ICAT '99

Simple 3-D Mouse with Auto-error-correction Function

Masamichi HOSODA

Business Communications Department,
NTT East Saitama Branch
Arche 7F, 2-1-1 Sakuragi-cho, Omiya-shi,
Saitama 331-0852 JAPAN
m-hosoda@bc.saitama.east.ntt.co.jp

Yukio KOBAYASHI

Department of Electronics,
Chiba Institute of Technology
2-17-1 Tsudanuma, Narashino-shi,
Chiba 275-0016 JAPAN
koba@es.it-chiba.ac.jp

Abstract

3-D mice are used for drawing 3-D computer graphics in the field of Virtual Reality. They are far more expensive and complicated than usual 2-D mice. This paper proposes a low-cost, simple 3-D mouse. It uses a part of the 2-D mouse's mechanism for low-cost. Its software automatically corrects the coordinate errors caused by hardware. Its performance is shown through experiments, comparing with other 3-D mice.

Key words: 3-D mouse, auto-error-correction, algorithm, stereoscopic vision, modeler

1. Introduction

A modeler using the stereoscopic vision and 3-D mouse is proposed for easy drawing of 3-D computer graphics [1][2]. Polhemus ISOTRACK II [3] was used as a 3-D mouse in the study.

The device uses magnetic sensors, and can measure 3-D positions and angles without physical devices such as strings and linkages. But, measured coordinates are distorted, when it is used near the computer display, where the magnetic field falls into disorder.

This paper proposes a low-cost and simple structural 3-D mouse, which uses a part of the 2-D mouse's mechanism.

A three-sensors version 3-D mouse [2] and a four-sensors version 3-D mouse [4] are discussed in this paper. The former mouse is simpler than latter one. However, it accumulates the positioning error, and needs frequent calibrations. The latter mouse has the automatic-error-correction function, and requires no calibration.

2. Simple 3-D Mouse

Several studies have been made on the 3-D pointing device [5]. However, they are far more complex and expensive than usual 2-D mice.

Our simple 3-D mice calculate the 3-D coordinates by measuring the lengths of tensed strings (fishing lines) attached to the pointer.

Sato's SPIDAR [6][7][8] which can obtain 3-D coordinates also uses the physical strings. However, its main purpose is to give users sense of feedback forces by fixing and pulling the strings, and the mechanism is large-scaled. The purpose of our 3-D mice is to move the cursor in the screen as well as usual 2-D mice.

Hirata investigated the calculation method of 3-D coordinates used for SPIDAR [7]. In the study, four strings are used and if the length of one string is wrong, correct coordinates can be obtained using the other 3 strings. However, it cannot detect which length is wrong. When two or more lengths are wrong, correct coordinates cannot be obtained. If the initial lengths are not given, coordinates cannot be calculated. SPIDAR needs to make calibrations beforehand to use it. 4-sensors version 3-D mouse of this paper provides a strong automatic error correction function. Therefore, even if all lengths are wrong, almost correct coordinates can be calculated, and the calibration is unnecessary.

An ultrasonic sensor is a typical solution for 3-D mice. It measures the distance by the ultrasonic wave, and can obtain 3-D coordinates. However, its spread speed changes depending on the temperature. Moreover, it interferes mutually with near other sources such as the same kind of 3-D mice. The 3-D mouse of this paper is using the string, and does not have these problems.

3. 3-sensors version 3-D Mouse

The outline of 3-sensors version 3-D mouse, which is the basic type of the simple 3-D mice, is described.

3.1 Mechanisms and Structure

To know coordinates of point $P(x, y, z)$ in the 3-D space, the equation (1) is solved, where l_1, l_2 and l_3 are distances from three already-known points $P_1(x_1, y_1, z_1)$,

$P_2(x_2, y_2, z_2)$ and $P_3(x_3, y_3, z_3)$ to point P.

$$\begin{cases} (x-x_1)^2 + (y-y_1)^2 + (z-z_1)^2 = l_1^2 \\ (x-x_2)^2 + (y-y_2)^2 + (z-z_2)^2 = l_2^2 \\ (x-x_3)^2 + (y-y_3)^2 + (z-z_3)^2 = l_3^2 \end{cases} \qquad (1)$$

