On Computer Mouse Pointing Model Online Identification and Endpoint Prediction

Anatolii Khalin, Rosane Ushirobira, Denis Efimov, Géry Casiez

On Computer Mouse Pointing Model Online Identification and Endpoint Prediction

Anatolii Khalin, Rosane Ushirobira, Denis Efimov, Géry Casiez

***Abstract*—This paper proposes a new simplified pointing model as a feedback-based dynamical system, including both human and computer sides of the process. It takes into account the commutation between the correction and ballistic phases in pointing tasks. We use the mouse position increment signal from noisy experimental data to achieve our main objectives: to estimate the model parameters online and predict the task endpoint. Some estimation tools and validation results, applying linear regression techniques on the experimental data are presented. We also compare with a similar prediction algorithm to show the potential of our algorithm's implementation.**

I. Introduction

COMPUTER and information technologies conquered a big part of our everyday lives, and in this way, the optimization, in a smooth and pleasurable manner, of Human-Computer Interaction (HCI) processes has become an important issue. There are different approaches to analyzing and designing HCI algorithms, and some of them are based on the control theory. According to [1], it allows the user and computer systems to be linked by describing the communication between the parts and the process dynamics as a whole. In such an approach, the human-computer interaction can be presented as a feedback interconnection. An experimental comparison of different methods demonstrates that control theory applications are up-and-coming in the HCI domain [2]. Most applications of control theory in HCI so far are directed to pointing tasks.

Pointing is a fundamental task, and it has been a research subject for many years in the HCI field. A pointing task consists of guiding some device in the control (also called motor) space for navigating the cursor in the operating space (a display). There are some papers about hardware optimization (for instance, a recent study about the optimal sensor position in the mouse [3]), but the majority of papers in HCI with regards to pointing concern the so-called interaction techniques [4], [5], [6], [7]. Despite the development of new devices, touchpads and computer mice are the most popular input devices for most users, and they are still widely used for such tasks as pointing, tracking, dragging, targeted-tracking [8], *etc*. That is why there are several models for direct and indirect pointing tasks available in the literature, providing different levels of the process description.

Regarding the goals of this work, the considered problem can be divided into two parts: the first one is the **modeling of a pointing task** and the second one is the endpoint prediction.

The first concept related to the modeling of a pointing task was the Fitts law [9], which states that the pointing time has a logarithmic relation to the ratio of distance and the target's width, also called Index of Difficulty (ID). The Fitts law has been experimentally validated many times [10], [11]. However, Fitts law can model pointing time in relation to the ID of the task, but it does not allow to model the dynamics of the movement. The first attempt to provide a *dynamical* pointing model was the so-called Surge model [12]. It was the first try to model the task in two modes: the constant-coefficient model and the second-order controller model, representing open-loop and closed-loop phases, correspondingly. Later, VITE (Vector Integration to Endpoint) model was introduced in [13] as a linear time-invariant system describing the motion, controlled by an agonist-antagonist pair of muscles (wrist rotation). Simultaneously, Meyer *et al.* [14] established that the pointing task contains two phases: a ballistic (supposed to be an open-loop process of the initial movement towards the target) and a correction phase (that finishes the task by closed-loop visual guidance until the target is reached). This concept was called Meyer's Optimized Initial Impulse model [14]. Another feature of this model was the possibility of modeling the multiple open-loop phases (also called sub-movements). More recently, based on this principle and the VITE model, a united switched dynamical model was created in [15], verified on experimental data of [16], and whose closed-loop stability has been proven. Another line of research was based on Todorov's optimal feedback control of motor coordination [17], [18], [19]. The main idea was to apply Stochastic Optimal Control to model a human movement [20], [21], [22]. The latest result of such an approach can be found in [23], where, besides feedback, the authors also consider feedforward control, which models the part of the process with motor planning to generate a command (before the movement begins). Another recent idea was based on the so-called Intermittent Control [24]. The key feature of the latter framework was a build-in event-based trigger, which allowed the model to switch between closed- and open-loop control depending on the error between the goal point and the one observed by the user. Such an approach appeared to be compatible with current physiological and psychological theories. A different view on modeling worth mentioning is through the information theory. In the recent work [25], the two-phase process is viewed as a communication problem where the information is transmitted from a source (initial/current point) to a destination (endpoint) over a so-called channel, perturbed by Gaussian noise with a presence of feedback (in a second phase). Despite the diversity of approaches for modeling the pointing task, we believe that searching for a simple, and at the same time, an accurate model is yet a relevant research direction.

The second problem considered in this paper with regards to pointing is the **endpoint prediction**. Indeed, it is possible

to facilitate the interaction and shorten the movement time by predicting the user's desired destination. Several papers have been published regarding endpoint prediction, and few algorithms in this area were proposed. Starting from Lank *et al.* [26], the Kinematic Endpoint Prediction (KEP) model was introduced. This model is based on the minimum jerk law, formulated in [27], and the proposed prediction relies on a polynomial curve fitting (based on a fourth-order polynomial). The model showed the best performance at 80% of the path and distances more than 600 pixels. Another idea was presented in [28], where a simple regression-based extrapolation was used, making the endpoint guess at the peak of the velocity. The prediction technique based on the inverse optimal control model was introduced in [29], where a probabilistic model and machine learning were combined. It showed better performance compared to previously proposed models, especially during the short-distance trials. The Kinematic Template Matching concept was introduced in [30]. The template matching algorithm compares the velocity profile and decides which distance the user wants to cover based on previous trials. The algorithm showed a better performance for the 2D-task than the KEP model and reasonable general accuracy.

We can formally divide all existing endpoint prediction algorithms into two groups:

group I) algorithms with knowledge about previous trials (memory), and

group II) without any knowledge (no memory).

In the first group, we can put the algorithms based on the linear regression [28], any machine learning tools [29], and the recently introduced KTM algorithm [30] because all of them require several trials of the same user or on the same setup to identify the endpoint online. The algorithms without memory, such as KEP [26], rely on more general rules and assumptions about the motion, and they can be applied from the first try.

Another innovative idea for facilitating the target acquisition for the user was presented in [31]. The authors used a target prediction approach with the eye-gaze device, and the idea was to skip the ballistic phase entirely by cursor warping (moving the pointer directly to the area of users' sight). The results of the experiments showed a significant decrease in the pointing time. In this paper, we will follow the classical pointing task setup, using only the computer mouse for our prediction algorithm, since it is the case for most users.

Despite all these achievements, the current state of the art can still be improved in many ways. For example, since all the algorithms mentioned above do not utilize a dynamical pointing model representing both sides of the process, the results are less general and less accessible for analysis. Algorithms based on machine learning or probability techniques do not explain the nature of the movement well, and their results are unreliable in unusual setups and situations.

Both problems described above are usually separated and not considered together in a single model-based prediction algorithm, which could be more advantageous for description and analysis. We fill the indicated gap in the present work, and an endpoint prediction algorithm is presented, based on a new simplified version of the dynamical model given in [15].

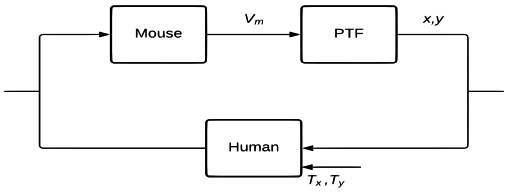

Fig. 1. Closed-loop model, conventional block diagram

The aim of simplifying the model from [15] is to improve the possibility of online parameter identification and allow the process characteristics to be adjusted or modified online. Another advantage of the presented model is its portability since we do not suppose to know the used surfaces that the computer mouse operates on, the Pointing Transfer Function (PTF) of the operating systems (which will be introduced later in the text), or other varying setup conditions. A recent adaptive observer technique [32] is used to adjust the model parameters and identify everything *on the fly*. It is suggested how to modify the PTF once the endpoint coordinates are derived. Experimental data with a computer mouse [16], [33] is applied for this work's model development and validation.

