6DoF Tracking in Virtual Reality by Deep RNN Model

1st Yun-Kai Chang
Department of Computer Science
National Chung Cheng University
City, Country
tony091520933@gmail.com

2nd Mai-Keh Chen
Department of Computer Science
National Chung Cheng University
City, Country
o860116@gmail.com

3rd Yun-Lun Li
Department of Computer Science
National Chung Cheng University
City, Country
xu3mp6xjp6@gmail.com

4nd Hao-Ting Li
Department of Computer Science
National Chung Cheng University
City, Country
remidream@gmail.com

5th Chen-Kuo Chiang
Department of Computer Science
National Chung Cheng University
City, Country
cck.adrian@gmail.com

***Abstract*—In this paper, a novel coordinates tracking method is proposed for Virtual Reality (VR) environment using sensor signals. The purpose is to extend movement tracking in VR from 3 Degrees of Freedom (DOF) of rotation to 6DOF of position plus rotation. As a result, we can track VR coordinates without using controller or handler provided by VR devices. An RNN-based model is proposed to predict displacement of positions in each timestamp given measured acceleration and Euler angles from sensor signals. Experiments demonstrate that it is effective to predict correct position displacement, which not only models the relationship between sensor signals and displacement but also handles the cumulative errors during tracking.**

***Index Terms*—Coordinate tracking, Movement tracking, Virtual Reality, 6DoF, RNN, Deep Learning**

I. Introduction

In recent years, with the significant progress of virtual reality (VR), more and more fields are applied to VR, like education, medicine, military etc. [1]–[3] Mobile VR (MVR) is one of the important application in VR. Unlike HTC Vive (Vive) [4] or Oculus Rift [5], which need a computer and a set of related equipment of VR, users only need a smartphone and a head-mounted displays (HMD) to use VR. Although MVR isn't effective as large VR, it's still popular due to tiny, lightweight and portable characteristics [6].

Positional tracking is an important issue for VR. It can track the movement in real space by detecting the position of the HMD, controllers or body parts within Euclidean space and represent them inside VR. Basically, positional tracking in VR have to depend on space locator. For example, Vive needs two locators, which are called base station and placed on the diagonal of the indoor space. However, because MVR doesn't have the locator like Vive or Oculus Rift, it needs another ways to track the position.

Inertial tracking is one of positional tracking methods whose data is collected from accelerometers and gyroscopes. Accelerometers can measure linear acceleration and gyroscopes can measure angular velocity. Common inertial tracking uses low-cost inertial measurement units systems (IMU) based on Microelectromechanical systems (MEMS) to track the acceleration and orientation with high update rates. Generally, acceleration can be integrated to obtain the velocity and then be integrated again to obtain the displacement. However, restricted by size and cost, the accuracy of the MEMS IMU is greatly limited, and causes data (acceleration and Euler angle etc.) error [7]. With each acceleration error accumulation, the accuracy of the position integrated twice by acceleration will get less and less precise.

To address the problems of error accumulation, we propose a sequential deep learning method to predict displacement of positions with acceleration and Euler angle. In recent years, recurrent neural networks (RNN) have achieved remarkable success in dealing with sequential data. Body motions are strongly temporal dependent so that it's appropriate to use recurrent model to learn the temporal representations [8]–[10]. We consider that sequential model can not only predict position displacement with acceleration and Euler angle collected by IMU but also learn the cumulative errors during tracking body motions.

However, RNN is difficult to train due to vanishing gradient or exploding gradient problems [11], [12]. To avoid above problems, we adopt long sort-term memory (LSTM). LSTM uses memory to strengthen current and uses three control gates (Input Gate, Forget Gate and Output Gate) so that it has a much ability to handle the long-term dependencies [7], [12]. In addition to standard LSTM networks, we also use bidirectional LSTM (Bi-LSTM) networks, which is stacked two LSTM networks in forward and backward directions. Unlike standard LSTM networks, which can only consider the past information, Bi-LSTM networks can capture both past and future information by two opposite temporal order in hidden layers [13]–[15]. Furthermore, except for the LSTM networks, we also use Temporal Convolutional Networks (TCN), which

Identify applicable funding agency here. If none, delete this.

can capture long-range patterns by temporal convolutions. Recent research shows that the convolutional architectures can achieve a good performance as recurrent models [16]–[19].

