Fingertip data fusion of Kinect v2 and leap motion in unity

Bo Li[1], Chao Zhang[1,*], Cheng Han[1], Baoxing Bai[2]

*1. Changchun University of Science and Technology,
No.7186, Weixing Road, Changchun, 130022 China*

2. College of Optical and Electronical Information Changchun University of Science and Technology, No.333, Xueli Road, Changchun, 130114 China

zhangchao@cust.edu.cn

ABSTRACT. *This paper describes how the data fusion and application of Kinect v2 and Leap Motion in Unity3D are implemented. Firstly, it implements a method based on Kinect v2 to obtain fingertips. Then, it calibrates Kinect v2 and Leap Motion in two different orientations in two steps. The preliminary calibration uses a one-dimensional calibration rod algorithm, and the fine calibration keeps approximating the true value through iteration, which realizes the joint calibration of the two. Finally, this paper uses Unity3D to fuse the data of the two types of equipment and conducts human-computer interaction with the virtual object in the virtual space of Unity3D. Experiments show that the method proposed in this paper can extend the hand tracking range and improve the accuracy of the collision between the human hand and the virtual object.*

RÉSUMÉ. *Cet article décrit comment la fusion des données et l'application de Kinect v2 et de Leap Motion dans Unity3D sont mises en pratique. Premièrement, il s'applique une méthode basée sur Kinect v2 pour obtenir le bout des doigts. Ensuite, il calibre Kinect v2 et Leap Motion dans deux orientations différentes en deux étapes. La calibration préliminaire utilise un algorithme de tige de calibration unidimensionnelle, et la calibration fine continue à se rapprocher de la valeur vraie par itération, ce qui réalise la calibration conjointe des deux. Enfin, cet article utilise Unity3D pour fusionner les données des deux types d'équipement et effectue une interaction homme-machine avec l'objet virtuel dans l'espace virtuel d'Unity3D. Les expériences montrent que la méthode proposée dans cet article peut élargir la plage de suivi de la main et améliorer la précision de la collision entre la main humaine et l'objet virtuel.*

KEYWORDS: *fingertip recognition, joint calibration, data fusion, natural human-computer interaction, leap motion, Kinect v2.*

MOTS-CLÉS: *reconnaissance du bout des doigts, calibration commune, fusion de données, interaction naturelle homme-machine, leap motion, Kinect v2.*

DOI: 10.3166/ISI.23.6.143-159 © 2018 Lavoisier

1. Introduction

Natural human-computer interaction has always been the focus of research by experts and scholars in the field of human-computer interaction. The human hand, due to its many joints, high degree of freedom and various forms, is the most effective human body part in human-computer interaction and gives the most directive form of interaction. In contrast to the inconvenient equipment like data gloves, inertial sensors and marking points, etc., Kinect and Leap Motion can extract and track hands that are completely unmarked or without additional sensors. This natural human-computer interaction has important research value. For example, in the film and television field, virtual human hands are used to complete dangerous actions; in the game field, users interact with virtual objects in virtual space by hand; in the industrial field, robots are controlled by human hands to conduct operations.

There are many methods based on Kinect gesture recognition, most of which use the depth information obtained by Kinect for processing and recognition. For example, (Yang *et al.*, 2012) proposed recognizing gestures based on depth information and hidden Markov model classifier; (Li, 2012) extracted hand contours,

calculated the set of convex and concave points and acquired all finger areas, and then implemented gesture recognition; (He *et al*., 2011) used depth information to estimate fingertips, and then recognize finger-level gestures; (Meng and Wang, 2013) first used the edge contour curvature feature method to locate fingertips and then obtained the motion vectors of the fingers to implement the function of fingertip gesture recognition. There are also many researches on gesture recognition based on Leap Motion. For example, (Mapari and Kharat, 2016) developed an Indian Sign Language recognition system that uses the Leap Motion sensor to recognize both hands. Reference (Staretu and Moldovan, 2016) used the Leap Motion equipment to control a personified plier's machine with five fingers. Reference (Chuan *et al*., 2015) based on Leap Motion, classified the 26-letter finger language in American Sign Language using the K-nearest algorithm and support vector machine. Reference (Erdoğan *et al*., 2016) recognizes gestured based on Leap Motion and artificial neural networks and controlled the robot through gestures. Reference (Tauchida *et al*., 2015) used the Leap Motion and SVM algorithms to realize the recognition of gesture trajectories. Reference (Chan *et al*., 2015) captured hand geometric data to identify gestures and authenticate identities.