II. Problem Statement

The problem considered in this paper is modeling and parameter identification of an indirect pointing task.

The main idea here is to use the automatic control theory to design the pointing model, where the user is a controller in the closed-loop, whose behavior and decision we do not know exactly, but we can measure and estimate it indirectly and predict the desired position. The interaction and related input and output signals are presented in Fig. 1 for the x-direction displacement (there is also the y displacement in the 2-dimensional case). We suppose that the user positions the cursor on the screen having x and/or y as an input and transforms it with his decision into a wrist movement, producing the force that translates the mouse, which has the position x_m and y_m on the plane. So, the mouse output will be its velocity $V_m = \sqrt{V_{mx}^2 + V_{my}^2}$, where $\dot{x}_m = V_{mx}$ and $\dot{y}_m = V_{my}$ are the velocity projections on x and y axes, respectively. The term of mass was omitted: we suppose that the mass of a mouse and a hand on it is a unit, and all other parameters appearing in an equation are normalized by the mass. The basic equations for this process can be presented as follows:

$$\begin{aligned} \dot{x}_m(t) &= V_{mx}(t), \\ \dot{V}_{mx}(t) &= F_{\text{fr}}(t) - F_{\text{input}}(t), \\ \dot{x}(t) &= \frac{V_{mx}(t)}{V_m(t)}\ \text{PTF}(V_m(t)),\ \forall t \geq 0 \end{aligned} \tag{1}$$

where F_{fr} is the mouse friction force, and F_{input} is the force generated by the user, $\text{PTF} : \mathbb{R} \to \mathbb{R}$ is a pointing transfer function (see [16]), that transforms the mouse velocity V_m into the cursor velocity on the screen $V_c(t) = \frac{\partial}{\partial t}\sqrt{x^2(t) + y^2(t)}$, $\forall t \geq 0$ (usually, through look-up tables that realize a nonlinear function in all existing operating systems [16]), the ratio

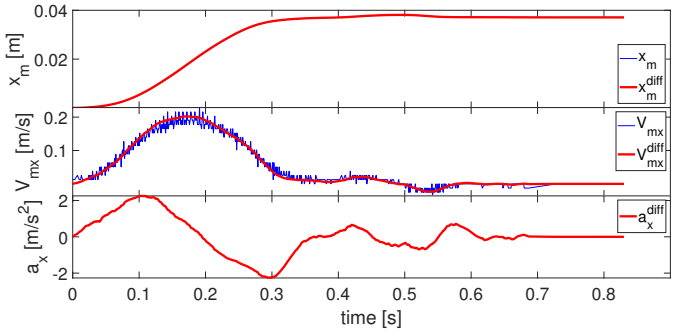

Fig. 2. Position differentiation: a) t [s] vs x_m and x_m^{diff} [m]; b) t [s] vs V_{mx} and V_{mx}^{diff} [m/s]; c) t [s] vs a_x^{diff} [s]

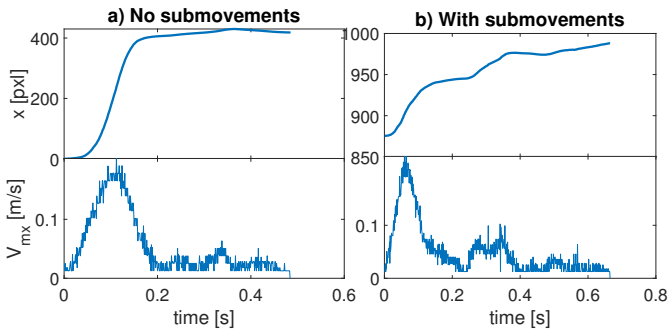

Fig. 3. Trial with and without sub-movements: a), c) x [pxl] vs time, b), d) V_{mx} [m/s] vs time

$\frac{V_{mx}}{V_m}$ is used to project the PTF effort on the corresponding movement direction. We assume that the human force F_{input} is proportional to the endpoint position T_x (in the ballistic phase) or the current error between x and T_x (in the correction phase).

Remark 1. Our idea is to estimate the decision process online, predict the reference position T_x (which we originally do not know, only the user does), and improve PTF using the estimated value $\widehat{T}_x$ of the endpoint T_x. To this end, we introduce an additional feedback in the PTF block to adjust the navigation, shorten the pointing time, and possibly reduce the error rate (usually present in all known tests). Thus, we add a correction term $\rho(x-\widehat{T}_x)$ with some tuning parameter $\rho > 0$, so the last equations in (1) takes the form:

$$\dot{x}(t) = \frac{V_{mx}(t)}{V_m(t)}\ \text{PTF}(V_m(t)) - \rho\left(x(t) - \widehat{T}_x(t)\right),\ \forall t \geq 0.$$

It is also worth mentioning that we still do not know how a user will react to such a modification of PTF, will it be more optimal or not, but we will provide the related experiments to check it in a later future. An example of how users can react to a similar approach can be found in [34], where the endpoint prediction was used to help the elderly navigate on the screen.

For 2-D tasks, the equations for the vertical components y_m, V_{my}, y, T_y, $\widehat{T}_y$ can be presented in the same way.

So the motivating technical issue of this work is to estimate the cursor's wanted position T_x and T_y as fast as possible using the dataset obtained in the experiment [33].

III. Data description and processing

According to [33], a mouse with a frequency of 1 kHz and 2000 CPI was used, and all measurements are provided at discrete instants of time t_k [s] with a varying sampling period $h^{k-1} = t_k - t_{k-1}$ for $k = 1,2,\ldots$ and $t_0 = 0$ (the sampling is not exactly 0.001 [s] due to the event-based nature of the operating system). We have an optical sensor inside the mouse, that measures the integer value increment of the mouse displacement $\Delta x_m^k = x_m^k - x_m^{k-1}$ in counts (with a measurement noise that is skipped for brevity), where $x_m^k = x_m(t_k)$, which can be converted into velocity in [m/s] for the conformity, considering the mouse CPI and inches to meters ratio $\iota = 0.0254$:

$$V_{mx}(t_k) = V_{mx}^k = \frac{\iota\ \Delta x_m^k}{\text{CPI}\ h^{k-1}},\ k = 1,2,\ldots.$$

The mouse position in [m] is obtained from the calculated velocity (scaled mouse displacement) by each time step accumulation:

$$x_m^k = x_m^{k-1} + h^{k-1} V_{mx}^k.$$

The cursor position on the screen $x^k = x(t_k)$ in [pixel] is available at the same time instants. Similarly, we have $y_m^k = y_m(t_k)$, $\Delta y_m^k = y_m^k - y_m^{k-1}$, $V_{my}(t_k)$ and $y^k = y(t_k)$.

