Performance of displayed-marker-based position tracking

Akio YAMAMOTO† and Taku NAKAMURA†

† Department of Precision Engineering, The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan
E-mail: akio@aml.t.u-tokyo.ac.jp

Received: 9 July 2018; Revised: 23 August 2018; Accepted: 24 October 2018

Abstract
This paper analyzes stability and tracking performance of displayed-marker-based position tracking on visual displays, such as an LCD (Liquid Crystal Display), which have been utilized in several studies about computer-human interaction systems. The tracking system dealt in this paper consists of an LCD and a target object with photo sensors that is placed on the display surface. A fiducial marker image is displayed beneath the object and the photo sensors detect the relative displacement between the marker and the object. The detected displacement is fed back to the tracking program and the program updates the marker position such that it will track the object. In general, this kind of tracking methods can suffer from display latency that arises from programs, graphic engines, and internal signal processing circuits of LCDs. This paper investigates the latency characteristics to reveal that the lag is not constant; it fluctuates with time. The paper, then, formulates the tracking system and analyzes how the lag affects the tracking stability. Then, the paper analyzes the tracking performance of two different classes of stable tracking algorithms, which are PD control and lag compensation. Based on the analyses, the paper provides a guideline on the selection of tracking algorithms, as well as tracking parameters. The analyzed results, as well as the guideline, are verified by experiments in 1-DOF horizontal motions.

Keywords : Marker tracking, Position sensing, Interactive systems, Built-in sensor, Display lag, LCD

1. Introduction

Recently, large-screen liquid crystal displays (LCDs) have spread widely across our society. Physical interactions on such large displays have been of interest in the field of computer-human interactions (CHI). A good example of such physical interactions is tangible interface systems (Ullmer and Ishii 1997; Pangaro, et al. 2002; Weiss, et al. 2010; Mi, et al. 2012). In those systems, physical objects are placed on displays such that users can intuitively interact with the computers through manipulation of those physical objects. Another example is planar haptic systems on display surfaces. In a conventional class of those systems, actuators to provide haptic feedback are arranged around the display, whose mechanical outputs are delivered to the operating pad on the screen, through mechanical links (Portillo-Rodriguez, et al. 2012) or cables (Saga and Deguchi 2012). Another class of planar haptic systems utilizes planar actuators that are arranged on or beneath the display to provide haptic force directly to the operating pad without any mechanical links or cables (Noma, et al. 2004; Nakamura and Yamamoto 2016).

One of the technical issues in developing those physical interaction systems is how to detect the position of the tangible objects or the operating pads. In case of link-driven or cable-driven haptic systems, the position can be easily estimated from the rotation angles of the motors. However, for the other cases, dedicated sensors to detect the positions are required. Visual tracking using an overhead camera is one of the common methods for such a purpose (Pangaro, et al. 2002; Noma, et al. 2004). However, they typically suffer from the occlusion problem. The occlusion problem can be solved by using another class of visual tracking that utilizes frustrated total internal reflection (FTIR) (Mi, et al. 2012), diffused surface illumination (DSI) (Weiss, et al. 2010), or back-projected infrared light (Echtler and Kaltenbrunner 2016). However, they require special setups, which cannot be easily mounted on off-the-shelf LCDs. Capacitive sensing is another popular detection method on display surfaces. It has been favored in small displays such as smart-phones and
tablet PCs; many modern displays are now shipped with capacitive position sensing functions. In principle, the sensing method can be expanded to large displays, but the use of capacitive position sensing on large displays is still quite limited.

In this work, we focus on another, much simpler, position detection method that is displayed (or projected) marker tracking (Lee, et al. 2005, Summet and Sukthankar 2005; Kojima, et al. 2006; Sugimoto, et al. 2007; Kawamoto, et al. 2008). In this method, a fiducial marker is graphically rendered beneath the target object on the display, as shown in Fig. 1. The object is equipped with photo sensors such that the relative position, or positional error, between the marker and the object can be detected. The detected error is sent to the tracking program through an analog-to-digital (AD) converter and the program updates the marker position to eliminate the error. As a result, the marker tracks the object and the program can estimate the position of the object. This method can be easily implemented to any LCDs or any other types of displays including projectors, since it does not require any modification to the display. Only requirements are that the object must be equipped with photo sensors (typically photo transistors/diodes) and that there is a communication channel (either wired or wireless) between the photo sensors and the computer program. In addition, the method can be easily extended to multi-object detections.

There are, however, two technical concerns for this method, arising from the nature of its principle. The first concern is regarding stability. In this method, the positional error between the marker and the object is fed back to the tracking program. The resulting closed loop may have instability if there is any lags (or latencies) within the loop. The second is regarding the tracking speed. For the system to operate properly, the moving speed of the physical object must be limited such that the positional error never exceeds the marker size. Otherwise, the photo sensors will lose the marker and the object tracking will fail. Once the tracking fails, the system cannot recover the position tracking unless an initialization process is activated. Therefore tracking failure must be avoided. These concerns, albeit their importance, have not been studied well in literature. In most of the literature about the object tracking systems, only the concept of the tracking was described and no explicit formulation was given. In addition, they did not explicitly discuss the stability problem or the performance limitation. There would be trade-off between the stability and the tracking performance, but so far, there is almost no clue for this trade-off. Therefore, if a developer of CHI systems tries to implement this kind of tracking method to their systems, the implementation would require a lot of trial-and-error adjustments to ensure stability and good tracking performance.