However, Kinect and Leap Motion still have their own shortcomings. For example, Kinect does not have high recognition accuracy for fingers. When the finger points to Kinect, the details of the hand cannot be detected; although Leap Motion has high recognition accuracy, but its recognition space is very limited, and when the finger is blocked by other fingers, the recognition effect is not good. Few researches have been conducted on the combination of Kinect and Leap Motion. Reference (Craig and Krishnan, 2016) only fused the speed value for hand tracking. Each device has an independent trigger, which mitigates the impacts of blocking. Reference (Marin *et al*., 2015) introduced a gesture recognition method based on Leap Motion and Kinect, which acquires data through the two types of equipment and achieves gesture recognition in combination with the SVM classifier. Reference (Sreejith *et al*., 2015) used Kinect v2 and Leap Motion to implement an image navigation system, but this method is just a simple combination of Kinect and Leap Motion, where Kinect is used to recognize slightly distant gestures while Leap Motion to recognize close-range gestures. Reference (Debeir, 2014) fused the position data of the hand acquired by Leap Motion and Kinect sensors to improve the hand tracking performance.

The research content of this paper is to combine the advantages of Kinect v2 and Leap Motion. Kinect v2 has a large recognition space whereas Leap Motion has a high recognition precision. When the two are placed in different positions, the data observed from different angles are complementary and can be integrated into Unity3D to improve the accuracy of finger detection and make the interactions between human hands and virtual objects in virtual scenes more reliable. The rest of the paper is organized as follows: Section II describes the fingertip detection method based on Kinect v2 depth image, which extracts the hand area, obtains the hand contour and obtains the fingertip pixels according to the curve of the distance from the hand contour to the centre of the hand, and converts them to fingertip coordinates. Section III describes the calibration methods for Leap Motion and Kinect v2. The first step is to perform a preliminary calibration using a 1-dimensional calibration object, and then perform an accurate calibration and obtain a more accurate rotation matrix and translation vector through iterations. Section IV is about the data fusion, i.e. the fusion of the fingertip data of Leap Motion and Kinect v2 in Unity3D, including the temporal registration and spatial fusion. Section V conducts experiments on the method proposed in this paper and applies it in Unity3D. Section VI is a summary of and outlook on the research content of this paper.

2. Acquisition of fingertip data from depth images

2.1. Hand segmentation

This paper initially uses OpenCV and OpenNi to obtain the returned palm position, then quickly locates the hand region according to the vertical coordinate threshold of the palm and then uses the depth binary mask algorithm to separate the hand from the background. The algorithm can use the depth threshold to filter out the background with a similar colour to skin. For example, if the face overlaps with the hand, the face can be easily removed with the depth threshold. Note that this paper uses the original depth image of Kinect v2 (hereafter referred to as Kinect) for processing instead of the depth images obtained by OpenNi.

After the hand is detected by Kinect and OpenNi, the system will call the Nite library to return the coordinates of the palm. Let the y value of the palm coordinates returned be m, and then the upper and lower thresholds splitting the hand are respectively $\mathrm{m}+\Delta$ and $\mathrm{m}-\Delta$. According to the actual test, it is better if 7cm<Δ<14cm. According to the image area cut per the upper and lower thresholds, the hand can be accurately extracted.

After the hand region is extracted, the hand is separated from the background using a depth binary mask method. The mask is a template for image filter. In this paper, in order to extract the hand and remove the background, a matrix of $n * n$ is used to filter the pixels of the image so as to highlight the desired object. This matrix is called a mask, a binary image consisting of only 0 and 1. The binary mask P_d constructed in this paper is a mask window with given width and height and with the palm as the centre of the mask, which is defined by (1):

$$P_d(x,y) = \begin{cases} 1 & Z_h - d < Z(x,y) < Z_h + d \\ 0 & otherwise \end{cases} \quad (1)$$

After many experiments, the depth threshold $d = 8cm$ is obtained. Z_h is the coordinate depth value of the palm, Z(x,y) represents the depth value at the pixel (x,y) of the image. After binary mask, the hand image segmented is shown in Fig. 1:

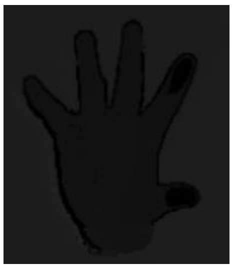


Figure 1. Depth image of a hand extracted

2.2. Depth image filter

The presence of noise and black holes in the depth image acquired by the Kinect sensor results in poor effects of target recognition and tracking (Bratoszewski and Czyżewski, 2015; Song *et al*., 2017). Therefore, this paper proposes a joint bilateral filter algorithm using the depth images and colour images acquired by the Kinect sensor. Bilateral filtering is an improved algorithm of Gaussian filtering. Compared with the latter, one of its most important feature is filtering while maintaining the edges of the image. Gaussian filtering is a linear filter that can effectively eliminate Gaussian noise, whose expression is shown in (2):