Examples of obtained noisy signals are shown in Fig. 2 and demonstrate a bad velocity estimation due to the noise. To obtain a better velocity profile and the respective acceleration (which we need later for the mode identification) we applied a homogeneous differentiator to the mouse position (see [35], and Appendix A for the used equations). This type of differentiator's main advantage is that it possesses a finite-time convergence rate [35], which is faster than any exponential, and in addition, it is proven to be robust to the disturbances [36]. Finding a trade-off between smoothness and convergence accuracy, the parameters are tuned as

$$\lambda = 5,\ \xi = 2,\ \alpha = -0.2,$$

then $V_{mx} = \widehat{z}_2$ and $a_{mx} = \widehat{z}_3$ can be used for further parameter identification of the model. The filtered mouse position, velocity and acceleration are derived by applying the homogeneous differentiator in [35] to the mouse position presented in Fig. 2. From these plots, we can conclude that this homogeneous differentiator filters well the noise and generates rather smooth estimates of the velocity and acceleration.

Remark 2. The experiments of [33] included 10 users, which navigate the cursor on the screen using the mouse in a task with 13 targets positioned along a circle, following the norm ISO 9241-9 (§B.6.2.2). Two different distances (150 and 75 mm), target widths (7 and 2 mm), and two different PTFs were used. Depending on the velocity or position profile, it is possible to detect which user is more experienced with the setup and who is less. An experienced user makes a ballistic movement very close towards the goal, and the motion is realized in one round without sub-movements (without additional ballistic corrections). A less-experienced user needs more attempts to reach the surrounding area and may produce several sub-movements to attain the goal (see Fig. 3 for illustrations). Another way to define experienced users is according to the movement time: the lower their time on completing

the trial, the more experienced they are. In this paper, we will rely on the first definition since our prediction algorithm will be generalized to the cases with sub-movements. The performance also depends on the target width and the number of trials made with the current setup. Therefore, it makes sense to evaluate the model on the trials without additional sub-movements because when a user utilizes his/her personal setup (PC and mouse), the movements are much more precise than with an unfamiliar one. This will be done first, and next, the model presented in this paper is generalized in subsection V.*D*, considering all possible cases.

IV. Pointing Transfer Function

As was mentioned earlier, the PTF provides the gain from the mouse velocity V_m to the cursor one V_c. Every operating system has its look-up tables with standard gains, and every mouse velocity input is interpolated accordingly. The set of the final gains constitute a nonlinear function. The authors of [16] created a useful library, called Libpointing, which captures raw mouse data and allows one to create and use one's own PTF.

In this work, we assume that the shape of the PTF is unknown to develop a platform-independent approach. Therefore, it has to be reconstructed from the measured data, and our idea is that a polynomial of the mouse velocity can approximate the PTF function in (1):

$$\mathrm{PTF}(V_m(t)) = V_c(t) = \sum_{i=0}^{q} c_i V_m^i(t),\ \forall t \geq 0$$

where the nonlinear gain is a polynomial of order q with unknown coefficients $c_0, \ldots c_q$. Taking into account the discrete-time nature of the measurements, we have:

$$V_c^k = \sum_{i=0}^{q} c_i V_m^i(k),$$

where $V_c^k = V_c(t_k)$ and $V_m(k) = V_m(t_k)$. The cursor velocity can be approximated by a simple Euler formula

$$\widehat{V}_c^k = \frac{d^k - d^{k-1}}{h^{k-1}},$$

where $d^k = \sqrt{x_k^2 + y_k^2}$ is the distance on the screen from the origin corner, or by applying the homogeneous differentiator in [35] to d^k. Next, a simple linear regression estimation technique [37] can be used to estimate the parameters $c_0, \ldots, c_q$:

$$\begin{bmatrix}\widehat{c}_0 & \ldots & \widehat{c}_q\end{bmatrix}^\top = A_k^{-1} \psi^k$$

provided that the matrix A_k is not singular, where

$$A_k = \delta A_{k-1} + \omega_k \omega_k^\top,\ \psi^k = \delta \psi^{k-1} + \omega_k \widehat{V}_c^k,$$
$$\omega_k = \begin{bmatrix}V_m^0(k) & \ldots & V_m^q(k)\end{bmatrix}^\top$$

with the regression vector ω_k; ψ^k and A^k are auxiliary matrix variables, and $\delta \in (0,1]$ is a forgetting factor.

After some trials, we obtain the polynomial function of the current PTF, which can be used for modeling and prediction. The higher is the polynomial order, the more accurately the final coefficient set will represent the current PTF in future trials. So, it can be helpful to utilize the model with continuous functions, avoiding the usage of look-up tables.

V. Pointing task modeling and identification

A pointing model has to reflect both muscular and perception, human feedback following the current position on the screen. The task is to estimate the desired position of the cursor given by the constants T_x (and T_y), based on the dataset of x and V_{mx} (y and V_{my}, respectively). As usual, the real-world data is noisy, and a challenging estimation issue is that we have no good mouse velocity and acceleration measurements in the experiments. In this section, with a small ambiguity of notation, we will use the signals x_m, V_{mx} and a_{mx} gathered after applying the homogeneous differentiator, as previously explained.

Remark 3. Although we estimate the horizontal and vertical cursor position dynamics separately, the general pointing task is two-dimensional. As observed in the experimental data provided in [33], the correcting (tracking) phase usually starts approximately in about $\frac{1}{5}$ of the maximal amplitude of V_{mx} on its downhill. In the sequel, we use this rule to define the threshold for commutation between the modes.

A. Correction phase

In (1), we choose the input force as a simple linear feedback

$$F_{\text{input}} = b_2(x - T_x),$$

with a parameter $b_2 > 0$, which is unknown and can be different for each trial (such a choice is often assumed in the so-called Equilibrium-point hypothesis [38]). We can select the dynamic friction force in a usual form

$$F_{\text{fr}} = -b_1 V_{mx},$$

where $b_1 > 0$ is the friction coefficient, which is also unknown and trial-sensitive. Usually, $b_1 > 0$ is dependent on the mouse and the surface mouse operated on, but it also depends on the users' hand position, which influences the mass and makes this coefficient not constant, despite operating the same surfaces. If the parameters b_1 and b_2 are non-negative, then the resulting dynamics is stable, which corresponds to our intuition about the pointing process. Thus, estimation techniques can be applied to this model:

$$\dot{V}_{mx}(t) = -b_1 V_{mx}(t) - b_2(x(t) - T_x),\ \forall t \geq 0$$

and dividing it by b_2 to decouple the main parameter we want to estimate here, T_x, we can present this equation in a suitable form for the linear regression:

$$x(t) = -\frac{1}{b_2}\dot{V}_{mx}(t) - \frac{b_1}{b_2}V_{mx}(t) + T_x = \omega^\top(t)\theta,$$

where $\omega(t) = \begin{bmatrix}-a_{mx}(t) & -V_{mx}(t) & 1\end{bmatrix}^\top$ is the regression vector, which contains measured signals, and $\theta = \begin{bmatrix}\frac{1}{b_2} & \frac{b_1}{b_2} & T_x\end{bmatrix}^\top$ is the vector of unknown constant parameters to estimate. Next, a linear estimation technique (as for PTF previously) can be applied. However, the Persistence of Excitation (PE) condition [39] needs to be satisfied for the convergence of the parameter estimation error, which is not confirmed in our setting (the system is stable, and the external input is constant; therefore, V_x and a_{mx} approach zero asymptotically, while $x(t)$

goes to the constant T_x as t goes to infinity). Since we mainly need to estimate only one parameter, *i.e.*, the desired position T_x, then a strong excitation providing the PE property for all three parameters in θ is not needed. It is useful to separate the estimation variables for better identification. An efficient and recently introduced DREM procedure [40], [41] (see Appendix B) allows the coefficients in θ to be evaluated separately. This procedure can also provide a finite-time convergent estimation even without the satisfaction of PE condition [32] (interval excitation, which is presented in pointing tasks, is enough). In this work, we selected the auxiliary filters as delays: $H_i(s) = e^{i\tau s}$ for $i = 0,1,2$ for some basic delay $\tau > 0$, then using the steps described in Appendix B, we can get three scalar equations with separated parameters:

$$\widetilde{Y}_i(t) = \varphi(t)\theta_i + v(t),\ \forall t \geq 0 \tag{2}$$

where v is the error variable related to the measurement and differentiation noise, and φ is the new scalar regressor, which leads to the adaptation algorithm

$$\dot{\widehat{T}}_x(t) = -\gamma_3\varphi(t)(\varphi(t)\widehat{T}_x(t) - \widetilde{Y}_3(t)),$$

with $\gamma_3 > 0$ a tuning parameter and $\widehat{T}_x$ the estimate of T_x.