With such a background, this work formulates and analyzes the method in terms of its stability and tracking errors to provide an insight for better understanding of the method, as well as to provide a design guideline for CHI system developers. The work explicitly formulates two representative algorithms that can ensure stability and provides reference materials for developers of this kind of tracking systems. The main contributions of the work are: (1) the work reveals how a display system behaves in the context of displayed-marker tracking, (2) the work proposes a simple definition of the display lag for analyses, and (3) the work provides numerically obtained stability criteria, which are complicated for obtaining analytically, for two different tracking algorithms. These contributions are explained in the rest of the paper as follows. The next section explains the principle of the detection method, with a brief review of related studies. Section 3 investigates the lag of a tracking system, which mainly comes from graphic engines and LCDs, to reveal its characteristics, and defines the lag of the system for analyses. Section 4 models the detection system as a simple discrete linear system in one degree-of-freedom (DOF). Based on the model, the impact of a lag on the stability is discussed. To recover the stability, two tracking algorithms are compared, which are proportional-differential (PD) control and lag compensation. Their stabilities and maximum errors are discussed. The theoretical analyses are experimentally verified in section 5. Under two different rendering environments with different lags, the stability and tracking performance in one-DOF motion are investigated. Section 6 concludes the work.

Fig. 1 Marker-based on-display position tracking.
2. Principle and related work

In literature, two similar but different types are discussed for position detection based on the displayed-marker tracking. In one type, a global marker that covers the whole (or a large part of) display area is utilized (Raskar, et al. 2004; Lee, et al. 2004). The other type, which is focused on in this work, utilizes a local marker that has, more or less, the same size as the footprint of a target object, such as in Fig. 1 (Lee, et al. 2005, Summet and Sukthankar 2005; Kojima, et al, 2006; Sugimoto, et al. 2007; Kawamoto, et al. 2008).

In global marker systems, position information is buried in the marker such that the absolute position can be estimated only by reading the marker information. To facilitate it, the marker image is modulated, either spatially, temporally, or both, such that the position information can be obtained by demodulating the pattern. One good example is to use sequential Gray-codes (Raskar, et al. 2004). As a global marker occupies the majority of the display area, the display needs to be fully devoted for position detection purpose. Therefore, the applications of the global marker systems are limited. In literature, the method is, for example, combined with a projector for screen alignment (Lee, et al. 2004). Also, the global marker is temporarily utilized for initialization of the local marker systems (Kojima, et al. 2006, Kawamoto, et al. 2008).

In local marker systems, which this paper deals with, markers typically have a similar size as the target objects and are rendered just beneath them such that the rest of the display area can be used for other purposes. The local markers also have modulated image patterns such that the photo sensors in the object can detect its relative position, or positional error, to the marker. In literatures (Kojima, et al. 2006; Sugimoto, et al. 2007; Kawamoto, et al. 2008), dual gradation patterns are proposed as a robust image against rotation (see Fig. 2 (a)). The photo sensors detect the brightness of the pixels beneath them, which is sent to the tracking program. The program then calculates the positional error from the brightness information and updates the rendering position of the marker, such that the marker always presents beneath the object.

In this work, we simplify the system to horizontal one-DOF motion to facilitate analyses. Fig. 2 (b) shows a dual gradation marker pattern, simplified for horizontal one-DOF. For one-DOF motion, the positional error, $e[n]$, can be calculated as

$$e[n] = \frac{b_1 - b_2}{b_{\text{max}} - b_{\text{min}}} \cdot \frac{W_{\text{marker}}}{2},$$

(1)

where $b_1$ and $b_2$ are the brightness detected by the photo sensors, $b_{\text{max}}$ and $b_{\text{min}}$ are the maximum and minimum brightness of the gradations, and $W_{\text{marker}}$ is the width of the marker. The error and the width are measured in pixels.

Most of the literature on displayed-marker tracking do not explicitly show the tracking algorithm. However, their algorithms can be interpreted as follows for the 1-DOF case. Let $x[n]$ denote the position of the object and $m[n]$ the marker position at the $n$-th program execution cycle. The relative position, or positional error, $e[n]$, is expressed as

$$e[n] = x[n] - m[n].$$

(2)

The error is read by the program and is used to update the marker position in the next cycle as

$$m[n] = m[n-1] + e[n-1].$$

(3)

The newest marker position represents the estimated object position and will be used as the output of the system, $y[n]$.

$$y[n] = m[n].$$

(4)
From (2)-(4), we yield \( y[n] = x[n – 1] \), which shows the system tracks the target object with one cycle delay. For two-DOF planar motion, the same will be done along the other axis.

If all the components work ideally, this algorithm should realize perfect motion tracking. However, in actual implementations, this simple algorithm is sometimes found unstable. Especially, when the program execution cycle is minimized to match the display refresh cycle for high-speed tracking, the above simple algorithm easily oscillates due to the lag of the display. For static or relatively slower applications, the program cycle time can be set much longer than the refresh cycle time of the display. In such a case, the display lag can be ignored. However, for faster applications, such as games or interactive systems, lag evaluations are imperative.

Fig. 3 explains the above using block diagrams. Here, \( X(z) \), \( Y(z) \), \( M(z) \), and \( E(z) \) are the Z-transform of \( x[n] \), \( y[n] \), \( m[n] \), and \( e[n] \), respectively. Fig. 3(a) shows the ideal case when the display lag is negligible. If there is a non-negligible display lag, it appears as a feedback element \( D(z) \) in the control loop, as shown in Fig. 3(b). This feedback element can destabilize the system.