$$w_g(i,j) = \exp\left(-\frac{(i-x)^2 + (j-y)^2}{2\sigma_g^2} \right) \quad (2)$$

where, σ_g is the standard deviation of the Gaussian function; $w_g(i,j)$ represents the weight at the point (i,j). Gaussian filtering can only take into account the changes in the spatial distances of image pixels, but not those in the image pixel values. Bilateral filtering, on the other hand, pays attention to not only the spatial correlation between pixel values, but also the approximation between them. The filtering formula is as shown in (3):

$$w_s = w_g \times w_p \tag{3}$$

where:

$$w_p = \exp\left(-\frac{\left(I\left(i,j\right)+I\left(x-y\right)\right)^2}{2\sigma_p^2}\right) \tag{4}$$

In the above equation, w_g is the Gaussian filtering. The performance of the bilateral filter depends on the value of σ_p, which limits the relative positions of the pixels and the variation range of gray scale of the image. Compared with Gaussian filtering, bilateral filtering can preserve the edge information of the depth image and reduce the edge blurring caused by Gaussian filtering. The joint bilateral filter is proposed on the basis of the bilateral filter, of which the calculation formula is shown in (5):

$$F_p = \frac{1}{k_p}\sum_{q\in\Omega} I_q f\left(\left\|p-q\right\|\right) g\left(\left\|\bar{I}_p - \bar{I}_q\right\|\right) \tag{5}$$

where, I represent the input image, p and q the coordinates of the pixel in the image, I_p the pixel value at the position p in the image, F the output, and f and g the weight distribution functions, which are usually Gaussian functions. Joint bilateral filtering is to input another new image $\bar{I}$ in the weight calculation in the value domain, which must be similar to the image to be processed. The image of the hand obtained in the previous step is bilaterally filtered, and the binarization result is shown in Fig. 2(b):

Fig.2(a) is the binarized image without bilateral filtering. It is obvious that the contour and the corner points of the hand are clearer in Fig. 2(b), showing the image is enhanced. With Gaussian filtering in the joint bilateral filtering, Gaussian noise is suppressed, and the edges of the image are also smoothed, and with the introduction of a similar image in this filtering, the edge blur caused by Gaussian filtering is prevented.

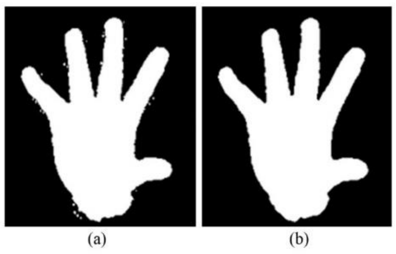


Figure 2. Comparison of the implementation results

2.3. Calculation of fingertip coordinates

The Canny operator is used to extract the outline of the hand region. After the distance between the centre of the hand and each pixel on the contour of the hand is calculated, the distance curve is obtained, as shown in Figure 3.

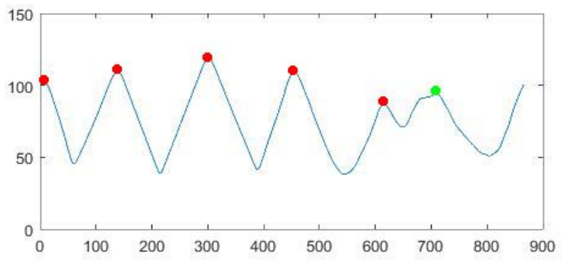


Figure 3. Curve of the distance between the centre and the contour of the hand

The coordinates corresponding to most peak points of the distance curve are located above the centre of the hand, which are regarded as fingers, like the Pink, Ring, Middle, and Index shown in Fig. 4. The coordinates corresponding to a few peak points are below the centre of the hand, which are subject to judgment. If the coordinate points are closer to the y-axis of the centre of the hand, they are regarded as fingertip coordinates, like the Thumb shown in Fig. 4, whereas those away from the y-axis of the centre of the hand are not considered as fingertip coordinates, like the green point shown in Figure 4.

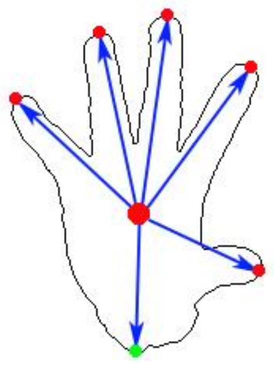


Figure 4. Determination of fingertips

After the pixel positions of the fingertips are determined, they are converted to the corresponding three-dimensional space coordinates using Kinect SDK, which will then be used for data fusion of the two somatosensory devices.