The rest of the paper is organized as follows. In Section II, is presented our data process and deep learning model in detail. In Section III, we describe details about the setup of experimental process and experimental results. Finally, we have our conclusion in Section IV.

II. Methodology

First of all, we use Vive and IMU to collect data, coordinates of Vive controller and sensor signals (acceleration and Euler angle) of IMU, respectively. Next, we do some preprocessing of the data. After that, we take processed data as input of the model, and then model will predict the displacement of position in each timestamp. Finally, the displacements is added up to get the position of each timestamp. In this section, data preprocessing and position prediction would be discussed.

A. Data Preprocessing

The low-cost IMU is sometimes unstable in frame rate during collecting data. In the experiments, we set the frame rate of IMU with 50 Hz which means that data will be collected per 0.02 second. Unfortunately, sometimes IMU could occur delay, and the time gap of two samples may be reduced to 0.01 second or increased to 0.05 second. To deal with above situation, we set up the thresholds. If time gap is less than 0.01 second, the two samples will be removed; if time gap is greater than 0.1 second, we interpolate the dropped data using linear interpolation.

Apart from doing preprocessing for IMU data, we also need to process the coordinates data from Vive. First, we would activate Vive controller and IMU Simultaneously when start collecting the data. Notably, the activation time of Vive controller and IMU would be slightly different, which would cause a slight time error of each timestamp between Vive controller and IMU. To align time of the two, we use linear interpolation to interpolate the coordinate of each timestamp of Vive controller to the timestamp of IMU. Next, we convert the coordinates of Vive controller into displacement for each timestamp. Then, we would interpolate the displacement at this timestamp through the previous and next data if IMU occurs delay (time interval of two samples over 0.05 second). After that, we would filter the displacement using median filter if amplitude of displacement over 0.06. Finally, use arithmetic mean filter to filter all displacements.

B. Position Prediction Method

For position prediction, instead of predicting the position directly, we predict the displacement first, and then add it to obtain the position. To get displacement in each time t, we trained three different models, including LSTM, Bi-LSTM and TCN, using acceleration and Euler angle as input of model. In section III, we would show the results of three different models.

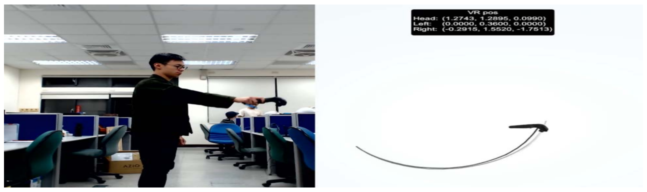
VR pos
Head: (1.2743, 1.2895, 0.0990)
Left: (0.0000, 0.3600, 0.0000)
Right: (-0.2915, 1.5520, -1.7513)

Fig. 1. Data collection with HTC Vive and IMU. The collection environment was built by Unity3D.

III. Experiments

In our experiments, we used mean square error (MSE) which is usually used in regression problem to evaluate performance of our method. The smaller the value of MSE, the better the result. MSE formula can be illustrated by Equation (1):

$$MSE = \Sigma_{i=1}^{n}(y_i - y_i^p)^2 \tag{1}$$

In this section, we would present how we collect and preprocess the data, and compare the displacement prediction results of different models, and then compare our method with traditional way that obtains the displacement by integrating the acceleration twice.

A. Data Collection

We invited 8 people as subjects to collect the movement data. Before starting, subjects need to hold the Vive controller and IMU in hand. The frame rate was set to 50 both of Vive controller and IMU. After preparation, each subject would be free to do any movement in ten minutes. There would be about 80 minutes of movement data in the end. Fig. 1 is a schematic diagram of data collection.

B. Model Training

First, we tried the LSTM, Bi-LSTM and TCN. Table I shows the comparison results of MSE loss. Unexpectedly, the basic LSTM performs the best among the three models. We surmised that it could be possibly due to the short-term correlation of human actions, so it isn't often required to consider the long-term complexity. In other words, Bi-LSTM network and TCN which architectures are more complex could be possibly over-predicted instead so that their prediction results aren't as good as LSTM network in this case.

	LSTM	Bi-LSTM	TCN
MSE Loss	0.000033	0.000045	0.000052

TABLE I
MSE RESULTS IN TEST SET FOR DIFFERENT MODELS

According to the result above, we used LSTM network as the main structure and continue to adjust the parameters of model. Table II presents the result of MSE loss with different layers, and we found that 3 layers was the best numbers of the LSTM layers.

