SUMMARY  Augmented reality tasks require a high-reliability tracking method. Large tracking error causes many problems during AR applications. Tracking error estimation should be integrated with them to improve the reliability of tracking methods. Although some tracking error estimation methods have been developed, they are not feasible to be integrated because of computational speed and accuracy. For this study, a tracking error estimation algorithm with screen error estimation based on the characteristic of linecode marker was applied. It can rapidly estimate tracking error. An evaluation experiment was conducted to compare the estimated tracking error and the actual measured tracking error. Results show that the algorithm is reliable and sufficiently fast to be used for real-time tracking error warning or tracking accuracy improvement methods.

key words: augmented reality, linecode marker, tracking method, tracking error, tracking accuracy, error estimation, wheel tracking error computation method

1. Introduction

Augmented reality (AR) is a field of computer research which deals with the combination of real-world and computer-generated data. In fact, AR technology is broadly used in medical, manufacturing, entertainment, and military applications, and in others as well [1].

Tracking, which computes a camera pose (including position and orientation) from the relationship between the feature points in the environment and in the image, is a key technology for AR. Accurate tracking is necessary for many AR applications. For example, tracking accuracy is extremely important when AR is used in navigation tasks or field-work support tasks, such as displaying a radiation map in a nuclear power plant, to prevent field workers from entering dangerous regions. Inaccurate tracking might mean that information is not displayed in the correct position, possibly leading workers into dangerous areas.

In this study, we want to apply AR for use in nuclear power plant (NPP) field-work support. Typically, an NPP is a large area with a very complex environment. Tracking accuracy is the key requirement when applying AR. Furthermore, the conditions in the NPP field are also complex, with various light conditions (from very dark to very bright in different areas), cleanliness (the size of NPP, larger than 300,000 m², requires that an NPP be completely cleaned at long intervals). Therefore, the tracking method should be made in consideration of all possible conditions to meet the requirements of high accuracy, even under bad conditions.

Most traditional tracking methods give only a tracking result without Tracking Error Estimation (TEE) [2]. Others give only some screen error as a measurement of tracking error, such as the reprojection error [3]. The reprojection error cannot be used as a measurement of tracking error because tracking error is not directly and singularly affected by the projection error.

Many methods, such as camera calibration methods, have been proposed to improve the tracking accuracy in the design phase (before AR implementation). However, it is insufficient to consider only the problem in the design phase because the problem becomes much more complex in the run phase. Many causes can produce large tracking error during the run phase (during AR implementation). Those causes mainly include: illumination, marker misidentification, marker misplacement, camera distortion, inaccurate perspective n-point solution method, and low camera resolution.

Recently, some methods to improve tracking accuracy have been proposed in AR. Error estimation has been adopted in related works to improve the tracking robustness [3]–[6]. In fact, [3]–[6] have proposed TEEs and used them to increase the registration accuracy. However, these methods are not feasible for the NPP field-work support system because these methods, including M-estimation and RANSAC method, characterize reprojection error as a measure of tracking error. However, in real conditions, tracking error has many causes. For that reason, these methods are inappropriate to evaluate tracking error correctly, especially when the tracking error is largely affected by other causes. This study proposes a tracking error computation method which can simulate these causes.

To make the tracking methods reliable in the run phase, a real-time TEE (RTEE) method, which considers all causes listed above and can be integrated into the tracking methods, is needed.

Current TEE methods are not reliable RTEE methods because: (1) current TEE methods give no maximum possible tracking error, but instead give a probabilistic [7], [8] tracking error; (2) the computation speeds of these TEE methods are not sufficiently high. A new TEE algorithm must be proposed to compute tracking error from all possi-
ble causes of tracking error during run time.

When the AR applications are equipped with the RTEE, the AR applications can provide the following.

**Run-time Warning of Large Tracking Error** The integration tracking method can give a warning when the tracking error is larger than a threshold value.

**Run-time Tracking Accuracy Improvement** Tracking error might be lessened by selecting a set of the captured feature points with a smaller screen error because the screen error of each captured feature point can be computed.