3. Joint calibration of Kinect and leap motion

The coordinate systems of the two sensors are integrated into one by calibration. In this paper, the coordinate system of Leap Motion is the main coordinate system, and the coordinate system of Kinect should be rotated and translated to that of Leap Motion. Let the rotation matrix and translation vector be respectively R and T and let the coordinate of one point be Y in the Kinect coordinate system and X in the Leap Motion coordinate system, and then,

$$X = \mathbf{R}Y + \mathbf{T} \tag{6}$$

where, R is the 3×3 matrix, with 9 unknown numbers; and T is a 3×1 vector, with 3 unknown numbers.

Zhang's calibration method is used for preliminary calibration, where three reflective balls on one pole are used as the calibration points. The three reflective balls are simultaneously observed by Kinect and Leap Motion, one of which is fixed, and the other two changed in directions around this calibration point to generate 6 calibration maps and solve 12 unknowns' numbers of R and T, and from this, the preliminary calibration results are obtained. The specific method is detailed in (Zhang, 2004).

After preliminary calibration, accurate calibration is performed. Inspired by the ICP algorithm (Besl and Mckay, 1992), this paper iterates on the preliminary calibration results. The steps are as follows:

(1) The coordinates of the two types of five fingertips are taken as the initial positions of the point set. The point set *KP* is the set of the coordinates of the fingertips transformed through (5) from those in the Kinect system on the basis of the preliminarily calibrated R and T, which is expressed as:

$$\boldsymbol{KP} = \left\{ p_i \mid p_i \in \boldsymbol{R}^3, i = 1, 2, \dots m \right\} \tag{7}$$

where, $m = 5$. The point set *LP* consists of the coordinates of the fingertips in Leap Motion, denoted as:

$$\boldsymbol{LP} = \left\{ q_i \mid q_i \in \boldsymbol{R}^3, i = 1, 2, \dots n \right\} \tag{8}$$

where, $n = 5$

(2) Calculate the point q_i corresponding to the fingertip point $P_i \in$ KP, to make

$$\| p_i \text{-} q_i \| \to \min \tag{9}$$

(3) Recalculate R and T to make

$$\sum \| \mathbf{R}p_i + \mathbf{T}\text{-}q_i \| \rightarrow \min \tag{10}$$

(4) Use the recalculated R and T to transform *KP* to obtain the new point set;

(5) If the least square error is smaller than the threshold ε, the iteration stops. In this paper, $\varepsilon = 0.0000005$.

Through iterations, R and T are more accurate.

4. Data fusion

4.1. Integration of leap motion, Kinect and unity3D

The Kinect for Windows SDK cannot be directly applied to the Unity3D platform, so a middleware called Kinect v2 with MS-SDK20.unitypackage is required, through which, Unity 3D can obtain the data collected by Kinect in real time, so that the data can be used together with the trigger preset by Unity3D for development of application software.

The technical standards for Leap Motion are to use the right-hand coordinate system and process data in millimeters (mm); whereas the technical standards for Unity3D are to use left-handed coordinate system and process data in meters (m). This paper uses the middleware Leap Motion Core Assets 4.3.4. unity package to convert the technical standards for Leap Motion technical standards into those of Unity.

This paper uses Unity 5.6.4p2 (64-bit) to implement human-computer interaction application in virtual environment. Leap Motion and Kinect are connected to the same PC.

4.2. Temporal registration of data

Set the acquisition frequency of Leap Motion to 30fps in Unity 3D. This frequency is consistent with the acquisition frequency of Kinect, so that every time after the same period, the two devices output one frame of data at the same time to achieve temporal registration.

4.3. Spatial fusion of data

The data collected by the two devices are registered in the virtual space of Unity3D. The five fingertip coordinate data collected by Kinect (see Section II) are (KX_i, KY_i, KZ_i) , and the five-fingertip data collected by Leap Motion are (LX_i, LY_i, LZ_i). The main steps are as follows:

(1) The transformation matrix R and the translation matrix T obtained in Section III transform (KX_i, KY_i, KZ_i) into the Leap Motion coordinate system;

(2) Transform (LX_i, LY_i, LZ_i) and (KX_i, KY_i, KZ_i) to the Unity3D coordinate system;

(3) There are 6 parameters $(LX_i, LY_i, LZ_i, KX_i, KY_i, KZ_i)$ for the coordinates of each fingertip. In this case, set the fingertip coordinates under Unity3D to:

$$(X_i, Y_i, Z_i) = \lambda_1 (LX_i, LY_i, LZ_i) + \lambda_2 (KX_i, KY_i, KZ_i) \tag{11}$$

where, $\lambda_1 + \lambda_2 = 1$. When the fingertip data obtained by Leap Motion fail or remain the same, $\lambda_2 = 1$; when the fingertip data obtained by Kinect fail, $\lambda_1 = 1$; and when the five-fingertip data of the two can all be detected, $\lambda_1 = 0.6, \lambda_2 = 0.4$.

