A NEW APPROACH FOR TRACKING HUMAN BODY MOVEMENTS BY KINECT SENSOR

Adjeisah Michael[1,], Zhao Chen[1], Guohua Liu[1], Yang Yi[2]*

[1]*School of Information Science and Technology, Donghua University, Shanghai, China*
[2]*School of Information Science and Engineering, Lanzhou University, Lanzhou, China*
**madjeisah@hotmail.com*

Keywords: Kinect, tracking, body movements, depth data, extended Kalman filter

Abstract

With the invention of the Microsoft Kinect sensor, high-resolution depth and visual sensing has become available for prevalent use in the smart home and medical rehabilitation. In this paper, we introduce an innovative approach for effective body movement tracking using Kinect Xbox 360 with a limited tracking system. A relatively scaled hand cursor mechanism is used for our system of interaction. Instead of tracking the whole body of the participant in the Kinect Depth space which produces 20 joints, we limit the tracking to only 2 joint (left and right hand) for the same action. Further, we have engaged Extended Kalman Filter to improve skeleton joint estimation which smooths the joint coordinates, placing the Z axis in a high level of calibration in order to make it work with X and Y coordinates simultaneously with a relatively high accuracy.

1 Introduction

Microsoft Kinect sensor, interacting with the Kinect Software Development Kit (SDK), can track the body movements and gestures of the user without the need of them holding any device. This is done only by using RGB camera and an infrared depth detection sensor [2]. These somatosensory interfaces are expected to allow more intuitive interaction with computers and provide a greater level of control than that offered by the traditional keyboard and mouse and other input devices [4]. Due to its cost effective, the Kinect can be applied to medical care and home automation, which are two important aspects of smart city.

Many of Kinect-based experiment aimed at using the X and the Y axes for their findings. Jody Shu et al. [3] [4] in their paper introduced the Extended Kalman Filter for improving the accuracy and smoothness of Kinect skeleton-joint estimates, but the focus was on the head pose based on the X and Y plane. Moreover, they did not conduct user-experience experiments on their findings. Previous studies [5] explored the accuracy performance in gesture-based game and hand written recognition as well.

However, only a very small number of them contributed to addressing the problem of the decrease in localization accuracy. The methods focusing on the Z-index in unification with X and Y axis are even fewer.

Our study aims at limiting joint tracking to only 2 joint (left and right hand), instead of 20 joints, to perform all actions in Kinect space accurately. To fulfil the task, we adopt a relatively scaled cursor for the hand joint tracking, which uses the Z elevation simultaneously with the X and Y axes yielding a sensing of depth and a great deal of intuitive interaction. We also explore Extended Kalman Filter (EKF) as a fine tuning cursor mechanism for accurate calibration in a relatively open social context devoid of human distraction. Kinect provides an open framework that produces 640 X 480 RGB, depth images at 30fps and 20 joints detection for pose classification; this was used to detect the above mentioned values as shown in Figure 1.The experimental results validate the effectiveness of our method in body movement tracking.

(a) (b) (c)

Figure 1: Screen capture of Kinect depth, color and skeleton data superimposed in real time. (a): tracked skeleton on depth map. (b): raw depth image frame(c): RGB image frame.

2 Methodology

2.1 Kinect and smart home

One fascinating feature of the Kinect sensing device in Xbox One stems from its infrared sensor, which can identify objects in a completely dark room. It can recognize people and track bodies even without any light visible to the naked eye. It can identify a hand pose from four meters away and remember your identity even minus room illumination. With the new console, as many as six users can crowd into one scene. Users get a better experience no matter if they're standing close by, further away, or on the periphery of the room to control an Internet browser or Google map by their body movements. Therefore, your home utility and leisure-time efficiencies can be improved with the technology based on Kinect.

Kinect also integrates a depth processor [1]. The field of view of Kinect is limited. As shown in Figure 2, a very single joint has 3 values: X, Y, and Z in a Cartesian coordinate system. Table 1 shows the Kinect's principle of measurement.