## 2. Screen Error Computation

Although some causes of tracking error are listed above, it is not possible to compute tracking error directly from them because too many causes exist. For that reason, these causes are simplified to be computed through a concept of screen error $\Delta$. Moreover, RTEE can be realized through estimation of screen error $\Delta$.

In this study, a linecode marker-based tracking method [9] is selected to be the base of the reliable tracking method. Figure 1 is a conceptual illustration of the linecode marker. The linecode marker is a combination of black elements: each element corresponds to one bit. The square element, $3 \text{ cm} \times 3 \text{ cm}$, signifies “0”. The double-sized rectangle element, $3 \text{ cm} \times 6 \text{ cm}$, denotes “1”. The linecode marker has two feature points at the center of the top element and at the bottom element of the linecode marker. The linecode-marker-based tracking method was developed based on a perspective n-point solution method [9].

Linecode markers are applied in this study because the study is a part of AR support system for NPP field-work. Many pipes exist in the NPP environment. For that reason, linecode markers can be installed easily in the NPP environment. The merit of linecode markers is their longer tracking distance (>15 m) than traditional square or circle markers (<3 m) when using a camera of 1024 × 768 with 3-cm-wide markers. This study is developed based on the shape of the linecode marker. Although this method is applied only to linecode markers now, it is possible to develop similar methods for other types of markers. For example, square marker methods can be developed based on the linearity of their side length. For circular markers, ellipticity is useful as a measure.

### 2.1 Screen Error Factors

Figure 2 shows the definition of screen error. Screen error $\Delta$ is the difference between: (1) the ideal image $D$ of the feature point in the screen and (2) the feature point’s estimated projection point $C$ in the screen.

The feature point position is at $D$ if the feature point is computed from the exact pose of the camera and the exact 3D position of the feature point with an ideal screen. Nevertheless, $D$ is unknown because neither the pose nor the 3D position is an exact known value. The recognized feature point is at the real image position $A$. After eliminating the screen distortion, the feature point is at $B$. Common tracking methods assumed $B$ and $D$ as the same point. It is noteworthy that $\Delta_m = ||BD||$ must not be neglected because $\Delta_m$ is sometimes very large.

The estimated projection error from the tracking method is $\Delta_e = ||BC||$, where $C$ is the projected position computed from the estimated pose. The tracking method used in our study, and in most studies, will try to minimize $\Delta_e$ because we do not know the ideal image position $D$. This is insufficient: even if $\Delta_e$ is minimized to zero, the screen error is as large as $\Delta_m$; thereby, the tracking error would be large. Therefore, we must consider a means to compute the screen error for computation of the tracking error.

The screen error $\Delta$ is expected to satisfy

$$\Delta \leq \Delta_m + \Delta_e$$

(1)

because of the triangle inequality. We must consider the worst case in the run phase. The maximum tracking error is expected to be acquired from the maximum possible screen error $\Delta$. Therefore, we take $\Delta = \Delta_m + \Delta_e$ to increase the reliability of our algorithm.

All the causes of tracking error are represented by the screen error $\Delta$. In addition, $\Delta$ is categorized as the marker recognition error $\Delta_m$ and estimated projection error $\Delta_e$. Both the marker recognition error and estimated projection error cause tracking error. It is neither possible nor necessary to tell which cause of tracking error is related to which factor of screen error because all tracking error factors might be related to one or both of the screen error factors.
2.2 Marker Recognition Error

The marker recognition error $\Delta_m$ of the linecode marker has two components: error in the line of the linecode marker (linear error $\Delta_l$) and error perpendicular to the line (perpendicular error $\Delta_p$).

2.2.1 Linear Error Computation

The distance, measured in screen coordinates, between each pair of elements in the same linecode marker is directly related to the distance in the global coordinate if all conditions are ideal. In Fig. 3, this means that

$$\|A_gB_g\| : \|A_gC_g\| : \|A_gD_g\| = \|A_sB_s\| : \|A_sC_s\| : \|A_sD_s\|$$

for the 4-bit linecode marker.

During run time, (2) cannot be expected. The linear error $\Delta_l$ is computed from the difference between the left side and right side of (2).