5. Experiments and results

5.1. Experimental equipment

Hardware Environment: Computer (Intel Xeon(R), CPU E5-2650 32G memory, Nvidia Quadro K5000 GPU, 2 monitors), Kinect v2 for Windows, Leap Motion; Software Environment: 64-bit Windows 10, Visual Studio.net 2012, Kinect SDK 2.0, Leap Developer Kit 3.2.1, Unity 5.6.4p2.

5.2. Fingertip detection experiment based on Kinect depth image

This paper uses Kinect to process 15 gestures, each of which is acquired 50 times. Through the experiment, it can be seen that the Kinect-based fingertip detection method proposed in this paper can detect the fingertips of most gestures and it is very robust, as even under complex backgrounds, it can extract the hand region according to the depth relationship between the hand and other parts of the body. However, when the fingers are put together, the fingertips are difficult to detect accurately. See gesture 2, 3 and 5 in Line 3, Fig. 5. In addition, since the method detects fingertips based on the distance between the centre of the hand and the contour of the hand, the pixel corresponding to the peak of the distance curve is not necessarily the fingertip, like gesture 4 in Line 2, Fig. 5, where the distance from the point to the centre of the hand is greater than that between the fingertip and the centre of the hand, but under this method, it is recognized as a fingertip. In the future work, further studies need to be conducted on the fingertip detection method.

According to the fingertip pixel position obtained by the above fingertip detection method, the three-dimensional coordinates corresponding to the pixel are then obtained using the Kinect SDK.

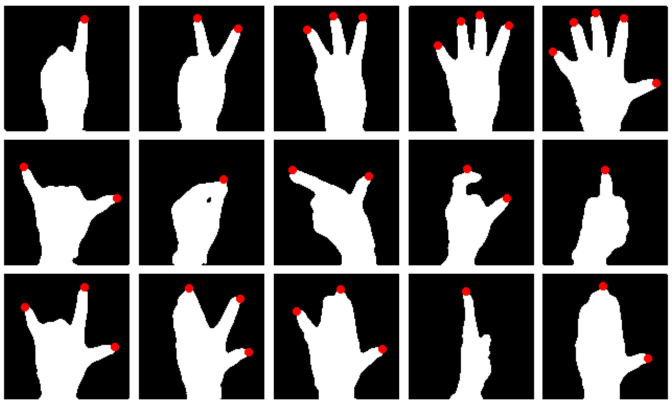
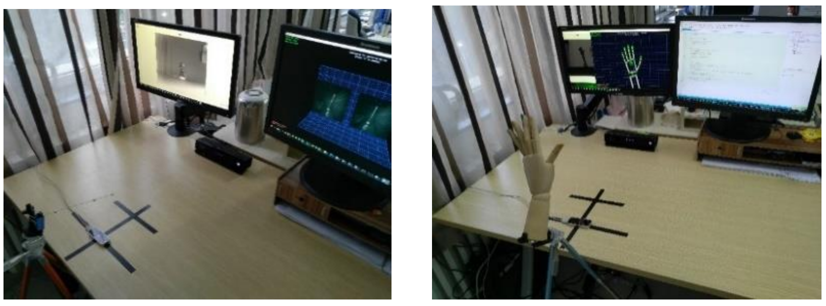


Figure 5. Effect diagram of fingertip positioning

5.3. Joint calibration experiment and analysis

Kinect and Leap Motion are placed at a distance of about 0.7 meters. The two devices are connected to the same PC and a one-dimensional calibration rod is placed within the detection range of the two devices, as shown in in Fig. 6(a).



(a) (b)

Figure 6. Calibration experiment

5.4. Preliminary calibration

In the preliminary calibration, a reflective ball is fixed at a great distance from Kinect is changed in directions for 6 times, and Kinect and Leap Motion simultaneously take the images of the calibration rod, and then find the pixel coordinates of the ball. The resulting images are shown in Figure 7. Figure 7(a) shows the reflective ball taken by Kinect, and Figure 7(b) shows the one taken by Leap Motion.

Through preliminary calibration, the internal parameter calibration results of the two devices are obtained, as shown in

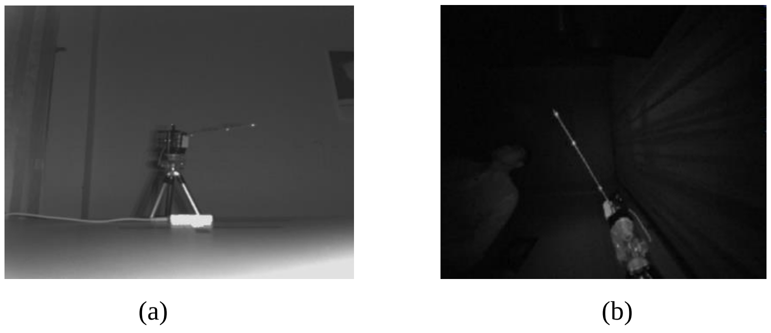


Figure 7. The reflective ball is photographed by the equipment

Table 1, where α and β represent the focal lengths of the devices, u_0 and v_0 the positions of the main points of the devices and γ indicates the tilting parameter of the coordinate axis.