	1 layer	2 layers	3 layers	4 layers
MSE Loss	0.00004	0.000024	0.0000103	0.000024

TABLE II
MSE RESULTS IN TEST SET FOR DIFFERENT NUMBER OF LSTM LAYERS

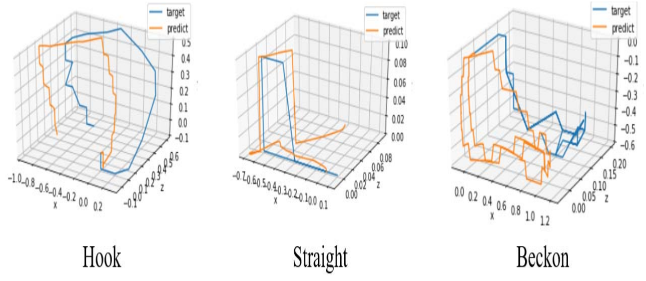

Fig. 2. Prediction result by LSTM. The picture shows the prediction result of hook punch, straight punch and beckon. Blue line is target, and orange line is prediction result

After training the model, we recorded some motion data as test set. Fig. 2 expresses the 3d visualization results of some motions, like hook punch, straight punch and beckon. We can see that the contours of the motion which predicted by the model (orange line) is similar to the target (blue line).

C. Acceleration Integration

The data collected by IMU includes acceleration and Euler angle. Generally speaking, displacement can be obtained by integrating acceleration twice. In addition to using model to predict displacement, We also tried to integrate the acceleration collected by IMU to get the velocity by Equation (2) where v_t is the velocity at time t; a_t is the acceleration at time t. The integration result of velocity is in Fig. 3. We can see the x-axis has an increasing positive deviation as time goes by; y-axis and z-axis also have small deviations respectively. Next, we used Equation (3) where d_t is the displacement at time t to integrate the velocity and get the displacement. The displacement of the target, prediction and integration compared result is in Fig. 4. Green line is integration result, as for blue and orange line are the target and prediction result respectively. We can see that integration result is much worse than prediction result of model. We speculate that it should be related to the errors of IMU. When testing IMU, we found that it still has a slight acceleration even if IMU is static. These errors would be accumulated and cause a large deviation in displacement when integration acceleration twice.

$$v_t = v_{t-1} + a_t * t \tag{2}$$

$$d_t = v * t + \frac{1}{2} * a_t * t^2 \tag{3}$$

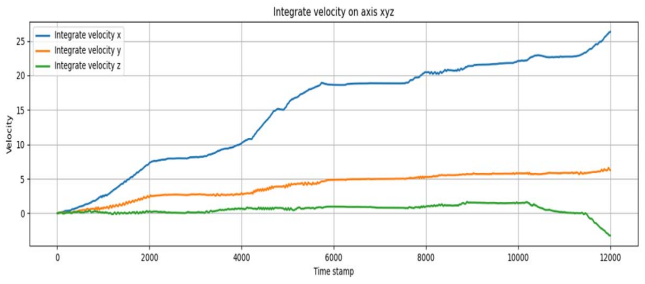

Fig. 3. x, y, z axis of the velocity integrated by the acceleration. Blue line is x axis; orange line is y axis; green line is z axis.

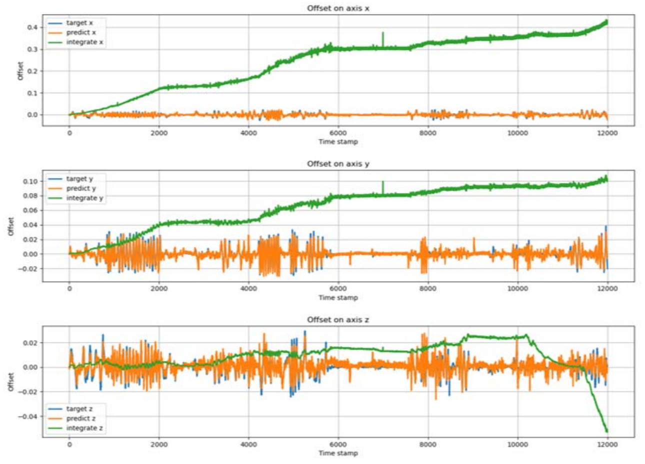

Fig. 4. x, y, z axis of the target, prediction and integration displacement. Green line is integration result, as for blue and orange line are the target and prediction result respectively.