Compute the unit length in the screen coordinate for the linecode marker. Here, the unit length means the length in the ideal screen coordinate corresponding to the side length of the "0" element in the global coordinate. The linecode marker definition shows that the length of linecode marker with $n$ bits is

$$l_g = \left( \sum c_i + n - 2 - (c_1 + c_n)/2 \right) s_g$$

in the global coordinate. Therein, $s_g$ is the side length of the "0" element; $c_i$ is the $i$th element’s code, either "0" or "1". The length of the linecode marker in the ideal screen coordinate is

$$l_s = \left( \sum c_i + n - 2 - (c_1 + c_n)/2 \right) s_s$$

in the ideal screen coordinate. In that equation, $s_s$ is the length in the screen coordinate corresponding to $s_g$ in global coordinates.

Moreover, $l_s$ can be computed from

$$l_s = \sqrt{(x_1 - x_n)^2 + (y_1 - y_n)^2}.$$  (5)

Here, $(x_1, y_1)$ is the top (first) element of the linecode marker in the ideal screen coordinate; $(x_n, y_n)$ is the bottom (last) element of the linecode marker in the ideal screen coordinate. Therefore $s_s$ can be computed from

$$s_s = \frac{\sqrt{(x_1 - x_n)^2 + (y_1 - y_n)^2}}{\sum c_i + n - 2 - (c_1 + c_n)/2}$$  (6)

The distance between the $i$th element and the top element is expected to be the following.

$$d_i = s_s \left( \sum_{j=1}^{i} c_j + i - 2 - (c_1 + c_n)/2 \right), i = 1, 2, \ldots, n$$  (7)

Compute the differences between the two series of distances. Because the largest possible error should be computed, the maximum possible linear error is computed from the following.

$$\Delta_l = \max_{i=2}^{n} \sqrt{(x_1 - x_i)^2 + (y_1 - y_i)^2} - d_i$$  (8)

For the bottom element, the following is used.

$$\Delta_l = \max_{i=1}^{n-1} \sqrt{(x_n - x_i)^2 + (y_n - y_i)^2} - s_s \left( \sum_{j=1}^{n} c_j + i - 2 - (c_1 + c_n)/2 \right)$$  (9)

2.2.2 Perpendicular Error Computation

All elements of one linecode marker are in one line in the global coordinate system. Consequently, all the elements of one linecode marker are also expected to be in one line in the ideal screen coordinate system. The perpendicular error is computed from the distance between the selected element and the expected line of the linecode marker.

The computation method of $\Delta_p$, the marker recognition error perpendicular to the line of the linecode marker, is as follows.

The line of the linecode can be computed using the least-squares method using each element’s coordinates as

$$K_1 x + K_2 y + K_3 = 0.$$  (10)

In that equation,

$$K_1 = n \sum x_i y_i - \sum x_i \sum y_i,$$

$$K_2 = (\sum x_i)^2 - n \sum x_i^2,$$

$$K_3 = \sum y_i \sum x_i^2 - \sum x_i \sum x_i y_i,$$

and $i = 1, 2, \ldots, n, i \neq \text{the selected element}$. Actually, $\Delta_p$’s definition is shown in Fig. 4. For the top element,

$$\Delta_p = \max_{i=2}^{n} ||K_1 x_1 + K_2 y_1 - K_1 x_i - K_2 y_i|| / \sqrt{K_1^2 + K_2^2}.$$  (11)

For the bottom element,
Fig. 4 Definition of the perpendicular marker recognition error.

\[ \Delta_p = \max_{i=1}^{n-1} \left( \frac{K_1 x_i + K_2 y_i - K_1 x_n - K_2 y_n}{\sqrt{K_1^2 + K_2^2}} \right) \]  

Furthermore, we can obtain

\[ \Delta_m = \sqrt{\Delta_l^2 + \Delta_p^2} \]  

because \( \Delta_l \) and \( \Delta_p \) are mutually perpendicular.