Table 1. Internal parameters of the two sensors

Internal parameter	α (mm)	β (mm)	γ	u_0 (pixel)	v_0 (pixel)
Kinect v2	365.33	365.67	0.3026	262.4536	208.3791
Leap Motion	108.9845	54.5032	1.9314	321.0786	125.1219

5.5. Fine calibration

The device is replaced with a hand model with a fixed gesture, as shown in Figure 6(b).

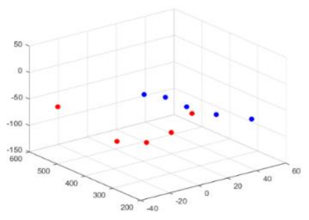
Leap Motion is able to obtain the fingertip coordinates via its SDK, whereas Kinect obtains them using the methods in Section II. In order to increase the robustness of fine calibration and avoid the instability of the data collected by the two devices, the experiment uses a hand model with a fixed gesture and adopts the fine calibration method in Section III to obtain a more accurate transformation matrix and translation vector.

5.6. Calibration results and analysis

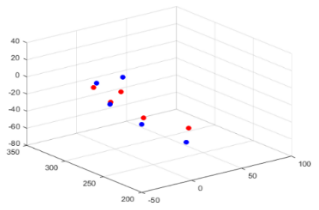
The registration results of the fingertips are shown in Fig. 8. Fig. 8(a) shows the fingertip data before calibration, where the red dots are the fingertip data obtained by

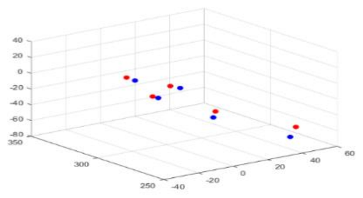
Kinect, and the blue ones are those obtained by Leap Motion. Through transformation, the Kinect coordinate system is changed to

the Leap Motion coordinate system. Fig.8(b) shows the results after preliminary calibration and Fig. 8(c) the results after fine calibration.



(a)

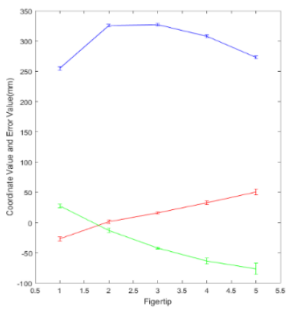


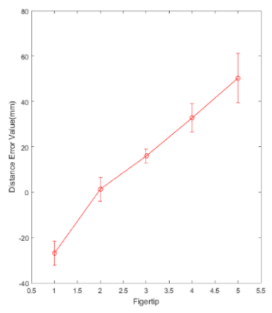


(b) (c)

Figure 8. Calibration results of fingertip data

After being transformed, the coordinates of the fingertips acquired by Kinect are compared with those obtained by Leap Motion in terms of coordinate value errors on the X-axis, Y-axis and Z-axis and the distance between the two types of fingertips as shown in Figure 9.





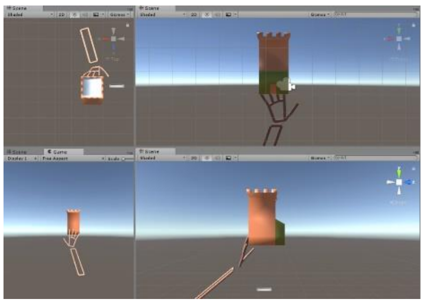
(a) (b)

Figure 9. Error distribution map of the two types of fingertip coordinates

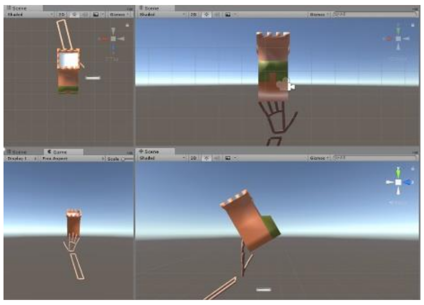
Figure 9(a) shows the errors of the two types of fingertip coordinates on the coordinate axes, wherein the red, blue, and green error lines respectively indicate the coordinate value errors on the X-axis, Y-axis, and Z-axis; and Figure 9(b) indicates the errors in the distance between the two types of fingertips. It can be seen that the errors of the two coordinates on the coordinate axes do not exceed 20 mm, and that the error in the coordinate distance is at most 22 mm, indicating that the calibration method proposed in this paper can spatially align the coordinates obtained by the two sensors.