The noise is unknown, but we can detect when its influence is too strong and cut the prediction in corresponding regions.

A simple way to do it would be to introduce an additional condition on the estimate $\widehat{T}_x$. To reduce the big influence of the noise, we can bound $\widehat{T}_x$ under the assumption that it is supposed to be close to the endpoint after the ballistic phase. Although the screen resolution naturally bounds the endpoint position, we want to avoid using the introduced model's setup-dependent data. We can restrict our prediction estimation using only the cursor position data obtained from the current trial. Because the correction phase serves for the final navigation of the pointer to the goal, it should ideally operate around the current position value. Choosing the approximate evaluation (bound) as $\frac{1}{10}$ of the distance already traveled, we can introduce the condition as follows:

$$\left|\widehat{T}_x(t) - \widehat{T}_x(t^-)\right| < 0.1\ \ |x(t)|$$

where $\widehat{T}_x(t^-)$ denotes the estimate of T_x at the previous instant of time. If it is satisfied, we update $\widehat{T}_x$. Otherwise, we keep the previous value, cutting out the noisy region affecting the estimation and producing numerical instability.

B. Ballistic phase

In this phase, we assume that the whole movement is realized *intuitively* or *automatically* in the motor space based on the averaged PTF gain sensed by the user (*i.e.*, it is an open-loop deterministic motion), and the mouse position follows the curve:

$$x_m(t) = \frac{T_x}{2c}\left(1 - \cos(\eta t)\right),\ \forall t \geq 0,$$

where $\eta > 0$ is a user-dependent parameter, which regulates the velocity of displacement, and the movement amplitude is proportional to the goal position T_x divided by an averaged PTF gain $c > 0$. This coefficient represents the ratio between the motor and operational spaces, and for the user, it is expected that

$$x(t) = cx_m(t).$$

In such a case $x_m\left(\frac{\pi}{\eta}\right) = \frac{T_x}{c}$ and $x\left(\frac{\pi}{\eta}\right) = T_x$. Note that this gain c can be easily estimated if the PTF function is known (if it has been already identified as explained above):

$$c = V_{\max}^{-1}\int_0^{V_{\max}} \mathrm{PTF}(s)ds,$$

where $V_{\max}$ is the maximal admissible computer mouse velocity for the user (can be extracted from the data). Nevertheless, as we demonstrate below, we do not need this parameter to estimate the endpoint. Thus, if the ballistic movement is well-realized, and the human approximation of the PTF gain c is adequate, then the cursor has to fall into a vicinity of the desired position T_x. Simple calculations show that

$$\dot{x}_m(t) = V_{mx}(t) = \frac{T_x\eta}{2c}\sin(\eta t),\ \dot{V}_{mx}(t) = a_{mx}(t) = \frac{T_x\eta^2}{2c}\cos(\eta t),$$

hence,

$$\dot{V}_{mx}(t) = -\eta^2\left(x_m(t) - \frac{T_x}{2c}\right) = -\frac{\eta^2}{c}\left(x(t) - \frac{T_x}{2}\right),$$

and the obtained model can be related to the one proposed previously for the correction phase by imposing $F_{\mathrm{fr}} \equiv 0$ and $F_{\mathrm{input}} = b_2\left(x - \frac{T_x}{2}\right)$ with $b_2 = \frac{\eta^2}{c}$. Consequently, in the ballistic phase, it is assumed that the friction force can be neglected during such a fast transient and the goal position for the ballistic phase is just half away from the desired set-point T_x (this is an interpretation in terms of visual user feedback).

To estimate the goal position T_x using this model, let us observe that if for some $t' \geq 0$, we have $a_{mx}(t') = 0$ then $x(t') = \frac{T_x}{2}$. So, a basic estimation algorithm during ballistic phase is:

$$\widehat{T}_x(t) = \begin{cases} 2x(t), & \text{if } a_{mx}(t)\cdot V_{mx}(t) \geq 0 \\ \widehat{T}_x(t^-), & \text{otherwise} \end{cases},\ \forall t \geq 0,$$

where $\widehat{T}_x(t^-)$ stands for the estimated value of T_x at the previous instant of time.

The idea of doubling the position until the peak of velocity for endpoint estimation is not new. In [42], observing the experimental data, the authors concluded that it could be an option to obtain the users' goal during the movement. We derived this conclusion here from the equations presented for the ballistic phase of the switched model (thus suggesting a mathematical explanation for [42]).

C. Switched model identification algorithm

The procedure for updating the endpoint estimation $\widehat{T}_x$ using the proposed simplified switched model is divided into three modes:

a) Ballistic phase before velocity peak (before a_{mx} reaches zero). We look for the velocity V_{mx} peak during this part of the ballistic phase by comparing the current absolute value to the previous maximum. While V_{mx} grows, $\widehat{T}_x$ can be found as:

$$\widehat{T}_x(t) = 2x(t).$$

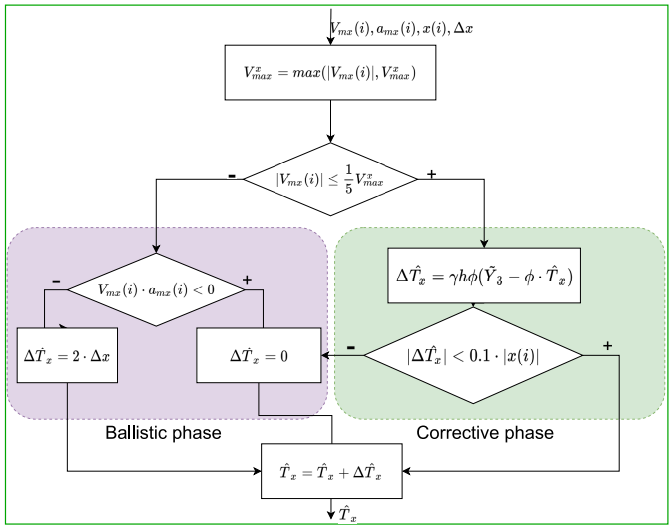

Fig. 4. Switched Model Diagram

b) Ballistic phase after velocity peak. If the current velocity V_{mx} is lower than its maximum, then $\widehat{T}_x$ is constant and defined by its previous value at the peak.

c) The correction phase. Here, we use the DREM estimation procedure as described before.

Moreover, we can generalize such an approach to the situation when the user's estimation of PTF is not adequate.

D. Prediction of trials with sub-movements

Sometimes during the trial, the user cannot reach the goal with a first try. It is a well-known problem in the pointing task. The additional ballistic phases, which appear before the goal is reached (it is easy to recognize them in the velocity profile shown in Fig. 3), are called sub-movements, and they make the endpoint prediction task much more complicated. The switching model presented in this paper allows us to change the mode at any point in time. Therefore, we can generalize the prediction algorithm presented above by considering possible sub-movements.