2.3 Estimated Projection Error

Estimated projection error \( \Delta_e \) is commonly used as the screen error and a measurement of tracking error directly in many studies [11]. \( \Delta_e \) is obtainable from tracking method directly as

\[ \Delta_e = \max_{i=1}^{k} \sqrt{(x_i - x'_i)^2 + (y_i - y'_i)^2} \]  

Therein, \( k \) is the number of feature points used for tracking. Also, \( (x_i, y_i) \) is the coordinates of the \( i \)-th feature point in the ideal screen coordinate; \( (x'_i, y'_i) \) corresponds to \( B \) point in Fig. 2; \( (x'_i, y'_i) \) is the projected position of the \( i \)-th feature point computed from the estimated pose using the tracking method and corresponds to \( C \) in Fig. 2.

The largest possible \( \Delta \) can be computed as

\[ \Delta = \Delta_m + \Delta_e \]  

3. Real-Time Tracking Error Estimation (RTEE)

3.1 Definitions and Settings

The camera pose is a six-dimensional vector including the camera’s three-dimensional (3D) position and 3D rotation. We say that a marker is visible to the camera if the marker is within the camera’s view.

Pose \( C' \) is the unknown real camera pose. Pose \( C \) is the estimated pose from the perspective n-point solution method. Here the following variables are used: \( Q_i, i = 1, 2, \ldots, n \) is the 3D position of visible FPs in the camera coordinate; \( m_i \), a 2D vector, is the image point of \( Q_i \) in pose \( C \) screen coordinate; \( m'_i \) is the image point of \( Q_i \) in pose \( C' \) screen coordinate. The image difference of \( Q_i \) between the camera at \( C \) and that at \( C' \) is defined as \( \|m_i - m'_i\| \). The largest image difference (LID) between the camera at \( C \) and that at \( C' \) is defined as the largest image difference among any two correspondent marker pairs, i.e. \( m_i \) and \( m'_i \), in both images captured by camera at \( C \) and at \( C' \). Finally, LID is defined as

\[ LID(C', C, M) = \max_{i=1,2,\ldots,n} \|m_i - m'_i\| \leq \Delta. \]  

In that equation, \( M \) denotes the marker arrangement. The marker’s position accuracy on the screen is always limited. The largest screen error of all the feature points is \( \Delta \). Therefore \( LID(C', C, M) \leq \Delta \). The tracking error region is composed by all possible poses \( C \) that satisfy \( LID(C', C, M) \leq \Delta \).

Figure 5 shows a conceptual image of the tracking error region. The perspective n-point problem solution, the estimated camera position, comes somewhere in the tracking error region. The unknown real pose should also be at some point in the tracking error region. The RTEE’s objective is to find the most distant possible real pose in the tracking error region from the estimated camera position.

3.2 Computation of Limitation Circles

The algorithm used for computing the tracking error region is not complicated. Presuming that \( n \) FPs are captured by the camera (\( n \) is expected to be greater than three for tracking [12]), compute the tracking error region as the intersection region of the following areas in the x-z plane: (1) for each 2 FPs among \( n \) FPs, find the \( C_2 = \frac{n(n-1)}{2} \) areas in which the real camera might be if x-coordinates of the two FPs have an error of \( \Delta \); (2) for each single marker, find the \( C_1 = n \) area in which the camera might be if the y-coordinate of the marker has an error of \( \Delta \). These areas are enclosed by circles. These circles are called limitation circles.

3.2.1 Limitation Circles for the x Coordinate

In Fig. 6, \( P_A(x_a, 0, z_a) \) and \( P_B(x_b, 0, z_b) \) signify projection points of \( Q_A(x_a, y_a, z_a) \) and \( Q_B(x_b, y_b, z_b) \) on the x-z plane in camera coordinates. In addition, \( A \) is the intersection point between \( CP_A \) and the screen x axis; \( B \) is the intersection point between \( CP_B \) and the screen x axis; and \( A_A, A_1, A_2, B, B_1, B_2 \) are on the screen x axis and set \( \|A_1A\| = \|AA_2\| = \|B_2B\| = \|BB_1\| = \Delta \). Any possible real pose should not capture \( A \) outside of the line segment \( [A_1, A_2] \) or \( B \) outside of the line segment \( [B_1, B_2] \). Consequently, pose \( C' \), which can capture same x dimension images of \( P_A \) and \( P_B \) as the real camera pose, satisfies
Here $\alpha_2 = \angle A_2CB_2$ and $\alpha_1 = \angle A_1CB_1$. They can be computed from