5.7. Human-computer interaction application test

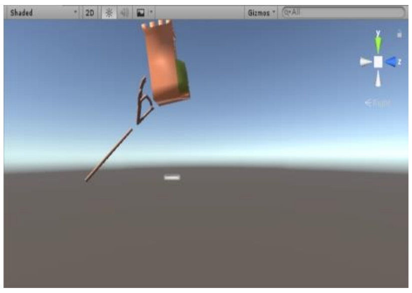
After the calibration and data fusion are completed, the hand can interact with the object in the virtual scene, which is a small version of castle. When a finger touches it, the castle will change its direction, as shown in Fig. 10. Fig. 10(a) shows the state of the virtual object observed from four angles before it is touched; (b) shows that after it is touched; (c) shows that, when the hand is out of the detection range of Leap Motion, per the fingertip data detected by Kinect, some fingers are incorrectly recognized as bent ones, but that the object can change its direction; (d) shows that when some fingers are bent, Kinect fails to detect the fingertips, but that Leap Motion's fingertip data alone can still be used to touch the object accurately.



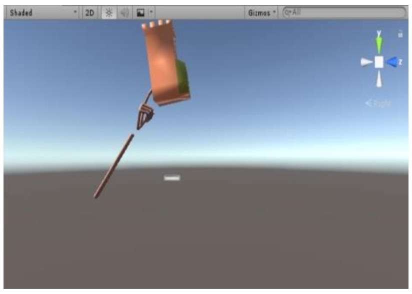
(a)



(b)



(c)



(d)

Figure 10. Results of touching the virtual objects

Experiments show that the data fusion of the two sensors expands the recognition range of the hand. When one sensor fails, the other sensor can still recognize the gesture, which reduces the impacts from hand joints blocking each other, enhances the robustness of the method and improves the recognition rate of the gesture.

The overall scene graph of the system is shown in Figure 11.

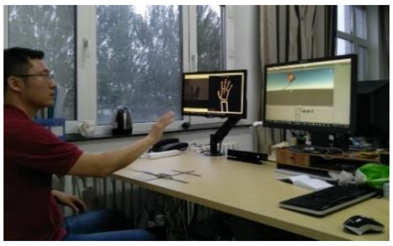


Figure 11. Overall scene of the system

5.8. Comparison with other methods

Table 2. Recognition rate comparison (%)

Gesture	Ours Method Only Kinect	Ours Method	Paper (Marin *et al.*, 2015)
1	94	96	96
2	96	98	94
3	96	96	86
4	98	98	-
5	100	100	97
6	92	94	90
7	10	90	-
8	94	92	91
9	80	80	-
10	90	90	-
11	88	90	86
12	60	90	-
13	62	88	-
14	10	82	-
15	8	84	-

The methods in (Marin *et al*., 2015; Sreejith *et al*., 2015; Debeir, 2014; Bratoszewski and Czyżewski, 2015) all used 1 Kinect and 1 Leap Motion, but only (Marin *et al*., 2015) recognized the gesture, and there was no touching the virtual object in Unity3D. Here the gestures in Unity3D under this method (the results of fusing the gestures recognized by Kinect and the Leap Motion data) are compared with the related gestures under (Marin *et al*., 2015) (using Kinect v1). Each gesture is tested for 50 times with the left hand, as shown in Table 2, 1-15 in the first column represent the gestures in Figure 5.

As shown in the table, there are great errors with respect to gesture 7, 12, 13, 14 and 15 if Kinect is used alone. After it is combined with Leap Motion, the recognition rate is greatly improved.

6. Conclusion and outlook

In order to meet the basic requirements of the natural human-computer interaction system, this paper integrates Leap Motion, Kinect v2 and Unity3D together and proposes a hybrid method consisting of multiple methods. The first is the fingertip detection method based on Kinect v2 depth image; the second is a two-step joint calibration method, including preliminary calibration and fine calibration, of which, the former uses the existing one-dimensional calibration method, and the latter uses the iterative method to obtain the transformation matrix and the translation matrix; and the third is the fusion of data. After the data of the two somatosensory devices are fused in space and time and also weighted, even if the joint of the hand is blocked in front of one device and the data are invalid, the data of the other device can still be used to recognize gestures, which improves the robustness of the system and helps achieve good results in the experiment.

However, this method still has some shortcomings. For example, when a finger is bent, the fingertip recognition rate of Kinect v2 decreases, causing failure of interaction with the virtual object; in addition, when the palm is perpendicular to the Leap Motion and points to Kinect v2, due to the blocking of fingers, the system cannot identify the fingertip position well. In future work, it is suggested adding one more Leap Motion or Kinect v2 so that the gestures can be observed from more angles and the recognition rates of fingers and gestures can be improved.