The generalized idea for updating the endpoint prediction is presented in Fig. 4. Basically, we check $\Delta\widehat{T}_x$ at each time step, depending on the conditions of the current value of the velocity $V_{mx}(i)$ and acceleration $a_{mx}(i)$.

First, we search for the maximum of the mouse velocity magnitude by comparing it to the current value. Second, we determine in which phase of motion we currently are. It is the same condition described in the previous section: below $\frac{1}{5}$ of the maximum velocity, we use the estimation from the model for the correction phase; otherwise, we use the ballistic prediction. If we are in the ballistic phase, we have the following condition to verify: whether the velocity is increasing with regard to the direction of the motion or decreasing (it means that we require current acceleration and velocity have the same sign to adjust the prediction by $2\Delta x$). Through this modified condition from the previous section, we also consider overshooting: when the target has already passed and the user moves the mouse in the opposite direction, $\Delta\widehat{T}_x$ is negative and our prediction is decreasing in the same way as

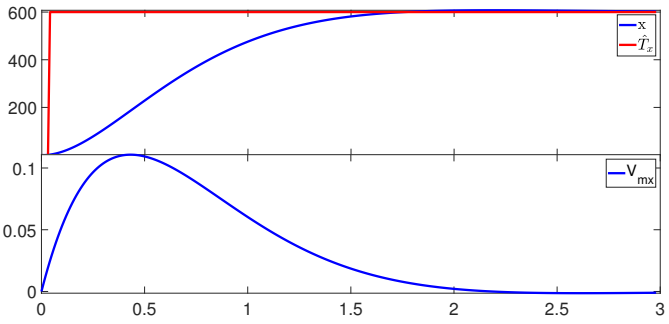

Fig. 5. Simulation of corrective phase model: a) t [s] vs x and $\widehat{T}_x$ [pxl]; b) t [s] vs V_{mx}[m/s]

with undershoot, the sign is considered. Simultaneously, we have the corrective phase condition to bound the prediction when the data is the noisiest. Since the noise itself is unknown, we cannot compensate for its influence, but we can restrict the estimation to those moments of time when the estimation is leaving the predefined boundaries. It usually signifies exactly when the measurement data is the noisiest. When it is satisfied, we update the $\widehat{T}_x$; otherwise, we keep the prediction to avoid possible numerical instabilities and make the algorithm more robust to the disturbances, as explained in subsection 5.1.

Remark 4. We have the same diagram for $\hat{T}_y$ in parallel at each timestep. It is possible that according to the V_{mx} velocity profile, we are still in the ballistic phase, while according to V_{my}, we are in the corrective phase and *vice versa*. We assume that this situation is possible, based on the observation of the 2-D cursor trajectories from the experimental data.

The generalized algorithm cannot estimate the final desired value in the case of sub-movements after the first velocity peak. However, we can estimate the sub-movement final value earlier by merely accumulating new estimates with the first peak value.

Let us demonstrate the efficiency of the proposed model and the estimation approach using real data and simulations.

VI. Results and Simulations

A. Validation and identification of our pointing model

First, the chosen model in the correction phase was evaluated in the simulation, where the model parameters were assigned artificially: $b_1 = 4$, $b_2 = 7$, $T_x = 600$ pxl and PTF= arctan (it can be any bounded function according to [15]). The model identification results are presented in Fig. 5, the final position and other coefficients are estimated almost simultaneously with the constant time step $h = 0.01$ [s], meaning that the proposed estimation method works for the designed model.

Next, real data from experiments in [33] were used for the switched model validation.

First, we try to avoid the division into two phases and represent the movement via the correction phase model. However, the estimation results given in Fig. 6 show that the high-velocity part (the ballistic phase) of the process does not fully correspond to the chosen model with the experimental data, and it is hard to estimate the value of T_x early enough. An

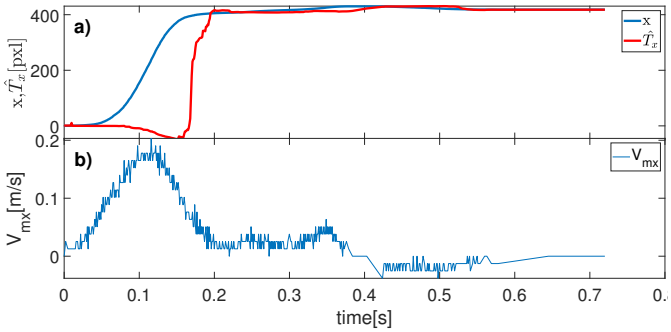

Fig. 6. Correction phase model estimation: a) t[s] vs x and $\widehat{T}_x$ [pxl]; b) t [s] vs V_{mx} [m/s]

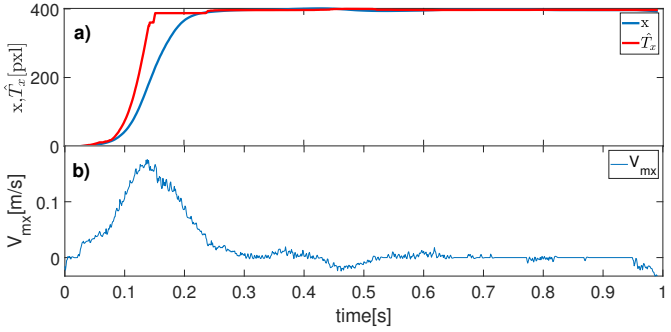

Fig. 7. Switched model estimation for one trial: a) t [s] vs x and $\widehat{T}_x$ [pxl]; b) t [s] vs V_{mx} [m/s]

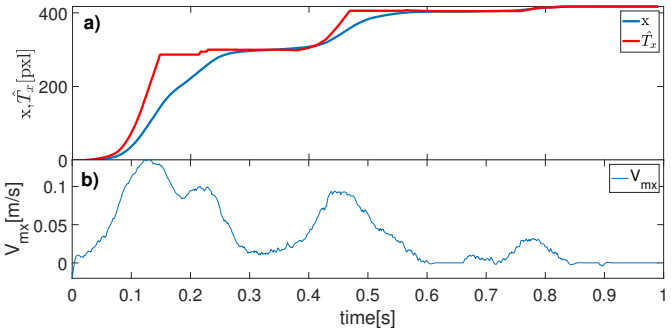

Fig. 8. Switched model estimation for one trial: a) t [s] vs x and $\widehat{T}_x$ [pxl]; b) t [s] vs V_{mx} [m/s]

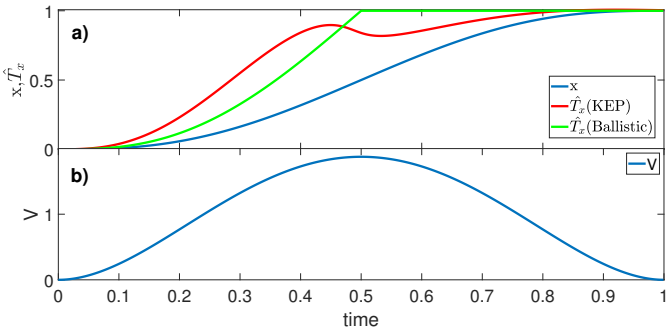

Fig. 9. Hogan's law theoretical model (unitless): a) t vs x and $\widehat{T}_x$; b) t vs V

explanation for this phenomenon is that there might be some nonlinear terms missed in the model (1) for the forces $F_{\text{fr}}(t)$ and $F_{\text{input}}(t)$, which leads to high modeling error and inaccurate identification during the ballistic phase. Investigating such a missed term can be a task for future model improvements. Still, as one can see in Fig. 6, the model represents well the process in the corrective phase.