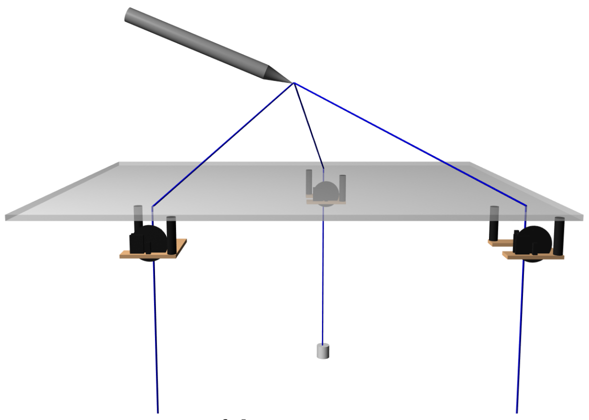


Fig. 1 Structure of the 3-sensors version 3-D mouse

Figure 1 shows the structure of the proposed 3-sensors version 3-D mouse. Three rotation sensors are attached under the board and three pinholes are made near the sensors. The pinholes form a equilateral triangle with 20cm long sides. These are already known points. Three strings are tied to the mouse pointer at an unknown point. The strings from the pointer go through the pinholes, are wound around the shafts of the rotation sensors and are connected to hanging weights at their ends.

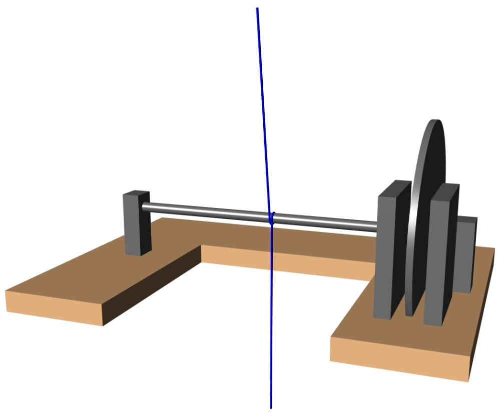


Fig. 2 Rotation Sensor

The rotation sensors shown in Figure 2 are usually used for 2-D mice to detect rotation of the mouse ball. The disk at the right side of the Figure has slits, and a photo-coupler beside the disk detects the cycle and direction of the disk rotation. In the proposed 3-D mouse, the rotation sensors detect the movements of the strings. The lengths of the strings can be obtained from the cycles and directions of the rotations if the initial lengths of the strings are known.

3.2 Coordinate System

Figure 3 shows the coordinate system of the 3-sensors version 3-D mouse. Coordinates of each pinhole are assumed to be $P_1(0,0,0)$, $P_2(1,0,0)$, and $P_3\left(\frac{1}{2}, \frac{\sqrt{3}}{2}, 0\right)$.

The unknown pointer coordinates $P(x_{123}, y_{123}, z_{123})$ can be calculated from the equations (2), (3) and (4), these are derived from the equation (1) by substituting the pinholes coordinates.

$$x_{123} = \frac{1}{2}\left(1 + l_1^2 - l_2^2\right) \qquad (2)$$

$$y_{123} = \frac{\sqrt{3}}{6}\left(1 + l_1^2 + l_2^2 - 2l_3^2\right) \qquad (3)$$

$$z_{123} = \frac{1}{3}\sqrt{3r} \qquad (4)$$

$$\left(\begin{array}{l} r = l_1^2 l_2^2 + l_2^2 l_3^2 + l_3^2 l_1^2 \\ -l_1^4 - l_2^4 - l_3^4 + l_1^2 + l_2^2 + l_3^2 - 1 \end{array}\right)$$

There are two solutions for the z coordinate because of quadratic equation. Since the pointer is able to move only over the upper side of the board, the positive value is adopted.