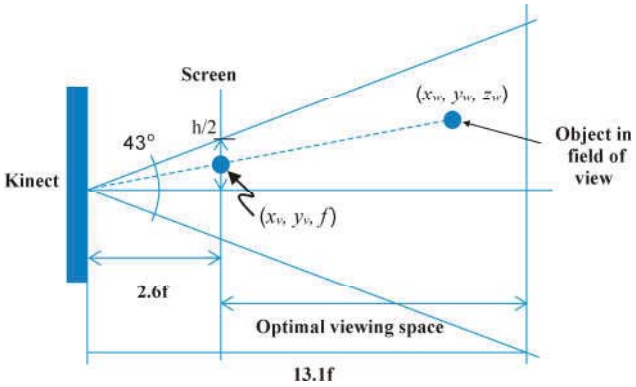


Figure 2. Kinect field of view and its relations to X, Y and Z coordinates.

The (0, 0, 0) point is the position of the sensor. Every other point is measured in terms of the position of the sensor as listed below:

- X is the position in the horizontal axis.
- Y is the position in the vertical axis.
- Z is the position in the depth axis.

The length of a vector is given by:

$$\sqrt{X^2 + Y^2 + Z^2} \tag{1}$$

One can manipulate these few lines of code to know exactly how far or how close someone is to the Kinect sensor as shown below.

Table 1. Depth Measurement and the Kinect Distance

Pseudo code 1

public Method (Argument type *Argument value*){
Return Statement (
(point.X * point.X) +
(point.Y * point.Y) +
(point.Z * point.Z)
)
}
declaration statement = body joints [Joint Type] via Position
declaration *distance* = Length (declaration statement)

Webb J, et al [1] further explained that, objects farther away from the camera have a greater lateral range than objects nearer to Kinect. This means that height and width pixel dimensions, such 640 X 480, do not correspond with a physical location in the camera's field of view [3]. The depth value of each pixel, however, does map to a physical distance in the field of view. Each pixel represented in a depth frame is 16 bits, making the bytes-per-pixel property of each frame a value of two [7]. When skeletal tracking identifies a user in the field of view, the location of the user is also reported in the depth plane. For every pixel in the depth map, the three lowest-order bits contain a user's index information [8]. An index of "0" indicates no user at that given pixel; an index of "1" through "6" indicates the presence of a user. Figure 2 illustrates the Kinect field of view and its relations to X, Y and Z coordinates.

2.2 Framework

A detailed framework for our relatively scaled hand cursor mechanism is displayed in Figure 3 below. The steps of the proposed system are as follows.

1) Collect data of human body movement by Kinect. The user needs to be in front of the sensor, making sure Kinect can see their head and upper body between a distance of 0.8 meters (2.6 feet) and 4.0 meters (13.1 feet) away, so that the user can be sensed.
2) Converse the captured images into depth images.
3) Track the whole skeleton by the joints being sensed. Kinect usually tracks up to six users and up to 20 joints for each skeleton. In our case, we limit the joint to 2 joints for a clear scalability. Only the left and the right hand are used to indicate how the tracked user should interact in terms of distance (forward and backward) and also horizontal transposition (left and right). One can use the shoulder or the entire body for the same purpose. For example, when controlling Google map with Kinect in your room, one can track only two hands instead of 20 joints to perform an action without tracking the entire body into the tracking framework which will need the participant to move around the Kinect space.
4) Extract pose features and identify them. A pattern of infrared light is used to calculate the depth of a participant in the field of view allowing the identification of body parts. The skeletal tacking is optimized to recognize user in a standing or sitting posture. The default initial pose for our application is set to standing.
5) Finally, recognize body movements. This is where the visualization of the process takes place based on a participant's hand position. The cursor changes in scale depending on the depth of the user. The closer the user is to the screen, the larger the cursors, and the farther from Kinect, the smaller the scale as mentioned before.

To express the 3-Dimensional (3D) coordinates of the participant position, we consider a depth coordinate system with its origin at the center of the standpoint of the infrared camera. The Z axis is perpendicular to the image horizon [7] [10] in the direction of the object, the X axis is perpendicular to the Z axis in the direction of the baseline between the infrared camera center and the projector, and the Y axis is perpendicular to X and Z, resulting in a right handed coordinate system. We make the Z axis work with the X-Z plane in order to exploit depth data and achieve better performance in body movement recognition.