After analyzing the performance of the reduced model, we would like to return to the classic division in two phases, where we use the complete designed model with separate equations for the ballistic phase until the motion is close to the correction phase. Such a model demonstrates a better overall performance (check Fig. 7), and it guesses the destination point on the peak of the velocity profile rather accurately, but only when the user has enough experience. When the sub-movements appear, the algorithm recognizes them as explained in Subsection 5.4 and gives a fair prediction reacting to them in the same way as to the main peak (check Fig. 8). However, to validate the model performance more concretely, let us compare it with another algorithm.

B. Comparison with existing prediction algorithms

As it was mentioned in the introduction, prediction algorithms can be divided into two groups (with or without prior knowledge). Both groups have their pros and cons. For instance, algorithms with foreknowledge can be more precise on average but require the user to complete all the range of motion, usually taking some time and memory. Moreover, the result can be unpredictable under unknown circumstances. Concurrently, algorithms without memory can produce immediate results, which may be less accurate but more stable at any setup and unpredictable situation.

The problem is then to find the trade-off between the final accuracy and robustness of the prediction. The paper [30] shows that the algorithm with foreknowledge does not improve the estimation significantly, even when theoretically, this should be a significant advantage.

This paper introduces an algorithm belonging to the second group and fits the theoretical and mechanical model presented above. For this reason, in this section, we make a comparison with the KEP algorithm to show our model's performance using the same data. It is worth to mention that the KEP algorithm was improved later in [43], the newer version became more stable in terms of estimation (the stability check was added), and the prediction is started in 85% of the movement, which improved the accuracy of the final prediction but removed information about early prediction. Later, the single point prediction, based on the KEP algorithm was introduced in [44], but since we are interested in the early prediction - a single point and the improved model was not considered in the comparison, only the stability condition from the improved model was applied. Apart from this, we reconstructed the KEP algorithm from the original paper [26], applying it to every step of the motion.

The interesting fact is that the ballistic phase prediction also satisfies the Hogan Minimal Jerk model [27] theoretical data, used as a basis for the KEP algorithm. It is easy to verify that the amplitude of the velocity in the equation is precisely the doubled position at the time of the maximum velocity

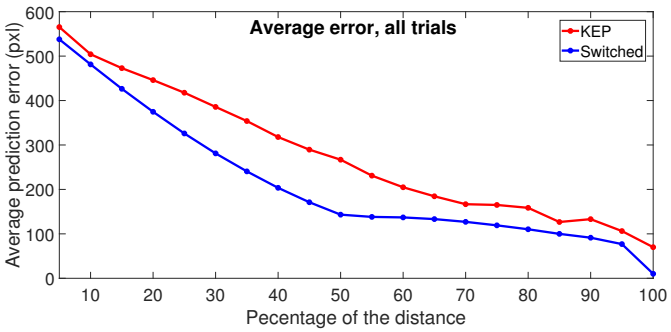

Fig. 10. Comparison: percentage of the path vs error [pxl]. All trials

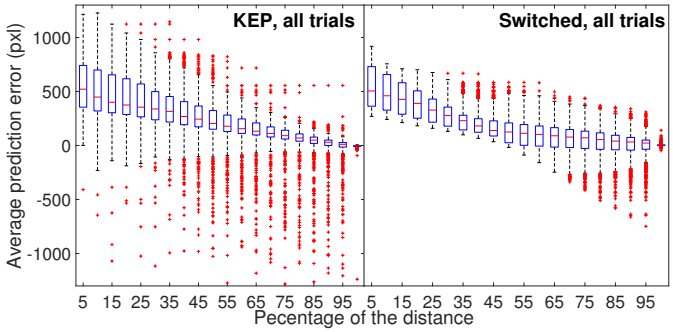

Fig. 11. Box-whisker plot comparison: percentage of the path vs error distribution [pxl]

$(x(1) = 2x(\frac{1}{2}),\ t \in [0,1]$ in Eqn. 8 of [26]). Therefore, if we apply our ballistic algorithm to the Hogan equations, we obtain the earlier exact prediction halfway on the velocity peak (Fig. 9). It is another reason why we think that our algorithm should outperform the KEP.

Since users were not used to the setup, most of the trials in the experiments [33] had sub-movements. It is important to mention that by trials without sub-movements, we mean the ones where the highest peak of the velocity is at least 5 times higher than all the others, and peaks below this threshold are not considered as sub-movements due to the low impact on the prediction, the example can be found in Fig. 3. Since our prediction algorithm considers trials with sub-movements in Fig. 10, we then chose all the experiment trials (960).

The way of evaluating the model was as follows: we had 960 trials from users, performing pointing tasks in 2-D space, but since the estimation in our model is separate for $\widehat{T}_x$ and $\widehat{T}_y$, we took one set of data from each trial, the vertical or the horizontal (x , y), depending on which distance (T_x or T_y) was longer. This choice was made to obtain the suitable range, for comparison with KEP (in [43], the target range was $200-600$ pxl). The range of target in the chosen data was $265-800$ pxl.

The MAE (Mean Absolute Error) comparison at the different percentages of the way shows that, on average, we have a better estimate at all the stages of the trajectory (see Fig. 10). In our switched model, we have faster error convergence to the 50 % of the way, where the velocity peak is supposed to be situated and after the rate decreases to 95 %, and finally, it converges to the endpoint value with an average error of 9.6 pxl. In contrast, KEP shows convergence close to linear with the final average error of 69.9 pxl.

However, a more detailed analysis in the box-whisker plot in Fig. 11 shows that our reconstructed KEP algorithms mean error is affected by the larger amount of the outlier points and their wider distribution. By outlier, we mean the value bigger than 1.5 times the interquartile (box range). This fact could mean that the stability check applied from the [43] might not be enough to provide bounded prediction using the curve fitting process in all real-world cases.

For the more solid comparison, the statistical test on the prediction error data for all 10 users, considering different model (KEP, Switched), different target width, distance, PTF was provided. Repeated measures ANOVA showed a significant effect of model on the error (from 0.1 to 0.75 of the distance $p < 0.001$), with the most significant in 0.5 - $F(1,9) = 38.81$, $p < 0.001$, $\eta_p^2 = 0.81$. In the range from 0.75 to 1, the test showed no significant difference in the model performance. Also, the effect of the distance was significant in all cases (for instance, $F(1,9) = 214.35$, $p < 0.001$, $\eta_p^2 = 0.96$ for 0.5) target width in some cases ($F(1,9) = 11.34$, $p < 0.008$, $\eta_p^2 = 0.56$ for 0.5), pair model*distance ($F(1,9) = 16.61$, $p < .003$, $\eta_p^2 = 0.65$ for 0.5) and there was no effect of the PTF.

Another test was conducted using the error data without outliers in Fig. 11 (for all outliers (red-crossed points), the quartile bound was assigned (0.25 or 0.75 percentile)). Repeated measures ANOVA still showed a significant effect of the model in the range from 0.15 to 0.6 of the distance ($p < 0.01$) with the highest in 0.5 with $F(1,9) = 15.70$, $p < 0.003$, $\eta_p^2 = 0.64$, and no significant effect after. This means that even considering the instability of KEP early prediction, the Switched model introduced in this paper still outperforms KEP in the early and mid-distance prediction. Another advantage is that after removing the outliers for both models, the average final prediction is still closer with the Switched (5.2 pxl to 6.4 in KEP).