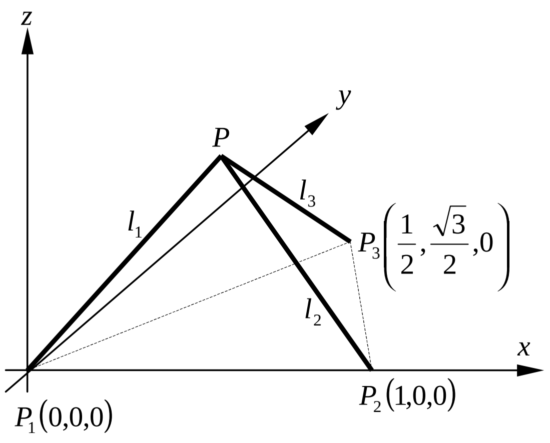


Fig. 3 Coordinate System of the 3-sensors 3-D mouse

3.3 Calibration

The pointing position is calculated from the data of the lengths. The 3-sensors 3-D mouse requires initial calibration to obtain the initial lengths of the strings, as the rotation sensors output cycles and directions which give only increment of the string lengths. This mouse needs frequent calibrations, because the string slips on the shafts of the rotation sensor bring about errors in the measured lengths.

3.4 Problems

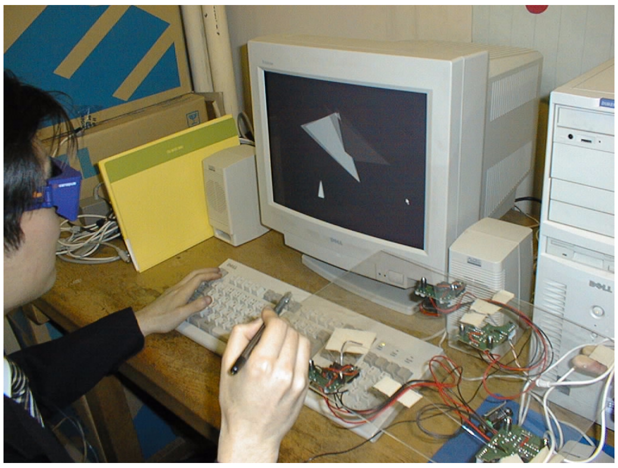


Fig. 4 3-sensors version 3-D Mouse

Figure 4 shows the 3-sensors version 3-D mouse. The coordinate error was 1 to 2 mm immediately after a calibration. After five minutes usage, the error became about 2.5 cm. This amount of error has no problem, because the pointer is handled, while seeing the movement of the cursor on the screen, like in the case of the 2-D mouse. It is enough for the pointer operation to know the relative amount of the movement. If the error is small, it is inconsiderable.

However, the error will accumulate in longer use, and make the coordinate distorted to the extent that the movement of the pointer is not proportional to the movement of the cursor, regarding direction and distance. Therefore, it is necessary to conduct frequently calibrations.

3-sensors version 3-D mouse has another problem in operation. The weight of the mouse pointer must be hold by an arm of the operator who is easily tired.

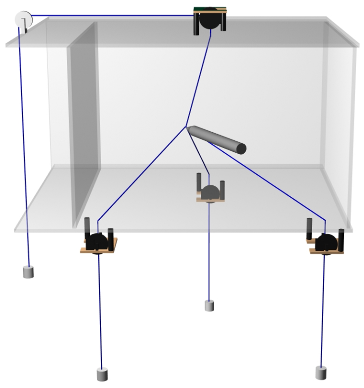


Fig. 5 Structure of the 4-sensors version 3-D mouse

4. 4-sensors version 3-D Mouse

To solve the 3-sensors version 3-D mouse's problems, the 4-sensors version 3-D mouse is proposed.

4.1 Mechanisms and Structures

Figure 5 shows the structure. The forth rotation sensor is attached to the second board above the pointer. The string is installed from the pointer to a hanging weight through a pinhole and on a pulley on the second board, winding around the shaft of the forth sensor. The forth pinhole and the other 3 pinholes form a tetrahedron with 20cm long sides.

The equation for the 4-sensors 3-D mouse is shown as follows.

$$\begin{cases} (x-x_1)^2+(y-y_1)^2+(z-z_1)^2=l_1{}^2 \\ (x-x_2)^2+(y-y_2)^2+(z-z_2)^2=l_2{}^2 \\ (x-x_3)^2+(y-y_3)^2+(z-z_3)^2=l_3{}^2 \\ (x-x_4)^2+(y-y_4)^2+(z-z_4)^2=l_4{}^2 \end{cases} \tag{5}$$

The linear equation (6) is obtained from equation (5).