IV. Conclusion

In this paper, we propose an deep learning based method to predict the displacement by using sensor signals like acceleration and Euler angle. The method not only learns the relationship between sensor signals and displacement but also handles the accumulative errors caused by low-cost IMU during tracking in MVR which doesn't have the space locator. In addition, to make the model more effective, we carried out various preprocessing for the data collected by HTC Vive and IMU to make the input data of the model more clear. Furthermore, we also attempted to traditional way which gets displacement by integrating acceleration twice. The results show that the accumulative errors would be increased over time, and the result predicted by model is much better than the traditional integration.

References

[1] E. D. Innocenti, M. Geronazzo, D. Vescovi, R. Nordahl, S. Serafin, L. A. Ludovico, and F. Avanzini, "Mobile virtual reality for musical genre learning in primary education," *Computers Education*, vol. 139, pp. 102 – 117, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360131519300971

[2] L. Li, F. Yu, D. Shi, J. Shi, Z. Tian, J. Yang, X. Wang, and P. Zhang, "Application of virtual reality technology in clinical medicine," *American Journal of Translational Research*, vol. 9, pp. 3867–3880, 09 2017.

[3] K. Bhagat, W.-K. Liou, and C.-Y. Chang, “A cost-effective interactive 3d virtual reality system applied to military live firing training,” *Virtual Reality*, vol. 20, pp. 127–140, 04 2016.

[4] Htc vive. [Online]. Available: https://www.vive.com/tw/

[5] Oculus rift. [Online]. Available: https://www.oculus.com/

[6] M. Saker and J. Frith, “From hybrid space to dislocated space: Mobile virtual reality and a third stage of mobile media theory,” *New Media Society*, vol. 21, p. 146144481879240, 08 2018.

[7] C. Chen, X. Lu, A. Markham, and A. Trigoni, “Ionet: Learning to cure the curse of drift in inertial odometry,” *ArXiv*, vol. abs/1802.02209, 2018.

[8] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng, “View adaptive recurrent neural networks for high performance human action recognition from skeleton data,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[9] J. Liu, G. Wang, P. Hu, L.-Y. Duan, and A. C. Kot, “Global context-aware attention lstm networks for 3d action recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[10] N. Jaouedi, N. Boujnah, and M. S. Bouhlel, “A new hybrid deep learning model for human action recognition,” *Journal of King Saud University - Computer and Information Sciences*, vol. 32, no. 4, pp. 447 – 453, 2020, emerging Software Systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1319157819300412

[11] Yong Du, W. Wang, and L. Wang, “Hierarchical recurrent neural network for skeleton based action recognition,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1110–1118.

[12] R. Jozefowicz, W. Zaremba, and I. Sutskever, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 2342–2350. [Online]. Available: http://proceedings.mlr.press/v37/jozefowicz15.html

[13] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, “Action recognition in video sequences using deep bi-directional lstm with cnn features,” *IEEE Access*, vol. 6, pp. 1155–1166, 2018.

[14] P. Zhou, W. Shi, J. Tian, Z. Qi, B. Li, H. Hao, and B. Xu, “Attention-based bidirectional long short-term memory networks for relation classification,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 207–212. [Online]. Available: https://www.aclweb.org/anthology/P16-2034

[15] A. Zhu, Q. Wu, R. Cui, T. Wang, W. Hang, G. Hua, and H. Snoussi, “Exploring a rich spatial–temporal dependent relational model for skeleton-based action recognition by bidirectional lstm-cnn,” *Neurocomputing*, vol. 414, pp. 90 – 100, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231220311760

[16] S. Bai, J. Kolter, and V. Koltun, “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling,” 03 2018.

[17] Y. Dauphin, A. Fan, M. Auli, and D. Grangier, “Language modeling with gated convolutional networks,” 12 2016.

[18] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, “Temporal convolutional networks for action segmentation and detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[19] F. Aparecido Garcia, C. Mazzoni Ranieri, and R. Aparecida Francelin Romero, “Temporal approaches for human activity recognition using inertial sensors,” in *2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE)*, 2019, pp. 121–125.