The time taken to identify the change in the prediction after class labels switch in time series data

These values were measured while varying the window size using a random forest for classification. The following two graphs illustrate the results.

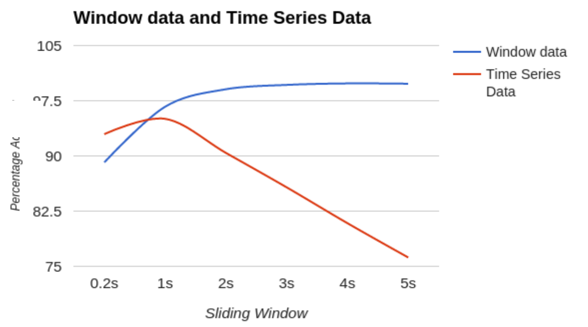

Fig. 6. Cross-validation and Time series accuracy against sliding window size.

Observing the above results, in order to get a satisfactory accuracy with low latency, a window size of 2 seconds was identified as the most appropriate.

B. Training Models

Using the features over a 2 second window, SVM [7], Random Forest [8] and ANN [9] learning models were generated with varying configuration parameters. The Random Forest was grown with 2000 classification trees. The SVM was of type C-Classification with a radial kernel, with parameters c and γ set to 1 and 0.055. The Neural Network configurations will be discussed in detail in the relevant section.

C. Testing

The data was collected while 20 individuals were using the VR application, for a span of 3 minutes each, with aforementioned Android application recording sensor data with a frequency of 0.2 seconds. This produced a data set consisting of 18000 data points. The dataset was then partitioned into test and training sets.

The three models were run on training data sets using 1/3 of the dataset, 2/3 of the dataset and finally the entire training set. A holdout testing set was used to measure the accuracy of the training model. The accuracy of the results is shown in Fig. 8.

D. Evaluation Schemes

Accuracy and latency were taken as the evaluation criteria for this scenario.

- Accuracy

1) Accuracy = (TP + TN) / (P + N)

where TP, TN, P and N corresponds to number of true positive, true negative, total positive and total negative results respectively.

2) Probability of predicted value matching the expected value.

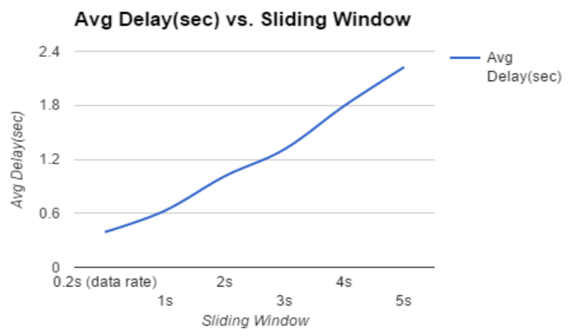

Fig. 7. Average delay against sliding window size.

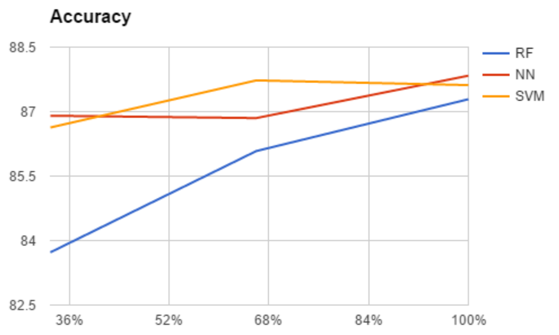

Fig. 8. Accuracy for different classification methods against different training set size.

- Latency

1) Measure of the response time of the model.
2) Individual latency for a transition is defined as the time taken to correctly detect a change in the gesture since the actual transition in test data (i.e. VR to Navigation or Navigation to VR). This is measured in seconds.
3) Latency for a model is taken as the average of individual latencies for all transitions in the test set.

Both of these metrics were calculated for each method, while varying the training set size. The graphs shown in Fig. 8 and Fig. 9 illustrate the results for accuracy and latency respectively.