$$\begin{cases} A_{11}x+A_{12}y+A_{13}z+A_{14}=l_1{}^2-l_2{}^2 \\ A_{21}x+A_{22}y+A_{23}z+A_{24}=l_2{}^2-l_3{}^2 \\ A_{31}x+A_{32}y+A_{33}z+A_{34}=l_3{}^2-l_4{}^2 \end{cases} \tag{6}$$

$$\left(\begin{aligned} A_{11}&=2(x_2-x_1) \\ A_{12}&=2(y_2-y_1) \\ A_{13}&=2(z_2-z_1) \\ A_{14}&=x_1{}^2-x_2{}^2+y_1-y_2{}^2+z_1-z_2{}^2 \\ A_{21}&=2(x_3-x_2) \\ A_{22}&=2(y_3-y_2) \\ A_{23}&=2(z_3-z_2) \\ A_{24}&=x_2{}^2-x_3{}^2+y_2-y_3{}^2+z_2-z_3{}^2 \\ A_{31}&=2(x_4-x_3) \\ A_{32}&=2(y_4-y_3) \\ A_{33}&=2(z_4-z_3) \\ A_{34}&=x_3{}^2-x_4{}^2+y_3-y_4{}^2+z_3-z_4{}^2 \end{aligned}\right)$$

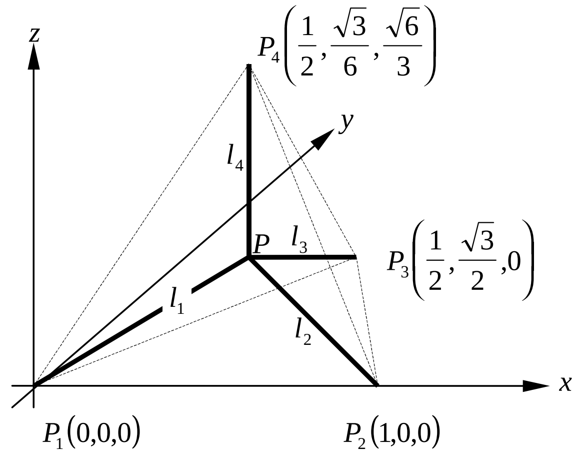
4.2 Coordinate System

Figure 6 shows the coordinate system of the 4-sensors version 3-D mouse. Coordinates of the each pinhole are assumed to be

$P_1(0,0,0)$, $P_2(1,0,0)$, $P_3\left(\frac{1}{2},\frac{\sqrt{3}}{2},0\right)$

and $P_4\left(\frac{1}{2},\frac{\sqrt{3}}{6},\frac{\sqrt{6}}{3}\right)$.

The pointer coordinates $P(x_{1234},y_{1234},z_{1234})$ can be calculated from the equation (7), (8) and (9), these are obtained from the equation (6) by substituting the pinhole coordinates.



z
$P_4\left(\frac{1}{2},\frac{\sqrt{3}}{6},\frac{\sqrt{6}}{3}\right)$
y
l_4
P
l_3
$P_3\left(\frac{1}{2},\frac{\sqrt{3}}{2},0\right)$
l_1
l_2
x
$P_1(0,0,0)$
$P_2(1,0,0)$

Fig. 6 Coordinate System of the 4-sensors 3-D mouse

$$x_{1234}=\frac{1}{2}\left(1+l_1{}^2-l_2{}^2\right) \qquad (7)$$

$$y_{1234}=\frac{\sqrt{3}}{6}\left(1+l_1{}^2+l_2{}^2-2l_3{}^2\right) \qquad (8)$$

$$z_{1234}=\frac{\sqrt{6}}{12}\left(1+l_1{}^2+l_2{}^2+l_3{}^2-3l_4{}^2\right) (9)$$

The solutions use the data from the fourth sensor. The coordinates can also be calculated from the equation (2), (3) and (4) without using the data from the fourth sensor. The equation (2) and (7) are the same. The equation (3) and (8) are also the same. The difference is the equation (4) and the equation (9) for the z coordinate.

4.3 Error Detect

This simple 3-D mouse has slips of the strings on the shafts of the rotation sensors. As a result, real lengths of the strings are different from calculated ones. This is the main reason why the coordinate errors occur.

The two sets of the equations can be used for the 4-sensors version 3-D mouse to calculate the same coordinates as mentioned in the previous section. The two sets of the calculation are different if there is a sensor, which outputs an erred length information.