$$\tan \angle A_1CD = \tan \angle ACD + \frac{\|A_1A\|}{f} = \frac{x_a}{z_a} + \frac{\Delta}{f}$$
(18)

$$\tan \angle A_2CD = \tan \angle ACD - \frac{\|A_2A\|}{f} = \frac{x_a}{z_a} - \frac{\Delta}{f}$$
(19)

$$\tan \angle B_1CD = \tan \angle BCD + \frac{\|B_1B\|}{f} = \frac{x_b}{z_b} + \frac{\Delta}{f}$$
(20)

$$\tan \angle B_2CD = \tan \angle BCD - \frac{\|B_2B\|}{f} = \frac{x_b}{z_b} - \frac{\Delta}{f},$$
(21)

where $D$ is the center of the screen and $f$ is the focal length of the camera focal length. The track of $C'$ on $\angle P_1P_2P = \alpha_1$ and $\angle P_1P_2P = \alpha_2$ are circles 1 and 2, as shown in Fig. 7. The $C'$ that satisfies (17) is expected to be in the gray area between the two circles.

The computation methods for circles 1 and 2 are identical. Here we take circle 1 for description in Fig. 7. First assume that $F$ is on circle 1 and satisfies $\|PF\| = \|OP_1\|$ to compute the center of circle 1, and that $E$ is on $P_1P_2$ and $\|PE\| = \|P_2E\|$. Because $\|PF\| = \|P_2F\|$, $EF$ is mid-normal of $P_1P_2$. Therefore, the center of circle 1, point $O(x_O, 0, z_O)$, is on $EF$. Because $\|OF\| = \|OP_2\| = R$ and $\angle P_1F_2 = \alpha_1$ ($R$ is the radius of circle 1), $\angle OP_2 = \angle OP_1 = \alpha_1/2$. Therefore,

$$R = \frac{\|OP_1E\|}{\sin \angle EOP_1} = \frac{\|OP_2E\|}{2 \sin \alpha_1}.$$  
(22)

The slope angle of $P_1P_2$ is $\gamma = \tan \frac{\Delta}{x_0 - z_0}$; the slope angle of $P_1O$ is $\theta = -(\pi/2 - \alpha_1 - \gamma)$. Consequently,

$$x_O = x_o + R \cos \theta$$
(23)

$$z_O = z_o + R \sin \theta.$$  
(24)

The gray area in Fig. 7 is called the x limitation area. We can compute the x limitation area for any two FPs in the $n$ FPs as for $QA$ and $QB$.

### 3.2.2 Limitation Circles for the y Coordinate

Consider the y direction in the screen coordinate, as shown in Fig. 8. Set $\|A_1A\| = \|AA_2\| = \Delta$, $C_1A$ is the projection line of $CQA$ on the x-z plane and $A_2A_1$ is parallel to the y axis of the screen coordinate. $D$ is the center of the screen and $\|CD\| = f$. Then $E$ is a projection point of $A$ on the x axis of the screen coordinate.

Consequently, $C'$, which can capture the image of $QA$ with a difference smaller than $\Delta$ in the y direction with $C$, is expected to satisfy

$$\beta_2 \leq \angle QAC'P_A \leq \beta_1.$$  
(25)

In that expression, $\beta_2 = \angle A_2CP_A$ and $\beta_1 = \angle A_1CP_A$. Centers of both circles are $P_A$. Radii of the circles can be computed as $\|QA_P\| = \|Q_A\|\tan \beta_2$ and $\|QA_P\|\tan \beta_1$, where the following are true.

$$\beta_2 = \tan (\angle QACPA - \Delta \cos \angle DCE)$$
(26)

$$\beta_1 = \tan (\angle QACPA + \Delta \cos \angle DCE)$$
(27)

$$\angle DCE = \tan (x_o/z_o).$$
(28)

The area (a ring) between the two circles is called the y limitation area. The y limitation areas must be computed for any marker in the $n$ FPs as for $QA$.