The accuracy of the tracking of skeleton joints by Kinect is limited by the hardware design of Kinect sensor and the algorithms Microsoft chooses to implement for estimating the joints' coordinates. The SDK released some Application Processing Interface (API) for smoothing parameters while setting up the skeleton stream [2] [6] but this had minor effect in achieving our purpose. Therefore, the smoothing algorithm used in this work is EKF. The model of EKF is explained as follows [11].

$$\frac{dx}{dt} = f(x, y), \tag{2}$$

Since we are dealing with 3D spaces, equation (2) can be written as:

$$\frac{dx}{dt} = f(x, y, z), \tag{3}$$

where the x vector denotes the filter variables, the y vector denotes the algebraic variables and z represents coordinate variables. The difference form of (3) can be written as:

$$x_k = x_{k-1} + f(x_{k-1}, y_{k-1}, z_{k-1})\lambda \equiv g(x_{k-1}, y_{k-1}, z_{k-1}), \tag{4}$$

where k is the time step number and λ is the time step in seconds. The Kinect skeleton frame becomes available at 30Hz, so λ = 1/30s. Measurements at the time step k can be represented as a vector of a nonlinear function h combined with the filter variables x and measurement noise v:

$$z_k = h(x_k, v_k). \tag{5}$$

The propagated error between the restrained and the deliberate values is prearranged as [11]:

$$\varepsilon_k = z_k - h(x_k, v_k). \tag{6}$$

The EKF is a two-step prediction correction process [11] [12]. The prediction step is a time update using the difference equation (4), which predicts the filter variables of the next step, x_k, based on the values produced by the previous steps, x_{k-1}. In addition to prediction of the filter variables, the a priori estimated error covariance is also predicted. The correlation step takes the measurement z_k and uses the error ε_k calculated from (6) to correct filter variables x_k. The prediction step and the correction step for EKF are formulated by equation (7), (8) for Measurement update step and Time update step (9), respectively, as follows [3] [12]:

$$\begin{cases} \hat{x}_k^- = \hat{x}_{k-1} + \dfrac{dx}{dt}|_{k-1}\,\lambda = \hat{x}_{k-1} + f(\hat{x}_{k-1}, z_{k-1})\lambda \\ P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T = A_k P_{k-1} A_k^T \end{cases}, \tag{7}$$

$$\begin{cases} K_k = P_k^- H_k^T \left(H_k P_k^- H_k^T + V_k R_k V_k^T\right)^{-1} \\ \hat{x}_k = \hat{x}_k^- + K_k\left(z_k - h(\hat{x}_k^-, 0)\right) \\ P_k = (I - K_k H_k) P_k^- \end{cases}, \tag{8}$$

$$\begin{cases} \hat{X}_{k+1}^- = F\hat{X}_k \\ P_{k+1}^- = \mid FP_k F^T + Q \end{cases}, \tag{9}$$

where K is the Kalman gain matrix, P is the estimation error covariance matrix, R is the propagated noise covariance matrix, and A, H, and V are the Jacobian matrices as defined in (10) and the matrix F is given in block form by (11). The R matrix is the measurement error covariance matrix and needs to be tuned if the contribution due to measurement noise is not known. Larger values of R place heavier weight on the predicted value while smaller values of R place the more weight on the measured values:

$$A = \frac{\partial g}{\partial x}, H = \frac{\partial h}{\partial x}, \mathrm{V} = \frac{\partial h}{\partial v}. \tag{10}$$

$$F = \begin{pmatrix} \mathrm{I}_{3x3} & \lambda \mathrm{I}_{3x3} \\ 0_{3x3} & \mathrm{I}_{3x3} \end{pmatrix}, \tag{11}$$

where I is elementary matrix and the size of I matches that of the time update data λ. The nonlinear differential algebraic equations of our filtered system can be readily expressed in the form of (3) and (4). Hence, EKF can be applied to the estimate the filter variables.