From a viewpoint of control engineering, the controlled plant in Fig. 3(a) and (b) would be \( 1/(z – 1) \), which is the combination of the integrator and the unit delay, and the controller is just a unity gain. Under such interpretation, the most straightforward approach to recover the stability is to add a compensator \( C(z) \) before the plant, as shown in Fig. 3(c). As the compensator, we will consider a PD controller in Section 4.1, where the system stability and performance under different PD gains will be analyzed. Another interpretation of Fig. 3(a) and (b) would be that the integrator is the controller and the unit delay is the controlled plant. The algorithm discussed in Section 4.2 is rather based on this interpretation. Regardless of how we interpret the system structure, to analyze the whole system, we should know the characteristics of the display lag, \( D(z) \), which will be investigated in the next section.

3. Lag of Tracking Systems

In typical PC environments, lags can arise from several different components involved in the graphic rendering: a rendering program, a graphic engine, and an internal processing circuit of an LCD. Such display lags have been popular issues in the field of interactive games and virtual realities, as the lag deteriorates the experiences. For example, in (Ivkovic, et al 2015) and on web sites (RenderingPipeline online; Simmons online), the display lag has been discussed from those viewpoints. However, how those lags affect the displayed-marker tracking has not yet been studied.

This work evaluated the lag from the view point of the displayed-marker tracking. The measurement was done using the PC and the LCD monitor used in the later experiments (Laptop PC: Sony VAIO SVS1512AJA in stamina mode, LCD monitor: BenQ FP71G+, 17 inch, 60 Hz progressive mode). The whole screen of the LCD was filled with a uniform brightness (grayscale) using an OpenGL(+glut) program running on Windows 8 operating system. The brightness was changed by the program with a constant interval. At the same time, the program outputs the commanded brightness, through a digital-to-analog (DA) converter, for comparison.

The program (or the display function of the program that handles the tracking, correctly speaking) was repeatedly executed at an almost constant rate. The execution was (a kind of) synchronized with the vsync (or vertical sync that represents refresh timing of the LCD) signal of the display, which occurs at 60 Hz in the experimental setup. The vsync synchronization was realized using wglSwapIntervalEXT() function provided in wgltext.h library, which is a common synchronization method in Windows OS. As verified later, however, this method does not perfectly “synchronize” the
On the surface of the LCD, the brightness of the screen was measured using photo transistors (PT, ParaLight L-31ROPT1C connected to a 100 kΩ resistor). An oscilloscope recorded both the readings of the PTs and the commanded brightness given through the DA converter. One of the measured results is plotted in Fig. 4. In this measurement, the commanded brightness was changed from 0.3 (dark gray) to 0.6 (light gray) to see how the actual brightness on the display changes. The vertical dashed lines represent the execution cycle of the program. The top plot represents the brightness commanded in the program. The middle plot is the reading of the photo sensor located on the surface of the LCD. This photo sensor output was sent to the program through an AD converter. The value read by the program was echoed back using the DA converter, which is plotted in the bottom plot as “obtained brightness”.

The comparison between the top and middle plots shows that the actual brightness change (which is represented by the reading of PT) on the LCD is delayed for more than two cycles. This would be due to the processing lag in graphic rendering pipelines. The middle plot shows that the rising spans for two cycles, which would be due to the slow response of liquid crystal. The middle plot also shows that the actual brightness change is not synchronized with the execution cycle of the program. As a result, the waveforms of “obtained brightness” are not always the same; they depend on the timing of the AD conversion that is synchronized with the program execution cycle.

In the actual implementation, the only information available for the tracking program is “obtained brightness”. Therefore, the lag that appears in this brightness is important. For the analyses in the following sections, this paper models the lag as in Fig. 5. As observed above, the lag consists of two parts. One is the lag caused by the execution delay in the graphic pipelines, which is an integer multiple of the execution cycle and thus can be counted using an integer. The other lag is caused by the slow rising response of liquid crystal. This lag typically results in two-step responses as schematically illustrated in Fig. 5. To evaluate these lags using a single parameter, we propose the following definition of the total lag, \( \tau \).

\[
\tau = \tau_b + \tau_r. \tag{5}
\]

In this definition, \( \tau_b \) is an integer number expressing the number of cycles until the rendering command takes effect. As there is always one cycle delay in the AD-converted brightness due to the nature of the system, one cycle is omitted when measuring \( \tau_b \). The second term, \( \tau_r \), represents the rising response, which is measured by the residual error after the first step. The residual error is normalized by the total step height, and therefore, \( \tau_r \) ranges within \( 0 \leq \tau_r < 1 \). For example, if the final value of the response has a height of \( A \) as in Fig. 5 and the residual error after the first step of the response is \( R_e \), \( \tau_r \) is defined as the ratio of \( R_e \) against \( A \), as \( \tau_r = R_e/A \). With these definitions, the transfer function of the display lag, \( D(z) \), can be written as

\[
D(z) = (1 - \tau_r)z^{-\tau_b} + \tau_rz^{-(\tau_b+1)}. \tag{6}
\]

It should be noted that \( \tau \) is not constant since the synchronization is not perfect between the execution of the program and the actual rendering on LCD. Since the relative timing of the rendering and program execution slightly shifts in every repetition, the resulting \( \tau \) also shifts. However, the maximum amount of the shift is almost limited by 1, since the function \( \text{wglSwapIntervalEXT}() \) adjusts the execution timing of the program. In Fig. 5, the first response, at around 0.1 seconds, has \( \tau \) of around 3.2. After one transient cycle, the third and the forth responses, at around 0.4 and 0.6 seconds, have \( \tau \) of
4. Performance comparison of stable algorithms

4.1. PD control

This section discusses PD control under existence of the display lag, as shown in Fig. 3(c). The basic algorithm expressed in Eqs. (2) through (4) and in Fig. 3(a) is a special case of the PD control, and thus is also analyzed here.