Acknowledgments

The authors feel like showing the sincerest as well as grandest gratitude to the Changchun University of Science and Technology, National and Local Joint Engineering Research Centre of Special Film Technology and Equipment for the support on equipment and technical support from members.

Reference

Besl P. J., Mckay N. D. (1992). A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence - Special issue on interpretation of 3-D* *scenes—part II*, Vol. 14, No. 2, pp. 239-256. http://doi.org/10.1109/34.121791

Bratoszewski P., Czyżewski A. (2015). Face profile view retrieval using time of flight camera image analysis. i*n Pattern Recognition and Machine Intelligence: 6th International Conference, PReMI 2015, Warsaw, Poland, Publisher: Springer*, pp. 159-168. http://doi.org/10.1007/978-3-319-19941-2_16

Chan A., Halevi T., Memon N. (2015). Leap motion controller for authentication via hand geometry and gestures, human aspects of information security, privacy, and trust. *Springer International Publishing*, pp. 13-22. http://doi.org/10.1007/978-3-319-20376-8_2

Chuan C. H., Regina E., Guardino C. (2014). American sign language recognition using leap motion sensor. *in 2015 International Conference on Machine Learning and Applications IEEE*, Vol. 13, pp. 541-544. http://doi.org/10.1109/ICMLA.2014.110

Craig A., Krishnan S. (2016). Fusion of leap motion and kinect sensors for improved field of view and accuracy for VR applications (course report). *Stanford University, unpublished*.

Erdoğan K., Durdu A., Yilmaz N. (2016). Intention recognition using leap motion controller and artificial neural networks. *International Conference on Control, Decision and Information Technologies IEEE*, pp. 689-693. http://doi.org/10.1109/CoDIT.2016.7593646

He G. F., Kang S. K., Song W. C., Jung S. T. (2011). Real-time gesture recognition using 3D depth camera. *International Conference on Software Engineering and Service Science IEEE*, pp. 187-190. http://doi.org/10.1109/ICSESS.2011.5982286

Li Y. (2012). Hand gesture recognition using Kinect. *IEEE 3rd International Conference on Software Engineering and Service Science (ICSESS)*, pp. 196–199.

Mapari R. B., Kharat G. (2015). Real time human pose recognition using leap motion sensor. *in 2016 IEEE International Conference on Research in Computational Intelligence and Communication Networks IEEE*, pp. 323-328. http://doi.org/10.1109/ICRCICN.2015.7434258

Marin G., Dominio F., Zanuttigh P. (2015). Hand gesture recognition with leap motion and kinect devices. *IEEE International Conference on Image Processing IEEE*, pp. 1565-1569. http://doi.org/10.1109/ICIP.2014.7025313

Meng G., Wang M. (2013). Hand gesture recognition based on fingertip detection. *Global Congress on Intelligent Systems IEEE Computer Society* pp. 107-111. http://doi.org/10.1109/GCIS.2013.23

Penelle B., Debeir O. (2014). Multi-sensor data fusion for hand tracking using Kinect and leap motion. *in Virtual Reality International Conference ACM, Laval, France, 2014*, pp. 22. http://doi.org/10.1145/2617841.2620710

Song X., Huang H., Zhong F., Ma X., Qin X. (2017). Edge-guided depth map enhancement. *International Conference on Pattern Recognition, IEEE*, http://doi.org/10.1109/ICPR.2016.7900053

Sreejith M., Rakesh S., Gupta S., Biswas S., Das P. P. (2015). Real-time hands-free immersive image navigation system using Microsoft Kinect 2.0 and Leap Motion Controller. *Computer Vision, Pattern Recognition, Image Processing and Graphics IEEE*, pp. 1-4. http://doi.org/ 10.1109/NCVPRIPG.2015.7489999

Staretu I., Moldovan C. (2016). Leap motion device used to control a real anthropomorphic gripper. *International Journal of Advanced Robotic Systems*, Vol. 13.

Tsuchida K., Miyao H., Maruyama M. (2015). Handwritten character recognition in the air by using leap motion controller. HCI International 2015 - Posters' Extended Abstracts. *Springer International Publishing*, pp. 534-538, 2015. http://doi.org/10.1007/978-3-319-21380-4_91

Yang C., Jang Y., Beh J., Han D. (2012). Gesture recognition using depth-based hand tracking for contactless controller application. *IEEE International Conference on Consumer Electronics IEEE*, pp. 297-298. http://doi.org/10.1109/ICCE.2012.6161876

Zhang Z. Y. (2004). Camera calibration with one-dimensional objects. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, Vol. 26, No. 7, pp. 892-899, http://doi.org/10.1109/TPAMI.2004.1304991