The 4-sensors version 3-D mouse of this paper can calculate the z coordinates by the two equations (the equation (4) that uses the three sensors and the equation (9) that uses the four sensors).

Expression (10) is derived from these two equations.

$$d=\left|z_{123}-z_{1234}\right| \qquad (10)$$

This expression does not represent the true error. However, the value of this expression becomes 0 if there is no error. Therefore, it can be used as an evaluation function of the error.

4.4 Auto-error-collection function

Auto-error-collection algorithms are proposed.

4.4.1 Normalization

This algorithm is to re-calculate string lengths, based on the output coordinates, when the value of expression (10) exceeds a threshold. This algorithm is called "Normalization".

Coordinates calculated from the equations (7), (8) and (9) are used for the re-calculated string lengths. The lengths of the strings are replaced by equation (11) as a result.

$$l_n=\sqrt{\left(x_{1234}-x_n\right)^2+\left(y_{1234}-y_n\right)^2+\left(z_{1234}-z_n\right)^2} \qquad (11)$$
$$(n=1,2,3,4)$$

When Normalization is applied, the value of expression (10) becomes 0. However, when the pointer is moved, the value of expression (10) increases again if the error still remains in coordinates. At this time, the value of expression (10) is almost directly proportional to the coordinate error. It is also almost directly proportional to the distance of the pointer movement from the position where previous Normalization was applied.

When the coordinate error is larger than the threshold, Normalization is applied once again, even if the pointer moves only short distance. Repeating Normalization several times, the error becomes smaller. When the coordinate error is small, long distance movement of the pointer is forgiven for next Normalization. Consequently, the error stays in a small value.

In the case that the coordinate error unfortunately increases after Normalization is applied, Normalization will be applied one after another in a short time until the error decreases. As the results, the coordinate error decreases in a short time.

The threshold is assumed to be 0.05(1cm) for the 4-sensors version 3-D mouse.

4.4.2 Limiter

This algorithm is to limit output coordinates within a range. If the calculated coordinates go beyond the range, it is reset to the range limitation. This algorithm called "Limiter".

Results of Normalization process are converged when the coordinate error is small. However, when the coordinate error is large, the pointer coordinates become unstable, jumping to far positions in Normalization process, and there is a possibility of divergence.

The range is assumed to be on a circumscribed hexahedron to the regular tetrahedron, which consists of the four pinholes for the 4-sensors version 3-D mouse.

4.5 Comparison of Algorithms

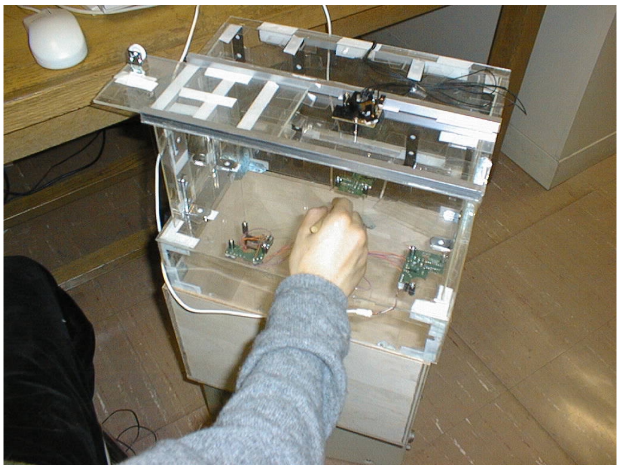


Fig. 7 4-sensors version 3-D mouse

Figure 7 shows the 4-sensors version 3-D mouse. The outputs of the rotation sensors were recorded directly by actually using the 3-D mouse for the comparative study of each algorithm. Each algorithm was evaluated by simulation based on the record.

The algorithms used for the comparison are as follows.

3 sensors:
The coordinate values are calculated
by equations (2) (3) (4).

4 sensors:
The coordinate values are calculated
by equations (7) (8) (9).

N. 4 sensors:
4 sensors with Normalization.

NL. 4 sensors:
4 sensors with Normalization and Limiter.