In the environment with the lag, the actual marker position is different from the commanded marker position due to the lag. Therefore, we discriminate them using different variables. Let $X(z)$ and $M(z)$ denote Z-transforms of the position of the target object ($x(n)$) and the actual position of the marker ($m(n)$), respectively. Their relative position, or positional error, is represented using $E(z)$. The commanded marker position is represented by $Y(z)$, which is also used as the output of the system (note that $M(z)$ is not available in the tracking program). Their relations are expressed as:

$$M(z) = D(z)Y(z),$$  \hspace{1cm} (7)
$$E(z) = X(z) - M(z),$$ \hspace{1cm} (8)
$$C(z) = K_p + K_d(1 - z^{-1}),$$ \hspace{1cm} (9)
$$Y(z) = z^{-1}(Y(z) + C(z)E(z)).$$ \hspace{1cm} (10)

where $K_p$ and $K_d$ are the proportional and derivative gains. The total transfer function, $G_{pd}(z)$, from the object position $X(z)$ to the output $Y(z)$, is

$$G_{pd}(z) = \frac{(K_p + K_d)z^{-1} - K_dz^{-2}}{1 + z^{-1}((K_p + K_d)D(z) - 1) - z^{-2}K_dD(z)}.$$ \hspace{1cm} (11)

Although the delay $\tau$ fluctuates in the actual systems, the fluctuation is ignored here; it will be discussed later in the experimental section.

Fig. 3(c) is the corresponding block diagram. Since the PD controller output gives the increment of the commanded marker position, it is naturally integrated as described in the block diagram. In that sense, if we regard the integrator as a part of the controller, the controller seems as a PI controller. Similar discussion can be made for the other algorithms, since the common underlying concept of the marker-position tracking is to calculate the increment to update the marker position.

Fig. 6 shows some responses of $G_{pd}(z)$ against a unit-step input for several different $\tau$; the basic algorithm in Fig. 3(a) was chosen as examples ($K_p = 1$ and $K_d = 0$). When the delay is less than one, the system behavior is kept stable. However, any $\tau$ larger than one results in instability. This shows that the display lag must be smaller than the execution cycle of the program, if the basic algorithm is utilized.

For more general P control case ($K_d = 0$), the stable region was numerically calculated for $\tau$ from 0 to 10. The result is shown in Fig. 7, which indicates that larger $\tau$ requires smaller $K_p$ for system stabilization. Although the stability border seems to form a single analytical curve, the analytical solutions are different for different $\tau_p$. As the analytical solutions for larger $\tau_p$ are considerably complicated (due to high orders of the resulting transfer functions), the stability area was obtained numerically.
Fig. 7 shows that the system can be stabilized even for large $\tau$ by using small gains. However, the use of smaller gains can deteriorate marker tracking performance. To reveal the performance, two tracking errors against ramp motion are evaluated: steady-state error and maximum transient error. The errors are evaluated in terms of the positional error $e[n] = x[n] - m[n]$, not the output error $x[n] - y[n]$, because $e[n]$ dominates the maximum tracking speed; if the error $e[n]$ exceeds the half of the marker width, the PT cannot detect the marker anymore. Therefore, the speed must be limited such that $e[n]$ is always kept smaller than the half marker width.

The steady-state error against a unit ramp ($R(z) = z/(z-1)^2$) can be analytically obtained using final value theorem for $Z$ transform:

$$\lim_{z \to 1} \left\{ R(z) - R(z)G_p(z)D(z) \right\} (z - 1) = \frac{1}{K_p}. \quad (12)$$

This shows that the steady-state error is simply inverse of the gain $K_p$; the differential gain $K_d$ does not affect the steady-state error. The maximum transient error was numerically evaluated using a finite-time ramp input as shown in Fig. 8. The input is a unit ramp for the first 50 samples and then constant after the 51st sample. The obtained maximum errors are shown in the stability region map of Fig. 7. The color in the stable region represents the maximum transient error.

The effect of $K_d$ gain was also investigated as in Fig. 8. It can be seen that, when $K_d$ gain is set, the oscillation damps earlier, as can be easily imagined. The maximum transient errors, as well as the stability limits, are numerically evaluated and are shown in Fig. 9. When the lag is large, having an appropriate $K_d$ can slightly enhance the stability limit for $K_p$. For example, when $\tau$ is 4.0, the maximum stable $K_p$ is about 0.37 for $K_d = 0.3$, whereas it is only 0.33 for $K_d = 0$, which means the differential action can slightly reduce steady-state error (as the steady-state error is the inverse of $K_p$). However, for smaller $\tau$, the effect of $K_d$ on the stability can be barely found; for example, $K_d \approx 0$ results in the largest stable $K_p$ when $\tau = 0.5$. These results indicate that the differential action hardly contributes to the stability or the errors; it simply damps the oscillation, as shown in Fig. 8, such that the response converges earlier.