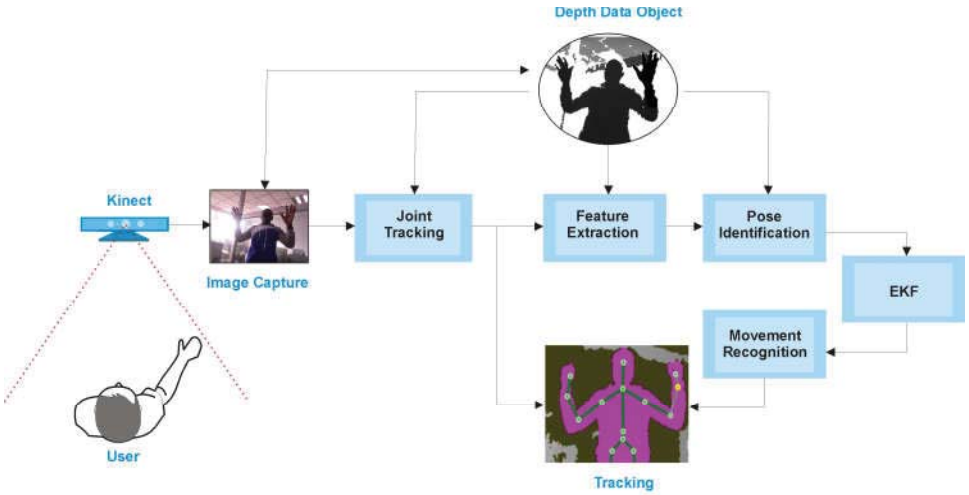


Figure 3. Framework of our proposed system.

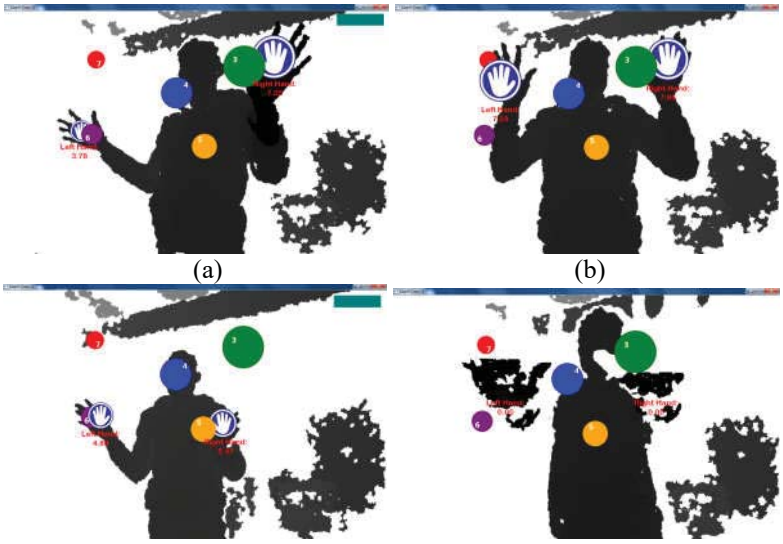
3 Results

3.1 Experiment setup

In our user-experience experiments, the Kinect was placed on a stand with a distance of 10 – 12 feet away. The participant was placed in front in a center of an area marked; restricting the Kinect's vision range where one could experience a finest interaction (that is the prime area for tracking depth data, which is between 6 and 8 feet from the Kinect's position) with user facing the sensor. The participant moved his hands around to multiple depths to see the effect within the aforementioned Kinects vision range area. Hands were placed on different positions, front and back as well as the same position. To reduce the possibility of interference, the application is set in an environment devoid of human distraction. The hand cursors scale according to the depth of the user's hand as shown in Figure 4.

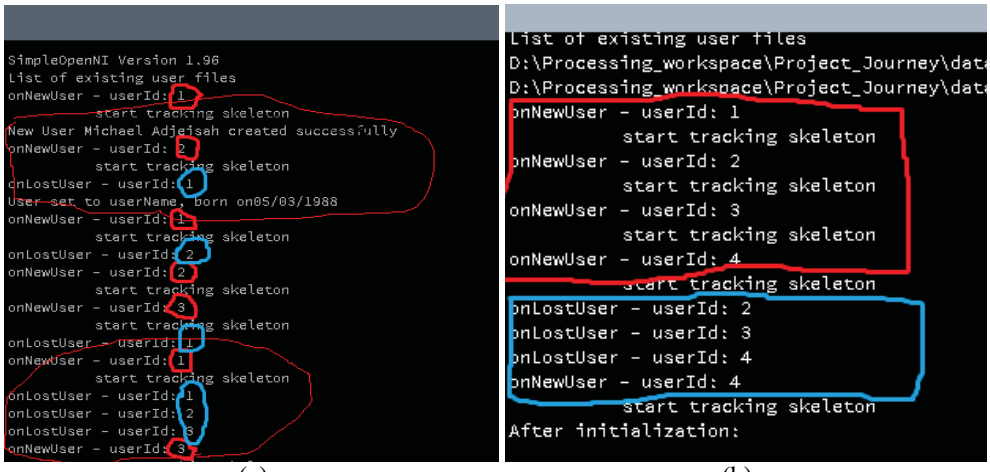
3.2 Handling skeleton data with application of EKF