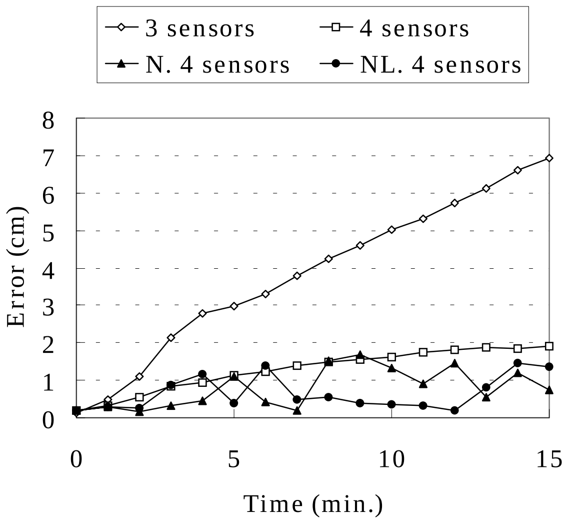


Fig. 8 3-D Error (with initial calibration)

4.5.1 Simulation with Initial Calibration

Figure 8 shows the simulation result with initial calibration.

The error accumulates in the “3 sensors” algorithm. The error accumulates little by little in the “4 sensors” algorithm. Accumulation is not seen in the “N. 4 sensors” and “NL. 4 sensors” algorithms.

4.5.2 Simulation without Initial Calibration

4.5.2.1 Small Initial Error

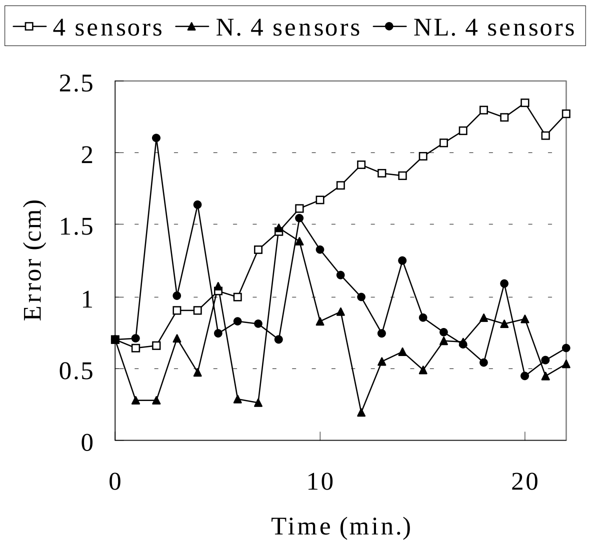


Fig. 9 3-D Error
(without calibration, initial error small)

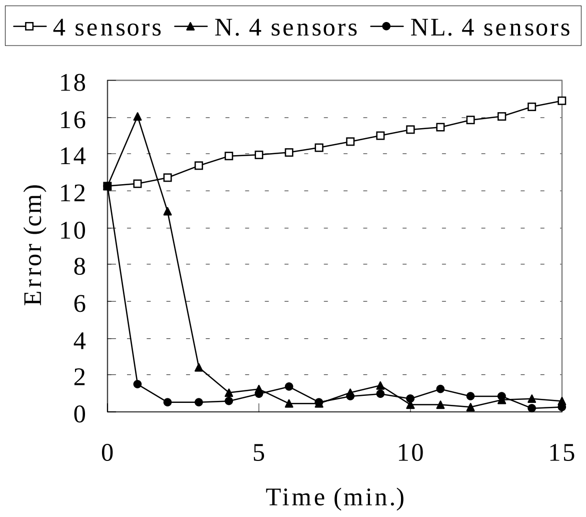


Fig. 10 3-D Error
(without calibration, initial error large)

Figure 9 shows the simulation results in the condition that an initial error is small without initial calibration.

The error is accumulated in the “4 sensors” algorithm. However, the accumulation of the error is not seen in the “N. 4 sensors” algorithm and in the “NL. 4 sensors” algorithm.

4.5.2.2 Large Initial Error

Figure 10 shows the simulation results in the condition that an initial error is large without initial calibration.

The error is accumulated in the “4 sensors” algorithm. The error rapidly becomes small within one minute in the “NL. 4 sensors” algorithm. The error become small within three minutes, though the error increases once, in the “N. 4 sensors” algorithm.

4.6 Summary of Algorithms

The simulation results are summarized.

4.6.1 4 sensors

The error is small compared with the error of the “3 sensors” algorithm. However, the error accumulates. Calibration is necessary.