### 3.3 Computation of the Tracking Error Region and Tracking Error

Section 3.2 introduced a computation method for computing $O(x_O, 0, z_O)$, is on $EF$. Because $\|OF\| = \|OP_2\| = R$ and $\angle P_1F_2 = \alpha_1$ ($R$ is the radius of circle 1), $\angle OP_2 = \angle OP_1 = \alpha_1/2$. Therefore,
x and y limitation areas. The intersection region of all the 
x and y limitation areas is the tracking error region because 
the camera can capture all FPs exactly as the real pose. Fig-
ures 9 and 10 show the computation method for the tracking 
error region. To introduce the computation method easily, 
we use a marker arrangement with three FPs.

In the following, the tracking error region computation 
procedure will be explained. All FPs, QA, QB, and QC are 
projected respectively to PA, PB, and PC in the x-z plane of 
the camera coordinate. Figure 9 shows that the camera at 
any position of x limitation area of P_A and P_B can assure 
x image error of Q_A and Q_B smaller than Δ; therefore, any 
point in the intersection region in Fig. 9 can capture all Q_A, 
Q_B and Q_C x images, exactly as a real camera can. The 
camera at any position y limitation area of QA can ensure 
y image error of QA smaller than Δ. Then, compute y lim-
itation areas for QB and QC similarly as that for QA. The 
intersection region depicted in Fig. 10 is the tracking error 
region.

We can compute the edges of the tracking error region 
as a set of arcs. The most distant position in the tracking error 
region from the real camera must be at one intersection 
point of the circles introduced above (the reason is explained 
in Sect. 3.4). We must compute two circles to generate an x 
limitation area for each pair of FPs and compute two cir-
tles to generate a y limitation area for each marker when 
the marker arrangement has n FPs. Consequently, the total 
number of limitation areas is 

\[ C_n^2 + C_n^1 = \frac{n(n-1)}{2} + n = \frac{n(n+1)}{2}. \]  

(29)

We must compute n(n+1) circles and n(n+1)/2 limitation 
areas for n FPs. Then we must compute the intersection 
points of these circles and thereby determine the most dis-
tant intersection point which can exist in all limitation areas. 
The distance between that point and C is the tracking error.

Rotate the x-z plane to y-z plane in the RTEE method 
for evaluation of the tracking error in the y direction. The 
TER in the y-z plane is different from that in the x-z plane. 
The tracking error in the y-z plane is possibly larger than that 
in the x-z plane. Demonstrably, this extension can improve 
the RTEE algorithm accuracy. The larger tracking error is 
selected from the two errors as the estimated tracking error.

3.4 Explanations for Computing Only Intersection Points

As described in Sect. 3.3, we need to prove that the most 
distant point in the tracking error region is one intersection 
point of all the limitation circles. In addition, the most dis-
tant point in any intersection arc to C is at one end of the 
arc.

Figure 11 shows that C is the origin of the camera co-
ordinate. One limitation circle O is at (x_O, 0, z_O) with radius 
of R. Furthermore, z_O > 0 because marker(s) that generated 
circle O is/are expected to be visible to the camera. Also, D 
and E are two intersection points between circle O and other 
circles. We next assume that DE is one edge of the tracking 
error region, and that K is the most distant point from C on 
circle O and \[\|CK\| = \|CO\| + R.\] We must prove that the 
greatest distance from C to any point on DE is either \[\|CD\|\] 
or \[\|CE\|.\] The problem can be described as 

\[\max f(t) = x^2 + z^2,\]  

(30)

where 

\[x = x_0 + R \cos(t)\]  

(31)

\[z = z_0 + R \sin(t)\]  

(32)

Fig. 11  Proof that the largest point is at one end of the arc.
strict to

\[ t \in [t_D, t_E] \subset \left[ \frac{\pi}{2} - \arctan \frac{x_0}{z_0}, \frac{5\pi}{2} - \arctan \frac{x_0}{z_0} \right]. \quad (33) \]

In fact, (33) includes the assumption that \( K \notin \overline{DE} \). Because (1) at least one marker exists on circle \( O \) or at \( O \) and (2) \( \overline{DE} \) is one edge of the tracking error region, if \( K \in \overline{DE} \), then the tracking error is equal to or greater than \( ||CK|| \). This tracking error is greater than the distance between the marker and the camera. Therefore, this kind of marker arrangement is not feasible and must be modified in the case in which \( K \in \overline{DE} \) can be determined and eliminated by checking whether \( t_K \in (t_D, t_E) \). Therefore, we need to consider only the case for which \( K \notin \overline{DE} \).