The comparison results are not surprising because the disadvantage of the KEP model is that the estimation is unstable in the early stage. This is why the KEP model authors claim to obtain a good estimation starting only from the 80 % of the total path, which is not the case in the model introduced in this paper. To conclude the section, we can state that the average error analysis and the showed that the presented switched algorithm outperforms the most known algorithm in the group of algorithms without memory (KEP), especially at the early phase (trajectory path 85 %) and converges to almost exact value at the end due to the separated correction phase estimation algorithm.

C. Pointing Transfer Function (PTF)

In terms of PTF estimation, a fifth-order polynomial was chosen to represent the gain, $q = 5$(the parameter tuning has been performed via the error and trial method). Several trials were used to identify the coefficients (12×4 trials in Fig. 12), and the position of the next trial (x,y) was built using only the velocity V_m set with the model (1). In Fig. 12, it is shown that the used approach reconstructs the PTF pretty accurately.

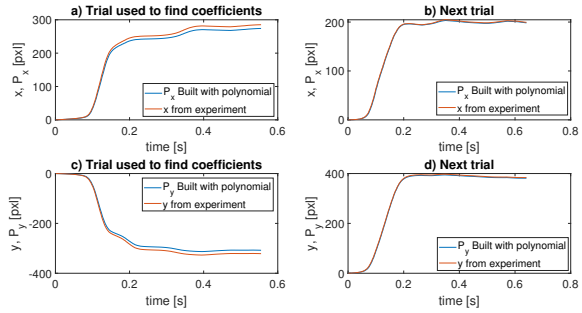

Fig. 12. Transfer functions: x, y [pxl] vs time

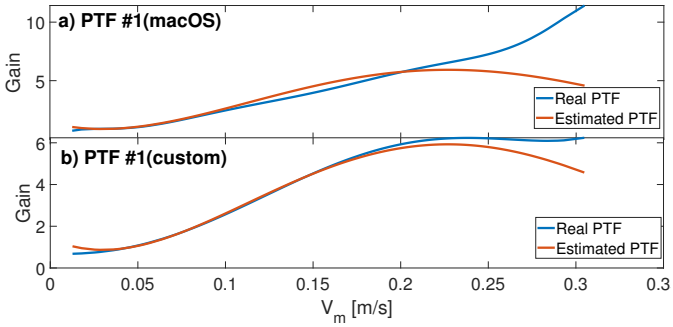

Fig. 13. Gains comparison: V_m [m/s] vs gain

The more trials are used to build the polynomial – the more accurate the approximation is. In the experiments recorded in [33], two different PTFs were used for the pointing task: a) the sigmoid function that mimics the default macOS transfer function provided by `libpointing`, configured with the following parameters: ($gmin = 1$ $gmax = 15$ $v1 = 0.05$ m/s $v2 = 0.6$ m/s), and b) the custom one, created with Libpointing which mimics constant gain of 4. The resulted gain curve,

$$\texttt{gain}(V_m) = \frac{TF(V_m)}{V_m},$$

of full trials for one user (in the range 0.0127-0.3 [m/s] of mouse inputted velocity) is represented in Fig. 13. One can see from these plots that in the lower range (0.0127-0.2 [m/s]), the curves tend to align more with the real gain curve because there were more points presented with these velocity values in the trials during the identification process, while in the range 0.2-0.3 [m/s] the values were rarer, so the deviation is bigger and the gain is less accurate.

These results confirm that the presented methodology provides an accurate way to reconstruct the PTF by an analytical function, which can help in future analysis and position estimation.

VII. Problems and Future Work

The presented results demonstrate the potential of the designed model and the tools for its identification. However, it leaves much room for improvement. In general, the human motion dynamics are nonlinear and highly uncertain (such as varying parameters, exogenous and endogenous perturbations). Hence, advanced robust-adaptive estimation and identification algorithms are essential in this domain, which constitutes a good benchmark for applying cut-edge technologies. A related issue is that a trade-off is necessary between the complexity and validity of the model used for estimation, and the possibility of designing a corresponding observer procedure. For many existing pointing movement models, as in [13]–[15], it is challenging to design a reliable identification method (due to, for instance, the switched nature of the model and its nonlinearity). That is why in this work, we have revisited the modeling theory for pointing and then developed new tools for identification.

In the previous section, it has been observed that the identification based on a linear time-invariant second-order model does not work well in the high-velocity ballistic phase yet. This finding motivates why we propose a switched model, with different dynamics in the correction and ballistic phases. This model can be considered as a simplified version of the one given in [15]. For the new model, the endpoint is well predicted at the velocity peak, but the idea of improving the correction phase dynamics to make it accurate even earlier is nevertheless relevant. How to correctly describe the ballistic phase in the model is a question for future research, for example, some interesting ideas were published recently in [45]. Using more data from trials of well-experienced users can provide a better understanding of the model's accuracy and limitations.

Another problem is the model's sensitivity to the data, which implies that sometimes the velocity profile requires an intensive tuning of the differentiator (thus, a high-frequency device should be acquired for better tuning). Although the velocity profile obtained from the differentiator is smooth enough, the acceleration is still noisy. Careful preliminary filtering of the sensor data may also improve accuracy.

The approach developed for the online identification of the pointing transfer function shows good performance, and increasing the degree of the polynomial applied for approximating the PTF leads to a more precise estimation. If more trials are used for the identification of the coefficients, then the better is the accuracy.

Our identification framework's bottleneck is the necessity to measure or calculate the mouse velocity and acceleration. The homogeneous differentiator in [35] provides a relatively good solution to this problem. Also, instead of differentiating x_m or V_{mx}, it is appealing to use another sensor, like in [33] with the inertial accelerometer. This accelerometer can provide more precise data with a much higher frequency. For comparison, the mouse optical sensor can run up to 1 kHz, while even a cheap inertial accelerometer can write data with the sampling up to 16 kHz. Then the only restriction would be the USB-port transmission capability. A fusion of the two approaches can lead to better identification and noise reduction.

VIII. Conclusion

A new simple dynamic pointing model was introduced as a feedback-based dynamical system with commutation between correction and ballistic phases. The model considers the cursor's position as the input and the human decision on

moving the mouse as an output. The dry friction, the users' regulating force, and the signal's treatment by a computer constitute the complete dynamical system, representing the indirect pointing task. Linear regression techniques for the identification of the model coefficients were used. The model was tested on the experimental data from [33] demonstrating a good prediction of the users' behavior. A comparison with an existing prediction algorithm was provided and showed better performance overall for our model. The estimation of PTF (a nonlinear function representing the gain from the mouse velocity to cursor velocity) was successfully represented as the polynomial with coefficients obtained from several trials by the linear regression technique. Discussion and future directions of research were given.

REFERENCES

[1] A. Oulasvirta, P. O. Kristensson, X. Bi, and A. Howes, *Computational interaction*. Oxford University Press, 2018.

[2] J. Müller, A. Oulasvirta, and R. Murray-Smith, "Control theoretic models of pointing," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 24, no. 4, pp. 1–36, 2017.

[3] S. Kim, B. Lee, T. Van Gemert, and A. Oulasvirta, "Optimal sensor position for a computer mouse," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–13.

[4] R. Blanch, Y. Guiard, and M. Beaudouin-Lafon, "Semantic pointing: improving target acquisition with control-display ratio adaptation," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2004, pp. 519–526.

[5] T. Grossman and R. Balakrishnan, "The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2005, pp. 281–290.

[6] O. Chapuis, J.-B. Labrune, and E. Pietriga, "Dynaspot: speed-dependent area cursor," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2009, pp. 1391–1400.