EKM shows potentials in improving skeleton joint estimation and smoothness. In our method, EKF is applied to the hand cursor joints (left and right hand) of the skeleton in relation to the Z coordinate, which can enhance the performance of body movement tracking.
In our tracking state it was realized that the tracked user is not made available for active calibration for the sensor. The sensor tracks user for only short period of time as demonstrated in figure 5 (a). In the figure, one can clearly see the instability of the kinect skeleton tracking system but is further demonstrated that with EKF, skeleton tracking is smoother and more accurate. Evidence is also presented that the filter is able to continue tracking accurately [9] for longer period when the skeleton stream becomes active, as shown in Figure 5 (b).



(a) (b) (c) (d)

Figure 4. Screen capture of Kinect data base interaction in real time. Hands (a) at different depth, (b) at the same depth, (c) at almost the same depth and (d) in an ambiguous depth position.



(a) (b)

Figure 5. Skeleton tracking (a) without and (b) with the application of EKF.

Various trials were run to finally achieve fine body movement recognition as shown in Figure 6 below. The line graphs represent the states of the hand cursor movements with the horizontal axis representing the distance to Kinect coordinate system in the X axis direction (in meters) and the vertical axis representing time steps. It shows the performance of tracked skeleton from its raw state, filtered data (20 joints by EKF) and finally the truth data (i.e., the result of our system, with 2 joints by EKF). The raw data demonstrate a reduction in the accuracy and precision of skeletal tracking due to the interference with the infrared light sources. This makes a classic GIGO situation [2]. The filtered data yield a fine calibration and action recognition which is relatively high and unfit for 2 joints estimation. The truth data results in a promising performance with an accuracy which is fairly good as represented in Figure 6(a), (b), (c) and (d). The parameters of EKF can be optimized to achieve satisfactory truth data based on the interactivity of a Kinect application. The more interactive a Kinect-enabled application, the higher implementation of fine-tuned filter parameter to match the specific needs.