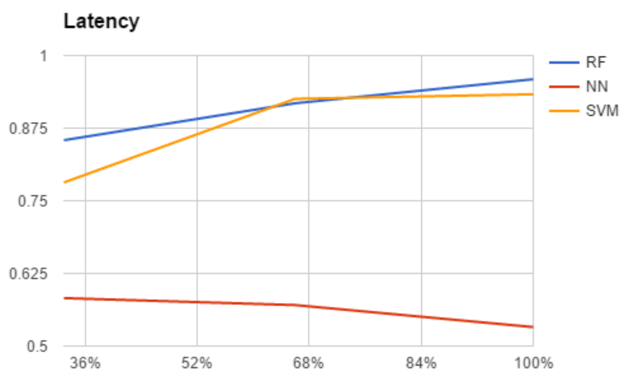

Fig. 9. Latency of different classification methods against different training set size.

As shown by Fig. 8 and Fig. 9, ANN was the most suitable for this work with a higher accuracy and a lower latency. Therefore, multiple architectures of ANNs were tested to find the best model configuration. Fig. 10 shows the results of different architectures with different configurations of hidden layers. The number of neurons in each hidden layer of the tested models is shown within brackets accompanied by the backpropagation algorithm used. All the models had logistic function as the activation function of neurons.

Artificial Neural Networks to classify Navigation and VR movements were created using the "neuralnet" package for R. [9], [10].

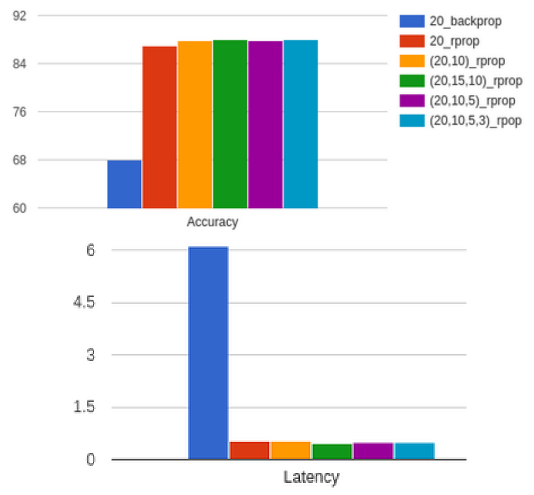

Fig. 10. Comparison of accuracy and latency of ANNs with varying hidden layers and algorithms.

The tests were carried out for 100% of the training set to determine a configuration for the ANN to be used for final testing. The results show that algorithm "backprop" (backpropagation) is suboptimal to "rprop+" (resilient backpropagation with weight backtracking) and shows a considerable improvement in accuracy and latency with two hidden layers when compared to a single hidden layer. However ANNs with different node combinations using two, three and four hidden layers show no significant improvement, although the complexity and size of the network increases.

V. Conclusion

When considering the results we obtained through the tests, we can come to the conclusion that the model which would be better for the purpose of our application is the Neural Network. Using a neural network it is possible to obtain a result with high accuracy with very low latency.

While the other two models also provide results with sufficient accuracy, the latency is quite high. Considering the practical aspects of our application, it is very important to use the mechanism which responds faster to the user input.

In order to increase the accuracy, an ensemble method was not considered as the accuracy was sufficiently achieved for this scenario and the selected method has the lowest latency. However a slight improvement may be achieved by further tuning the parameters of the neural network.

Acknowledgement

The authors would like to thank Dr. Amal S. Perera, Senior Lecturer at the Department of Computer Science and Engineering, University of Moratuwa, for the guidance and advice regarding Machine Learning related components of this research work. Without his support this research work would not have been possible.

References

[1] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 2, p. 74, March 2011.

[1] L. Sun, D. Q. Zhang, B. Li, B. Guo, and S. J. Li, "Activity recognition on an accelerometer embedded mobile phone with varying positions and orientations," pp. 548–562, October 2010.

[2] D. Chu, N. D. Lane, T. T.-T. Lai, C. Pang, X. Y. Meng, Q. Guo, F, Li, and F, Zhao, "Balancing energy, latency and accuracy for mobile sensor data classification, in *Proc. the 9th ACM Conference on Embedded Networked Sensor Systems*, New York, USA, November 2011, p. 54.