Substitute (31) (32) into (30):

\[ \max f(t') = k_1 + k_2 \sin(t') \quad (34) \]

where \( t' = t + \arctan(x_0/z_0), k_1 = x_0^2 + z_0^2 + R^2, k_2 = 2R \sqrt{x_0^2 + z_0^2}, t_D' = t_D + \arctan(x_0/z_0), t_E' = t_E + \arctan(x_0/z_0). \)

The problem can be resolved through proof by contradiction. Assume that \( \exists t' \in (t_D', t_E') \) satisfies \( f(t') > f(t_D') \) and \( f(t') > f(t_E') \). Using (34),

\[ f(t') - f(t_D') = 2k_2 \sin \frac{t' - t_D'}{2} \cos \frac{t' + t_D'}{2} > 0 \quad (35) \]

\[ f(t') - f(t_E') = 2k_2 \sin \frac{t' - t_E'}{2} \cos \frac{t' + t_E'}{2} > 0. \quad (36) \]

Because \( \sin \frac{t' - t_D'}{2} > 0, \sin \frac{t' - t_E'}{2} < 0 \) and \( k_2 > 0, \)

\[ \cos \frac{t' + t_D'}{2} > 0, \cos \frac{t' + t_E'}{2} < 0. \quad (37) \]

Furthermore, \( \frac{t' + t_D'}{2}, \frac{t' + t_E'}{2} \in \left( \frac{\pi}{2}, \frac{5\pi}{2} \right) \). Therefore,

\[ \frac{t' + t_D'}{2} > \frac{3\pi}{2} > \frac{t' + t_E'}{2}. \quad (38) \]

This contradicts \( t'_E > t'_D \). Therefore, \( f(t') < f(t_D') \) or \( f(t') < f(t_E') \), which means that \( ||CD|| \) or \( ||CE|| \) is the greatest distance from \( C \) to any point on \( \overline{DE} \).

4. Evaluation Experiment

An evaluation experiment was done in a large room to evaluate the RTPE performance. The evaluation points include reliability, computation speed, and accuracy.

4.1 Hardware and Software Specifications

Because of the desired long distance and high accuracy, a monochrome Dragonfly camera (1024 × 768, focal length = 6.37 mm, pixel size = 0.00465 mm) was used: a color camera’s pixel information accuracy is lower as a result of its color processing [13].

4.2 Experiment Configuration

Figure 12 portrays the experiment configuration. The global coordinate is defined with the origin at \( O \) in Fig. 12, the \( x \) direction is set toward the reader, the \( y \) direction is set as up, and the \( z \) direction is set as right. The NPP environment includes many vertical pipes. All markers are placed vertically to simulate the NPP environment because most markers will be placed vertically in the NPP environment on these vertical pipes. The placement error was within 1 cm. This configuration was adopted to simulate the complex and large-scale environment in the NPP field. The camera was placed in front of the markers; the camera was moved from 0 to 8 m in the \( x \) direction and from 2 to 11 m in the \( z \) direction. The camera was rotated at every point with 0, 20, 40 deg (to the direction in which markers can be captured). The camera captured three frames at each point. Subsequently, the average tracking error of the three frames was computed as the result of tracking error. The camera was fixed during image capturing because movement is dangerous when watching the AR devices in the complex NPP environment. The worker stands without much movement when using the NPP field-work support system. For that reason, the camera will not move very much and the effects of error, such as blur, can be neglected. The average illumination condition in the room was 1050 lux. No linecode-marker-like articles were present in the experiment room.