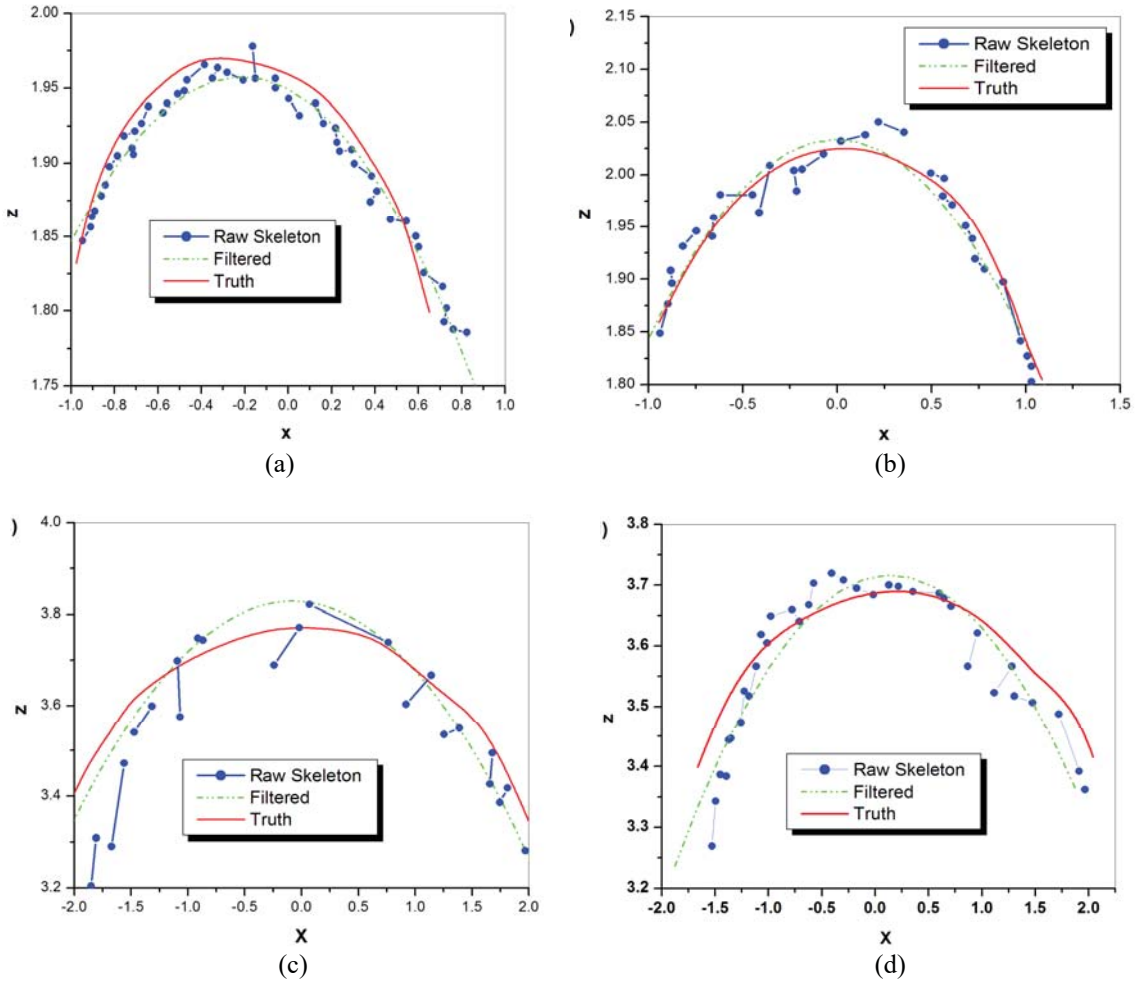
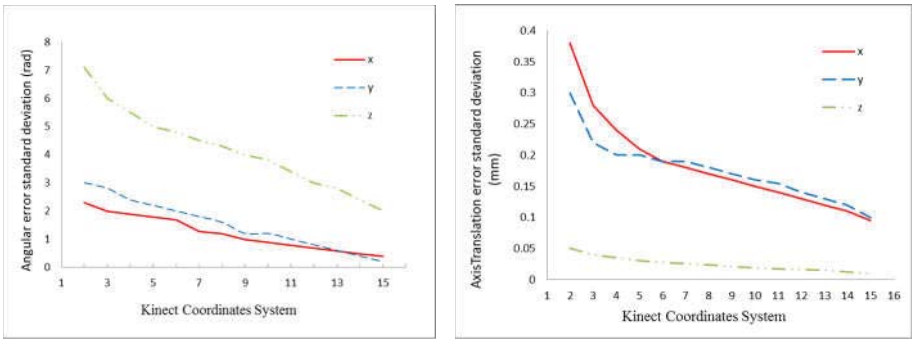


Figure 6. Accuracy evaluations.

Table 2. Summary of Calibration Accuracy Rate of X, Y and Z Coordinate Before and After EKF

Coordinates	EKF		
	Raw Skeleton	Filtered	Truth
X	72.89%	79.24%	76.48%
Y	72.34%	78.51%	76.23%
Z	68.28%	81.27%	83.37%

Table 2 shows summary of Calibration Accuracy Rate of X, Y and Z Coordinate before and after the application of EKF. A comparison from the table demonstrates a low performance of using the Z coordinates with an accuracy rate of 68.28% on the raw skeleton data, which is increased highly to the best of 81.27% for filtered data and 83.37% for truth data. It is also observed that there is a drop in accuracy rate for X and Y axis with 1.036% and 1.029% respectively. The best accuracy rate for X is 76.48%, 76.263 for Y and 83.37% for Z. It is also observed that working with X and Y axis needs little attention to filter technique to make Kinect application preform accurately as presented in Table 2 with accuracy of 72.89% and 72.34%, respectively. However, that is not the case of Z index. The raw skeleton data for Z yields low performance with 68.28%, which increased progressively on the application of EKF.