Fig. 10 shows responses against different input motions, for better understanding of the system behavior. In plot (a), the input motion has initial acceleration, which is followed by a unit ramp and ending deceleration. Compared with Fig. 8 (b), the maximum transient error is considerably decreased due to the slow acceleration. On contrary, the steady-state errors are the same in both responses. This comparison indicates that, when the object is accelerating slowly, the steady-state error will dominate the system performance. In plot (b), a ramp motion is flipped at the middle. In this case, the vibration amplitude is doubled in the backward motion since the change of velocity is twice as large as the initial velocity change. This doubled amplitude gives rise to the maximum transient error during the backward motion, but the steady-state error remains the same.
4.2. Lag compensation

The above has discussed a simple PD control. Here, we would like to discuss another simple algorithm. The concept is that if we know the display lag τ, we would be able to estimate the current marker position, which can be used to update the position command and may lead to better tracking performance.

From the measurement such as in Fig. 4, we can estimate the display lag τ. Although the actual τ fluctuates, here we consider a constant value as an estimated lag, which is denoted using ˜τ. With this estimated lag, the display response can be estimated as

\[
\tilde{D}(z) = (1 - \tilde{\tau})z^{-\tilde{\tau}} + \tilde{\tau}z^{-(\tilde{\tau}+1)},
\]

where \(\tilde{\tau}_b\) and \(\tilde{\tau}_r\) are the integer and fractional portions of \(\tilde{\tau}\), respectively.

In the method proposed here, the current marker position is estimated using \(\tilde{D}(z)\) and is utilized to update the marker position command, \(Y(z)\). The system equations are

\[
M(z) = D(z)Y(z),
\]
\[
E(z) = X(z) - M(z),
\]
\[
\hat{M}(z) = \tilde{D}(z)Y(z), \quad \text{and}
\]
\[
\hat{Y}(z) = z^{-1} \left( \hat{M}(z) + E(z) \right),
\]

where \(\hat{M}(z)\) is estimated current marker position. By adding the error \(E(z)\) to the estimated marker position, the position command \(Y(z)\) is determined. The system transfer function, \(G_{lc}(z)\), from the input \(X(z)\) to the output \(Y(z)\) becomes

\[
G_{lc}(z) = \frac{z^{-1}}{1 + z^{-1} (D(z) - \hat{D}(z))}.
\]

Fig. 11(a) shows the block diagram of this algorithm. This block diagram can be converted into the one in Fig. 11(b). In this representation, the feedback element \(D(z)\) is regarded as a part of the controlled plant, as well as the actual marker
position $M(z)$ being the system output. This diagram shows that the lag compensation described here is equivalent to internal model control (IMC) (Garcia and Morari 1982). Typically, a controller in IMC, which is $Q(z)$ in this particular diagram, is designed as the inverse of (the invertible part of) the estimated plant model (Morari 1983). As the estimated plant model $\tilde{D}(z)$ is a time-delay with unity gain, the controller $Q(z)$ is simply given as $Q(z) = 1$.

For this system, the stability and the maximum tracking error against the finite-time ramp input was numerically calculated and plotted in Fig. 12. Interestingly, the stability region does not form a single continuous area; there are some small isolated areas. As the display lag, $\tau$, has fluctuation width of 1 in the actual systems, the stable region should have vertical width larger than 1 for stable tracking. As the small isolated areas do not have enough vertical width, they cannot be utilized in actual implementations.

For the main area along $\tau = \bar{\tau}$, the vertical width exceeds 1 and thus can stabilize the system under fluctuation of the display lag. Relatively wider vertical width is obtained when $\tau = k + 0.5$ with $k$ being positive integer numbers. For example, while $\bar{\tau}$ of 4.0 can stabilize within $[3.5, 4.5]$, $\bar{\tau} = 4.5$ covers much wider range of $\tau$, i.e., $[3.3, 5.5]$.

The steady-state error against a unit ramp for the lag compensation method is calculated from the final value theorem as

$$ \lim_{z \to 1} (R(z) - R(z)G_{lc}(z)D(z)) (z - 1) = \bar{\tau} + 1. $$

(19)

It simply depends on the estimated lag.

In Fig. 13 (a), (c), and (d), several responses are plotted for the lag compensation algorithm. The display lag $\tau$ is assumed to be constant. In lag compensation, the error is almost constant and does not oscillate as in PD control cases. Especially when the lag estimation is perfect, the maximum transient error becomes the same as the steady-state error, as shown in (a). When the lag is underestimated as in (c), there is a slight overshoot in the error plot, meaning that the maximum transient error is larger than the steady-state error. On the other hand, when the lag is overestimated as in (d), the steady-state error defines the maximum error.

Fig. 9: Stable regions and maximum transient errors in PD control under different delays. Calculation intervals are 0.01 for $K_p$ and 0.005 for $K_d$.

Fig. 10: Responses of PD control against various input motions.

Fig. 11: Responses of PD control against various input motions.
4.3. Lag compensation with PD control

PD control can also be applied to the lag compensation method. As lag compensation can virtually create a situation like $\tau < 1$ in Fig. 7, $K_p$ larger than 1 may be used to improve the tracking performance ($K_p$ smaller than 1 is not useful as it simply increases the error). As discussed above, differential action hardly contributes to the stability or the maximum error especially when $\tau < 1$. Therefore, the following mainly focus on the proportional feedback.