[3] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, *Activity Recognition from Accelerometer Data*, July 2005, pp. 1541–1546.

[4] Kim *et al*., "Efficient dynamic time warping for 3D handwriting recognition using gyroscope equipped smartphones," *Expert Systems with Applications*, Sep. 2014.

[5] M. Sensors. (2015). 'Motion Sensors|Android Developers'. [Online]. Available: http://developer.android.com/guide/topics/sensors/sensors_motion.html

[6] D. Meyer, *Misc Functions of the Department of Statistics*, Probability Theory Group (Formerly: E1071), TU Wien, 1st ed. 2015.

[7] A. Liaw, *Breiman and Cutler's Random Forests for Classification and Regression*, 1st ed. 2015.

[8] S. Fritsch and F. Guenther, *Training of Neural Networks*, 1st ed. 2015.

[9] F. Günther and S. Fritsch, *Neuralnet: Training of Neural Networks*, 1st ed., 2015.

Vipula Dissanayake is an undergraduate at University of Moratuwa, Sri Lanka. He got the B.Sc. of engineering specialized in computer science & engineering in 2016.

He has worked as a trainee software engineer at the Research & Development Team of CodeGen International Pvt(Ltd.) where his work included machine learning based web page analyzer.

His interests of research are in the areas of computer vision, machine learning and mobile technologies.

Sachini Herath is an undergraduate at University of Moratuwa, Sri Lanka. She got the B.Sc. of engineering specialized in computer science & engineering in 2016.

She has worked as a trainee software engineer at AdroitLogic Pvt (Ltd.) where her work included developing the database connection pool, UltraCP and tools for ESB monitoring. She is also an active contributor to Drupal, and developed the modules RDF UI, RDF Builder and Linked Data Tools.

Her research interests include machine learning, data mining and semantic web.

Sanka Rasnayaka is an undergraduate at University of Moratuwa, Sri Lanka. He got the B.Sc. of engineering specialized in computer science & engineering in 2016.

He has worked as a software engineering intern at WSO2 Inc. where he worked with the CEP geo-dashboard to create a demo using realtime london traffic data for predictions which was exhibited at the strata hadoop world conference in NY.

His research interests lie in the fields of computer vision and machine learning.

Sachith Seneviratne is an undergraduate at University of Moratuwa, Sri Lanka. He got the B.Sc. of engineering specialized in computer science & engineering in 2016.

His recent work includes a consultancy for NASA's QuakeFinder project in collaboration with TopCoder Inc., where he was part of a team responsible for creating a crowdsourced contest aimed at detecting the time and geographical location of earthquake occurrences using sensor data. He has worked as a research intern at the Intelligent Systems Group at RMIT University, Melbourne focusing on the scalability of intelligent systems. He has also worked with TopCoder Inc. on a Natural Language Processing research project for Harvard University and the Scripps Research Institute which explored the possibility of extracting accurate expert biological annotations based on crowd-sourced (Amazon Mechanical Turk-based) biological annotation data.

His research interests include machine learning for bioinformatics, smart cities and IoT sensor data mining, natural disaster and atrocity prediction as well as computer vision for virtual and augmented reality.

Rajith Vidanaarachchi is an undergraduate at University of Moratuwa, Sri Lanka. He got the B.Sc. of engineering specialized in computer science & engineering (2016). He has worked as an intern Software Engineer at Kartosoft GmbH., Lüneburg, Germany, where his work included genetic algorithm based cargo loading optimizing.

His research interests lie in the fields of computer vision, machine learning & natural language processing.

Chandana Gamage has a PhD from Monash University in Melbourne, Australia specializing in computer network security and cryptography and completed his masters in engineering from the Asian Institute of Technology in Bangkok, Thailand and bachelors in computer science & engineering from University of Moratuwa in Sri Lanka where he presently works as a senior lecturer. His research interests are in the fields of computer security, enterprise architectures and mixed reality systems.