(a) (b)

Figure 7. Calibration accuracy (a) before and (b) after implementing EKF.

In Figure 7, the horizontal axis represents the Kinect coordinate system; the vertical axis (dependent variable) is the angular error standard deviation for Figure 7 (a) before implementing EKF and axis translation error standard deviation for Figure 7(b) after implementing EKF. These line graphs clearly demonstrate the improvement yielded by using the Z axis and EKF. The lower the values of the dependent variables, the higher the accuracy achieved.

4 Conclusion

In this work, a scaled cursor mechanism is proposed. It is limited to the tracking of 2 joints (the left and right hand) to perform body movement tracking in Kinect field of view instead of tracking the whole body of a participant with 20 joints. We further engage Extended Kalman Filter to improve skeleton joint estimation. The study shows the good performance of skeleton tracking upon the introduction of EKF. The result also places the Z axis in a high level of calibration simultaneously with X and Y coordinates.

Acknowledgements

This work was supported in part by Science and Technology Commission of Shanghai Municipal under Grant 17YF1427400 and in part by the Fundamental Research Funds for the Central Universities under Grant 17D111206.

References

[1] Jarrett Webb and James Ashley, “Beginning Kinect Programming with the Microsoft Kinect SDK”, Appress Publications, New York, 2012

[2] Adjeisah M., Yang Y., Lian L., “Joint Filtering: Enhancing Gesture and Mouse Movement in Microsoft Kinect Application”. 12th IEEE International Conference on Fuzzy Systems and Knowledge Discovery (FSKD’15), pp. 2528 - 2532, 2015

[3] Jody Shu et al. “Application of extended Kalman filter for improving the accuracy and smoothness of Kinect skeleton-joint estimates”, J Eng Math, 2014, 1(88):161-175.

[4] QIAO Ti-zhou and DAI Shu-ling, “Depth Data Filtering for Real-time Head Pose Estimation with Kinect” 6th International Congress on Image and Signal Processing (CISP 2013). Hangzhou, China, Dec.2013, pp. 953-958.

[5] Fakhteh Soltani, Fatemeh Eskandari, and Shadan Golestan, “Developing a gesture-based game for deaf/mute people Using Microsoft Kinect”, 6th International Conference on Complex, Intelligent, and Software Intensive Systems. Palermo, Italy. July 2012, pp. 491-495.

[6] Alana Da Gama et al. 2012. Poster: Improving Motor Rehabilitation Process through a Natural Interaction Based System Using Kinect Sensor. IEEE Symposium on 3D User Interfaces. Costa Mesa, CA USA, March 2012, pp: 145-146.

[7] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, “RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments,” Int. J. Robot. Res., vol. 31, no. 5, pp. 647-663, Apr. 2012.

[8] J. Tang, S. Miller, A. Singh, and P. Abbeel, “A textured object recognition pipeline for color and depth image data,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2012, pp. 3467-3474.

[9] R. A. Newcombe et al., “KinectFusion: Real-time dense surface mapping and tracking,” in Proc. 10th IEEE Int. Symp. Mixed Augmented Reality, Basel, Switzerland, Oct. 2011, pp. 127-136.

[10] E. Lachat, H. Macher, M.-A. Mittet, T. Landes, and P. Grussenmeyer, “First experiences with Kinect v2 sensor for close range 3d modelling,” in Proc. 3D-Arch 3D Virtual Reconstruction Visualizat. Complex Archit., Ávila, Spain, 2015, pp. 93-100.

[11] Zhenyu Huang et al, “Feasibility Studies of Applying Kalman Filter Techniques to the Power System Dynamic State Estimation” The 8th International Power Enginerring Conference (IPEC), 2007, pp.376-382.

[12] Greg Welch, and Gary Bishop, “An Introduction to Kalman Filter,” TR 95-041, Department of

Computer Scince, University of North Carolina at Chapel Hill, April 2004.