The block diagram for this method is illustrated in Fig. 11(c). The error, $E(z)$ read from the AD converter is fed into PD controller to calculate the increment of the marker position. The equation to update $Y(z)$ is given as,

$$Y(z) = z^{-1} (\tilde{M}(z) + C(z)E(z)), \quad \text{and}$$

$$C(z) = K_p + K_d (1 - z^{-1}), \quad \text{(21)}$$

and the system transfer function from the input $X(z)$ to the output $Y(z)$ becomes

$$G_{PD}(z) = \frac{(K_p + K_d) z^{-1} - K_d z^{-2}}{1 + z^{-1} ((K_p + K_d)D(z) - \tilde{D}(z)) - z^{-2} K_d D(z)}. \quad \text{(22)}$$

Fig. 14 shows the numerically calculated stable region and the maximum transient error for $K_p$ being 1.5 and 2.0, and $K_d = 0$. Compared with Fig. 12, improvement on the maximum transient error can be hardly observed, while the stable region becomes much narrower. For $K_p = 2.0$, the vertical width is smaller than 1 at any $\tau$, which means the actual system cannot be stabilized anymore. For $K_p = 1.5$, the vertical width is larger than 1 at some $\tau$. However, they do not cover the whole range of $\tau$; to apply $K_p = 1.5$, $\tau$ needs to be in a certain range. (For example, $\tau = [3.5, 4.5]$ can be stabilized with $\tau = 4.2$, but there is no stable $\tau$ for $\tau = [3.0, 4.0]$.) Therefore, there is a strict limitation for the additional use of PD controller, which is thus not practical.

The steady-state error for this case is

$$\lim_{z \to 1} (R(z) - R(z)G_{PD}(z)D(z)) (z - 1) = \frac{\tau + 1}{K_p}, \quad \text{(23)}$$
Fig. 13  Ramp responses of lag compensation.

which indicates the steady-state error can be improved by using larger $K_p$.

A typical response for the proportional feedback is plotted in Fig. 13 (b). Compared with Fig. 13 (a), which has the same $\tau$ and $\bar{\tau}$, the steady-state error is lowered to $1/K_p$, whereas the maximum transient errors are almost the same.

4.4. Discussions

In the above, two different classes of stable algorithms have been discussed: PD control and lag compensation using estimated lag (their combination was found non-practical). Figs. 8 and 13 compare their responses under the same lag ($\tau = 2.0$). From the comparison, the followings can be said for the two algorithms:

- PD control performs better in terms of steady-state error,
- Lag compensation performs slightly better in terms of maximum transient error, and
- Lag compensation provides less-oscillated responses. (PD control also shows less-oscillated responses when smaller gains are used, but they result in larger errors.)

Fig. 14  Stable regions and maximum transient errors in lag compensation with additional P control.
This comparison roughly provides a guideline for algorithm selection. For applications mainly with quick short-distance motions, lag compensation might perform better, since the initial transient response would dominate the overall performance. On the other hand, for applications with long-distance motions, PD control might be preferred as it shows smaller steady-state error.

5. Experiments

5.1. Experimental setup

Fig. 15 shows the experimental setup. A notebook PC (Sony VAIO SVS1512AJA, CPU: Intel Core i5-3320M, 2.6 GHz) with Windows 8.1 Pro 64 bit operating system has two different operation modes, stamina and speed modes. The PC is equipped with two different graphic engines, Intel HD graphics 4000 and NVIDIA GeForce GT 640M LE GPU, which are switched depending on the operation modes (speed mode uses NVIDIA and stamina uses Intel). The tracking program was built on Microsoft Visual C++ 2010 Express using libraries of OpenGL, glut, and wglext.h. The execution of the program was synchronized to the vsync signal using wglSwapIntervalEXT(). For the signal communications, NI USB-6341 (National Instruments) was connected to the PC, which provided AD/DA converters and digital in/out.

The LCD was BenQ FP71G+ (17 inch), which was connected to the PC through an analog VGA port. The LCD has a native resolution of 1280x1024 pixels. It runs at 60 Hz in progressive mode and the resulting refresh cycle time is about 17 ms. Horizontal screen width is about 34 cm and the pixel pitch is 264 μm. The marker displayed on the LCD screen had a width of 130 pixels (34 mm) and the grayscale has the minimum and the maximum brightness of 0.4 and 1.0, respectively.

The target object was made using a 3D-printer and was installed on a linear guide that was placed on the LCD display to allow horizontal 1-DOF motion. On the bottom surface of the target, four photo transistors were installed. As this work focuses on 1-DOF horizontal motion, only two of the four photo-transistors (top and bottom) were actually used. The outputs of photo-transistors were fed to the AD converter to be read by the program. In the tracking experiments described in the latter half of this section, a laser displacement sensor (Omron, ZX-LD100) was utilized as a reference for the target position.

Before the experiments, the readings of PT were calibrated. Fig. 16 (a) shows the output of the PT circuits against the commanded brightness. As the lower brightness region is distorted, the calibration was done for [0.4, 1.0] of commanded brightness, as shown in Fig. 16 (b). With this calibration taken into account, the lags τ for the two operation modes were identified as approximately [2.2, 3.2] for stamina and [4.3, 5.3] for speed mode. In the lag identification process, irregular lags were sometimes observed (probably because Windows is not real-time OS). However, we ignored those irregular lags.