4.6.2 N. 4 sensors

When an initial error is large, it takes few minutes until the error becomes small. However, the error does not accumulate.

4.6.3 NL. 4 sensors

The result is better compared with the result in the “N.4 sensors” algorithm, when the initial error is large. The error does not worsen against the time. Adopting this algorithm, initial and later-on calibrations will be unnecessary.

5. Evaluation Experiment

The evaluation experiment was performed with three 3-D mice. They are the 3-sensors version 3-D mouse, the 4-sensors version 3-D mouse, and Polhemus ISOTRACK II.

Subjects who evaluated the 3-D mice are ten people.

5.1 Evaluation Method

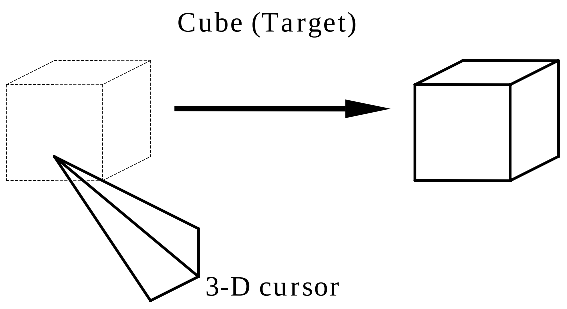


Fig. 11 Evaluation Method

A virtual space is created in the personal computer display by using the liquid crystal shutter glasses. A 3-D cursor, which synchronizes with a 3-D mouse, is displayed in the virtual space.

A cube, which is the target, is displayed there. The cube is moved to a new position at random when a 3-D cursor hit the target as shown in Figure 11. The evaluations are made by numbers of hit counts within a certain time.

For the 3-sensors version 3-D mouse, the subjects had to make initial calibrations. The 4-sensors version 3-D mouse has the “NL. 4 sensors” algorithm which is the automatic error correction without calibration.

5.2 Experiment Results

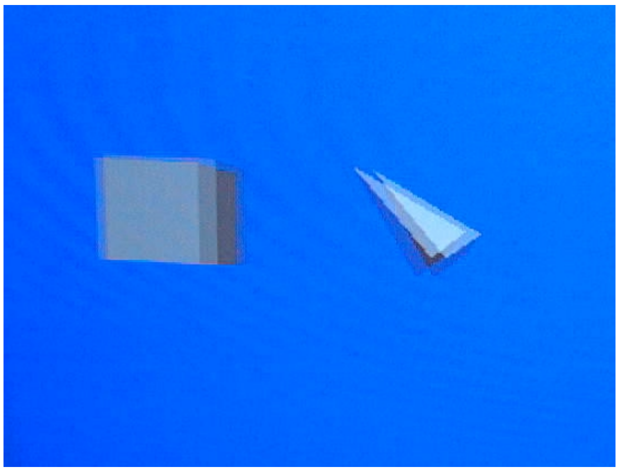


Fig. 12 Screen of the Evaluation Experiment

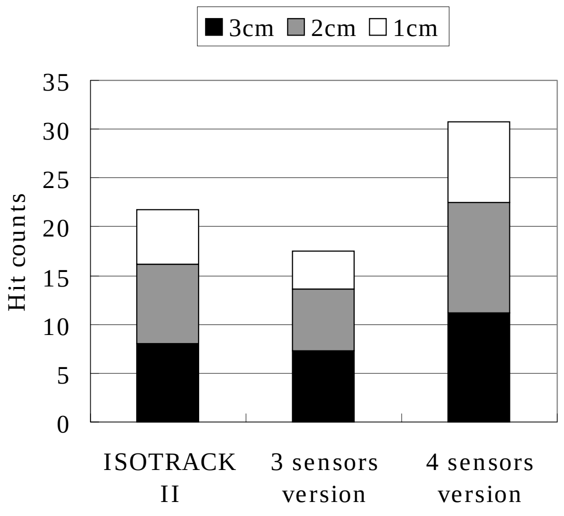


Fig. 13 Result of the Evaluation Experiment

Figure 12 shows the screen of the evaluation experiment. After having had the subjects practiced, they performed the evaluation experiment for three minutes per one mouse. The cubic size was reduced every one minute. Figure 13 shows the results of the evaluation experiment. There were a lot of hit counts for the 4-sensors version 3-D mouse.