4.3 Tracking Error Measurement

To evaluate the tracking accuracy, the estimated camera pose must be estimated from the tracking result and the real camera pose must be measured manually. However, the 3D rotation of the camera is difficult to measure; for that reason, only the 3D position is evaluated. The vector \( (X_C', Y_C', Z_C')^T \) is the estimated position of camera in global coordinate. The position of camera is measured manually as \( (X_C, Y_C, Z_C)^T \). The estimated camera position error is defined as

\[ E = \sqrt{(X_C - X_C')^2 + (Y_C - Y_C')^2 + (Z_C - Z_C')^2} \quad (39) \]
4.4 Experiment Result

In the resultant images, the real tracking error is computed from the difference between the measured position and estimated position from the tracking method. The estimated tracking error is computed from the screen error \( \Delta = \Delta_m + \Delta_e \) using the RTEE method. Figure 13 shows a comparison between the estimated and the real tracking error values.

When a camera is placed in 0 degrees, the estimated tracking error is greater than the real tracking error because \( \Delta_m \) and \( \Delta_e \) are not always in opposite directions. Few points have tracking error greater than 40 cm. Almost all, except the points at (8, -0.085, 10) m, are estimated correctly. Although the estimated tracking error is larger, it is more reliable because it is possible for the tracking error to be as large as estimated through some adjustment of the experimental conditions, such as the illumination or threshold. For that reason, the estimation method is more reliable, although the tracking error seems too large.

There are three points at which the tracking errors are larger than 40 cm when camera is placed in 20 degrees. These points are computed correctly. The pose at (2, -0.085, 3) has a large estimated tracking error (5.3 m) because multi-solution occurs in the y-z plane. Although the tracking method selected the correct one, it is not a reliable result because it is also possible that the tracking method can select the other one. Although the tracking method did not give a result with a tracking error as 5 m, we can not be certain that the tracking method will not give such a result at a nearby pose or at another capture at the same pose. We chose not to use a marker arrangement, as in the experiment, if (2, -0.085, 3) and 20 degrees is a pose that would be used in NPP field-work.

When the camera was placed at 40 degrees, many poses showed tracking error greater than 40 cm and estimated correctly using the RTEE algorithm.

The computation speed of the RTEE is quite feasible at 1 ms per frame using a Pentium M 1.6 GHz (Intel Corp.) laptop PC. Consequently, RTEE can be readily integrated into the tracking method.

Figure 14 shows a comparison of the results obtained using our approach and the traditional approach (left is the real tracking error, center is the estimated tracking error...
computed from our proposed method, and right is the estimated tracking error computed from $\Delta_e$. Traditionally, researchers have considered that $\Delta = \Delta_e$ [2]–[5]; for that reason, the estimation algorithm cannot detect a large tracking error (the right figure of Fig. 14). Both the marker recognition error and estimated projection error should be considered during the TEE.

5. Conclusion

A series of screen errors is defined and their computation methods are proposed to improve computation methods for tracking error. An RTTE method is developed to realize real-time tracking error estimation using a higher resolution camera for linecode markers. An evaluation experiment was performed to evaluate the estimation accuracy. Results show that The RTTE method is fast and sufficiently reliable to be integrated with the tracking method.

References


Zhiqiang Bian received B.Eng. and M.Eng. degrees, respectively, from Tsinghua University, China in 2001 and 2004. Now he is a Ph.D. student at the Graduate School of Energy Science, Kyoto University, Japan. His research interests include augmented reality, human interface evaluation, and numerical computation.
**Hirotake Ishii** received his B.S. and M.S. degrees in Electrical Engineering and Ph.D. in Energy Science from Kyoto University, respectively, in 1996, 1998, and 2000. Since 2000, he has been a Research Associate in Energy Science, Kyoto University. His research interests are in human–machine interfaces, virtual reality, and augmented reality.

**Hiroshi Shimoda** received a Ph.D. in Engineering from Kyoto University in 1998. He was a Research Associate and an Associate Professor at the Graduate School of Energy Science, Kyoto University from 1996 and 1999, respectively. His main research interest is human interfaces.

**Masanori Izumi** joined the Japan Nuclear Cycle Development Institute, JNC (now Japan Atomic Energy Agency, JAEO) in 2002. He has experience as an operator of Fugen NPS during 2002-2003. He then took charge of developing system engineering for the decommissioning of Fugen NPS from 2003. In addition, he received an M.Eng. degree from Fukui University in 2006.