5.2. Stability

Stability of the two algorithms were examined in the two operation modes. The target object was rested on the LCD and the marker was initially rendered just beneath the target. Then, the tracking results were recorded while no input
motion was given. Even though there was no input motion, the system can oscillate due to input noise if the system has instability.

The results for PD control in stamina mode are plotted in Fig. 17. As the mode has \( \tau = [2.2, 3.2] \), the stability limit for the gain \( K_p \) changes within \([0.42, 0.59]\), according to Fig. 7. Therefore, if \( K_p \) is less than 0.42, the system is always stable, regardless of the fluctuation of \( \tau \). On contrary, the system will be occasionally unstable if \( K_p \) is within \([0.42, 0.59]\). In the measured results, \( K_p = 0.4 \) resulted in stable response, as the analyses predicted. For \( K_p = 0.45 \), the output showed periodical bursts, due to the occasional instability. When \( K_p \) was 0.5 and 0.55, the system heavily oscillated, which sometimes resulted in marker lost. Similar results were obtained for speed mode, which are plotted in Fig. 18. The stability limits for \( \tau = 5.3 \) is approximately \( K_p = 0.27 \) and for \( \tau = 4.3 \) is 0.32. The gain \( K_p \) of 0.25, which is always in the stable region, showed stable output, whereas \( K_p \) of 0.28 showed periodic bursts. These results confirm that the system is stable when the \( K_p \) gain is smaller than the stability limit for the largest \( \tau \), which supports the analyses shown in Fig. 7.

Similar experiments were carried out for lag compensation. Additional PD controller was not used (which means \( K_p = 1 \) and \( K_d = 0 \)). According to the analyses, \( \tau \) needs to be selected such that the corresponding stable region covers the fluctuation of \( \tau \). It is found from Fig. 12 that, for stamina mode, \( \tau \) around 2.5 can cover \( \tau \) of \([2.2, 3.2]\), and for speed mode, \( \tau \) of around 4.6 can cover \( \tau \) of \([4.3, 5.3]\). The results plotted in Fig. 19 show that the system was stable for those \( \tau \). However, it was found that even when the fluctuation of \( \tau \) is not completely within the stable region, the system seemed stable at some \( \tau \). In that sense, the stable region map shown in Fig. 12 would be slightly conservative. However, the map can be a good guideline for selecting \( \tau \).
5.3. Tracking behaviors

Next, the target was moved manually along the linear guide to evaluate the tracking performance. As a reference, the laser displacement sensor measured the position of the target at the same time. Fig. 20 compares the performance of PD control and lag compensation in two operation modes. In these measurements, the target was manually moved back-and-forth with a stroke of approximately 10 to 20 mm. Although PD control utilized differential action such that the response converges earlier, the responses were still quite oscillatory. On the other hand, the lag compensation showed almost perfect tracking, although it sometimes showed a small overshoot (probably due to $\tau$ fluctuation). These results are in accordance with the simulations in section 4.

Fig. 20 compares the responses against long-distance motions. In this figure, a response of PD control and that of lag compensation are shown in the same plot for comparison. Even though the input motions, which are shown as laser sensor output, are almost the same for the two algorithms, lag compensation lost the marker at the middle of the motions. This would be due to the larger steady-state error in lag compensation, as discussed in section 4. Although the difference in the maximum tracking speeds was found not so large, PD control slightly outperformed lag compensation in long-distance motions.

These results generally support the guideline provided in section 4.4. For short-distance motions as in Fig. 20, lag
Fig. 21 Tracking results for long distance motions.

compensation would show much better tracking. For long-distance motions, PD control can track higher speed motion as in Fig. 21, although the difference in their maximum speeds was not very large.

6. Conclusion

Displayed-marker tracking is a powerful position detection method that does not require any modifications to the displays. The tracking method is expected to be utilized in many different application areas including physical CHI and augmented reality. However, the tracking method suffers from an instability problem if there is any lag in the display.

This work discussed the stability issue by investigating the characteristics of the lag and by analyzing two different classes of stable tracking algorithms. The two algorithms, PD control and lag compensation, have stable parameter regions depending on the lag. This work provided the stable region maps, such that one can determine the appropriate parameters once the lag is identified. In lag identification, however, one important thing should be noted; the lag is not constant and can fluctuate. Therefore, when choosing parameters, the fluctuation of the lag should be taken into account.

The numerical and experimental comparisons of the two stable algorithms revealed that PD control generally shows larger transient errors, whereas lag compensation shows relatively larger steady-state errors against ramp input. This led to a guideline that lag compensation would suit for short-distance motions and PD control would be better for long-distance motions.

In this paper, the discussions were limited to 1-DOF motions, but they can be easily extended to planar 2-DOF. However, in our measurements of the display lag, it was confirmed that the lag of the LCD depends on vertical position. This is because the display is refreshed from the top toward the bottom, as visually shown in (RenderingPipeline online). Therefore, lower parts have larger lag than the upper parts. This position dependency should also be taken into account when designing 2-DOF motion tracking systems.

Acknowledgement

This work was supported in part by Grand-in-Aid for Scientific Research (B) (No. 26280069) from JSPS, Japan. The authors express their appreciations to Mr. S. Kondo for his contribution in the earlier development of this work. The authors also appreciate the anonymous reviewers for their valuable comments. Especially, it should be mentioned that the discussion on IMC in section 4.2 was motivated by a suggestion from one of the reviewers.