5.3 Result of the Questionnaire

Figure 14 shows the results of the questionnaire carried out at the same time as the evaluation experiment. This questionnaire asked for answers by choosing one of five evaluation levels for each item 1-5. Figure 14 shows the average scores for each item.

The 3-sensors version 3-D mouse is inferior to ISOTRACKII in all the points. However, the 4-sensors version 3-D mouse is improved in all the aspects.

5.4 Summary of the Evaluation Experiment

The user should float the arm in the air, when using any 3-D mouse. Therefore, user's tiredness is larger than for the 2-D mouse.

The 3-sensors version 3-D mouse loads all the hanging weights on the pointer. Therefore, the load on the arm is large and user's tiredness is also large. The 4-sensors version 3-D mouse pulls the pointer from the upper board. The pointer does not drop even if the hand does not support the pointer, because the pointer has tendency to go to the center. In ISOTRACK II, the pointer drops when the hand does not support the pointer as in the case of the 3-sensors version 3-D mouse. Moreover, it is difficult to indicate a small object by moving the pointer in the empty air.

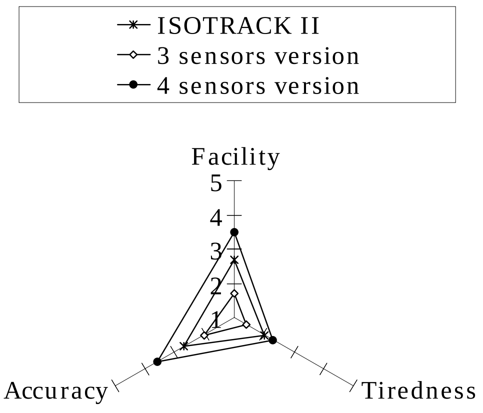


Fig. 14 Result of the Questionnaire

6. Conclusion

In this paper, the simple structured and low-cost 3-D mouse and the auto-error-collection function algorithm are proposed.

Existing 3-D mice are very expensive and far more complex than usual 2-D mice are. In this paper, the simple 3-D mouse, which uses a part of 2-D mouse's mechanism for low-cost, is proposed. The 3-sensors version 3-D mouse has the problem that initial calibration and later-on calibrations are necessary, because the coordinate error accumulate against the time. However, the 4-sensors version 3-D mouse has achieved Auto-error-collection function and requires no calibrations.

It will be possible to improve the hardware to prevent slipping in the future and reduce the coordinate error.

However, the 4-sensors 3-D mouse, which needs no initial calibration and gives easy operation, will keep superior to the 3-sensors 3-D mouse, etc.

References

1. M. Hosoda, K. Itoi and Y. Kobayashi: “Stereoscopic Vision Modeling,” *proceedings of the 1997 information and systems society conference of IEICE*, D-12-69, pp.261 (1997) in Japanese.
2. M. Hosoda, Y. Shimano, K. Itoi and Y. Kobayashi: “3-D CG Modeler Using Stereoscopic Vision,” *The Journal of The IIEEJ*, Vol. 27 No. 6, pp813-818 (1998) in Japanese.
3. “3 SPACE(R) ISOTRACK II USER’S MANUAL,” Polhemus Incorporated (1993).
4. M. Hosoda and Y. Kobayashi: “Simple 3D Mouse with Auto-error-collection Function,” *The Journal of The IIEEJ*, Vol. 28 No. 3, pp270-277 (1999) in Japanese.
5. M. Hirose: “Virtual Reality,” Sangyo Tosho (1993) in Japanese.
6. M. Sato, Y. Hirata, H. Kawarada: “Space Interface Device for Artificial Reality –SPIDAR-,” *The Transactions of IEICE*, Vol. J-74-D-II No.7, pp887-894 (1991) in Japanese.
7. Y. Hirata, M. Sato, H. Kawarada: “A Mesuring Method of Finger Position in Virtual Work Space,” *Forma*, 6, pp171-179 (1991).
8. Y. Cai, S. Wang and M. Sato: “A Human-Scale Direct Motion Instruction System Device for Education Systems,” *IEICE TRANS. INF. & SYST.*, Vol. E80-D No. 2, pp212-217 (1997).