References

Echtler, F. and Kaltenbrunner, M., SUR40 Linux: Reanimating an Obsolete Tangible Interaction Platform, Proceedings of 2016 ACM on Interactive Surfaces and Spaces (2016), pp. 343-348.
Garcia, CE. and Morari, M., Internal Model Control. 1. A Unifying Review and Some New Results, Ind. Eng. Chem. Process Des. Dev. vol. 21 (1982), pp. 308-323.
Ivkovic, Z., Stavness, I., Cutwin, C. and Sutcliffe, S., Quantifying and Mitigating the Negative Effects of Local Latencies

[DOI: 10.1299/jamdsm.2018jamdsm0125] © 2018 The Japan Society of Mechanical Engineers
on Aiming in 3D Shooter Games, Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI ’15) (2015), pp. 135-144.

Kawamoto, U., Kurata, T., Sakata, N., Okuma, T. and Kuzuoka, H., Position/Orientation-Aware Physical Tags Using Photo Sensors and Accelerometers for a Tangible Tabletop Interface, Proceedings of 2008 IEEE International Conference on Distributed Human-Machine Systems (2008), pp. 397-404.

Khronos OpenGL Registry, WGL_EXT_swap_control (online), available from [https://www.khronos.org/registry/OpenGL/registry/EXT/WGL_EXT_swap_control.txt], (accessed on Jun. 29, 2018).

Kojima, M., Sugimoto, M., Nakamura, A., Tomita, M., Nii, H. and Inami, M., Augmented Coliseum: An Augmented Game Environment with Small Vehicles, Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems (2006), pp. 3-8.

Lee,JC., Dietz, PH., Maynes-Aminzade, D., Raskar, R. and Hudson, S., Automatic Projector Calibration with Embedded Light Sensors, Proceedings of ACM User Interface Software and Technology (UIST ’04) (2004), pp. 123-126.

Lee, JC., Hudson, SE., Summer, JW. and Dietz, PH., Moveable Interactive Projected Displays Using Projector Based Tracking, Proceedings of ACM User Interface Software and Technology (UIST) 2005 (2005) pp. 63-72.

Mi, H., Krzywinski, A., Fujita, T. and Sugimoto, M., RoboTable: An Infrastructure for Intuitive Interaction with Mobile Robots in a Mixed-Reality Environment, Advances in Human-Computer Interaction (2012), Article ID 301608.

Morari, M., Internal Model Control - Theory and Applications, IFAC Proceedings Volumes, vol. 16, No. 21 (1983), pp. 1-18.

Nakamura, T. and Yamamoto, A., A Multi-user Surface Visuo-Haptic Display Using Electrostatic Friction Modulation and Capacitive-type Position Sensing, IEEE Transactions on Haptics, vol. 9, issue 3, (2016), pp. 311-322.

Noma, H., Yoshida, S., Yanagida, Y. and Tetsutani, N., The Proactive Desk: A New Haptic Display System for a Digital Desk Using a 2-DOF Linear Induction Motor, PRESENCE, vol. 13, No. 2 (2004), p. 146-163.

Pangaro, G., Maynes-Aminzade, D. and Ishii, H., The actuated workbench: computer-controlled actuation in tabletop tangible interfaces, Proceedings of ACM Symposium on User Interface Software and Technology (UIST ’02) (2002), pp. 181-190.

Portillo-Rodriguez, O., Avizzano, C., Gonzalez, OS., Vilchis-Gonzalez, A., Davila-Vilchis, M. and Bergamasco, M., Training Motor Skills Using Haptic Interfaces, Haptics Rendering and Applications, InTech (2012), p.195-214.

Raskar, R., Beardsley, P., van Baar, J., Wang, Y., Dietz, P., Lee, J., Leigh, D. and Willawcher, T., RFIG Lamps: Interacting with a Self-Describing World via Photosensing Wireless Tags and Projectors, Proceedings of ACM SIGGRAPH ’04, 2004;23, (2004), pp.406-415.

RenderingPipeline, Measuring Input Latency (online), available from [http://renderingpipeline.com/2013/09/measuring-input-latency/], (accessed on Jun. 29, 2018).

Saga, S. and Deguchi, K., Lateral-force-based 2.5-dimensional tactile display for touch screen, Proceedings of 2012 IEEE Haptics Symposium (2012), p. 15-22.

Simmons, A., Factors Affecting PC Monitor Responsiveness (online), available from [https://pcmonitors.info/articles/factors-affecting-pc-monitor-responsiveness/], (accessed on Jun. 29, 2018).

Sugimoto, M., Kodama, K., Nakamura, A., Kojima, M. and Inami, M., A Display-Based Tracking System: Display-Based Computing for Measurement Systems, Proceedings of 17th International Conference on Artificial Reality and Telexistence (2007), pp. 31-38.

Summet, J. and Sukthankar, R., Tracking Locations of Moving Hand-held Displays Using Projected Light, Proceedings of International Conference on Pervasive Computing (Pervasive 2005) (2005), pp. 37-46.

Ullmer, B. and Ishii, H., The metaDESK: models and prototypes for tangible user interfaces, Proceedings of ACM Symposium on User Interface Software and Technology (UIST ’97) (1997), pp. 223-232.

Weiss, M., Schwarz, F., Jakubowski, S., Borchers, J., Madgets: actuating widgets on interactive tabletops, Proceedings of ACM Symposium on User Interface Software and Technology (UIST ’10) (2010), pp. 293-302.