[7] R. Blanch and M. Ortega, "Rake cursor: improving pointing performance with concurrent input channels," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2009, pp. 1415–1418.

[8] R. Senanayake, R. S. Goonetilleke, and E. R. Hoffmann, "Targeted-tracking with pointing devices," *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 4, pp. 431–441, 2015.

[9] P. M. Fitts, "The information capacity of the human motor system in controlling the amplitude of movement." *Journal of experimental psychology*, vol. 47, no. 6, p. 381, 1954.

[10] S. Zhai, J. Kong, and X. Ren, "Speed–accuracy tradeoff in fitts' law tasks—on the equivalency of actual and nominal pointing precision," *International journal of human-computer studies*, vol. 61, no. 6, pp. 823–856, 2004.

[11] O. Chapuis, R. Blanch, and M. Beaudouin-Lafon, "Fitts' law in the wild: A field study of aimed movements," 2007.

[12] R. G. Costello, "The surge model of the well-trained human operator in simple manual control," *IEEE Transactions on Man-Machine Systems*, vol. 9, no. 1, pp. 2–9, 1968.

[13] D. Bullock and S. Grossberg, "Neural dynamics of planned arm movements: emergent invariants and speed-accuracy properties during trajectory formation." *Psychological review*, vol. 95, no. 1, p. 49, 1988.

[14] D. E. Meyer, R. A. Abrams, S. Kornblum, C. E. Wright, and J. Keith Smith, "Optimality in human motor performance: ideal control of rapid aimed movements." *Psychological review*, vol. 95, no. 3, p. 340, 1988.

[15] S. Aranovskiy, R. Ushirobira, D. Efimov, and G. Casiez, "A switched dynamic model for pointing tasks with a computer mouse," *Asian Journal of Control*, vol. 22, no. 4, pp. 1387–1400, 2020.

[16] G. Casiez and N. Roussel, "No more bricolage! methods and tools to characterize, replicate and compare pointing transfer functions," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 2011, pp. 603–614.

[17] E. Todorov and M. I. Jordan, "Optimal feedback control as a theory of motor coordination," *Nature neuroscience*, vol. 5, no. 11, pp. 1226–1235, 2002.

[18] E. Todorov, "Optimality principles in sensorimotor control," *Nature neuroscience*, vol. 7, no. 9, pp. 907–915, 2004.

[19] E. Todorov and W. Li, "A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proceedings of the 2005, American Control Conference, 2005.* IEEE, 2005, pp. 300–306.

[20] N. Qian, Y. Jiang, Z.-P. Jiang, and P. Mazzoni, "Movement duration, fitts's law, and an infinite-horizon optimal feedback control model for biological motor systems," *Neural computation*, vol. 25, no. 3, pp. 697–724, 2013.

[21] L. Rigoux and E. Guigon, "A model of reward-and effort-based optimal decision making and motor control," 2012.

[22] J. Izawa, T. Rane, O. Donchin, and R. Shadmehr, "Motor adaptation as a process of reoptimization," *Journal of Neuroscience*, vol. 28, no. 11, pp. 2883–2891, 2008.

[23] B. Berret, A. Conessa, N. Schweighofer, and E. Burdet, "Stochastic optimal feedforward-feedback control determines timing and variability of arm movements with or without vision," *PLOS Computational Biology*, vol. 17, no. 6, p. e1009047, 2021.

[24] J. A. Á. Martín, H. Gollee, J. Müller, and R. Murray-Smith, "Intermittent control as a model of mouse movements," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 28, no. 5, pp. 1–46, 2021.

[25] J. Gori and O. Rioul, "A feedback information-theoretic transmission scheme (FITTS) for modeling trajectory variability in aimed movements," *Biological Cybernetics*, vol. 114, no. 6, pp. 621–641, 2020.

[26] E. Lank, Y.-C. N. Cheng, and J. Ruiz, "Endpoint prediction using motion kinematics," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, 2007, pp. 637–646.

[27] N. Hogan, "An organizing principle for a class of voluntary movements," *Journal of neuroscience*, vol. 4, no. 11, pp. 2745–2754, 1984.

[28] T. Asano, E. Sharlin, Y. Kitamura, K. Takashima, and F. Kishino, "Predictive interaction using the delphian desktop," in *Proceedings of the 18th annual ACM symposium on User interface software and technology*, 2005, pp. 133–141.

[29] B. Ziebart, A. Dey, and J. A. Bagnell, "Probabilistic pointing target prediction via inverse optimal control," in *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, 2012, pp. 1–10.

[30] P. T. Pasqual and J. O. Wobbrock, "Mouse pointing endpoint prediction using kinematic template matching," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2014, pp. 743–752.

[31] D. Rozado, "Mouse and keyboard cursor warping to accelerate and reduce the effort of routine hci input tasks," *IEEE Transactions on Human-Machine Systems*, vol. 43, no. 5, pp. 487–493, 2013.

[32] J. Wang, D. Efimov, and A. A. Bobtsov, "Finite-time parameter estimation without persistence of excitation," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 2963–2968.

[33] A. Antoine, S. Malacria, and G. Casiez, "Using high frequency accelerometer and mouse to compensate for end-to-end latency in indirect interaction," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–11.

[34] J. P. Hourcade, C. M. Nguyen, K. B. Perry, and N. L. Denburg, "Pointassist for older adults: analyzing sub-movement characteristics to aid in pointing tasks," in *Proceedings of the sigchi conference on human factors in computing systems*, 2010, pp. 1115–1124.

[35] W. Perruquetti, T. Floquet, and E. Moulay, "Finite-time observers: application to secure communication," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 356–360, 2008.

[36] E. Bernuau, A. Polyakov, D. Efimov, and W. Perruquetti, "Verification of iss, iiss and ioss properties applying weighted homogeneity," *Systems & Control Letters*, vol. 62, no. 12, pp. 1159–1167, 2013.

[37] L. Ljung, *System identification: theory for the user*. Prentice Hall PTR, 1999.

[38] A. G. Feldman, "Once more on the equilibrium-point hypothesis (λ model) for motor control," *Journal of motor behavior*, vol. 18, no. 1, pp. 17–54, 1986.

[39] R. Bitmead, "Persistence of excitation conditions and the convergence of adaptive schemes," *IEEE Transactions on Information Theory*, vol. 30, no. 2, pp. 183–191, 1984.

[40] S. Aranovskiy, A. Bobtsov, R. Ortega, and A. Pyrkin, "Performance enhancement of parameter estimators via dynamic regressor extension and mixing," *IEEE Transactions on Automatic Control*, vol. 62, no. 7, pp. 3546–3550, 2016.

[41] M. Korotina, S. Aranovskiy, R. Ushirobira, and A. Vedyakov, "On parameter tuning and convergence properties of the drem procedure," in *2020 European Control Conference (ECC)*. IEEE, 2020, pp. 53–58.

[42] H. Keuning-van Oirschot and A. J. Houtsma, "Cursor displacement and velocity profiles for targets in various locations," in *Proceedings of Eurohaptics*. Citeseer, 2001, pp. 108–112.
[43] J. Ruiz and E. Lank, "Effects of target size and distance on kinematic endpoint prediction," 2009.
[44] ——, "Speeding pointing in tiled widgets: Understanding the effects of target expansion and misprediction," in *Proceedings of the 15th international conference on Intelligent user interfaces*, 2010, pp. 229–238.
[45] B. Berret and F. Jean, "Stochastic optimal open-loop control as a theory of force and impedance planning via muscle co-contraction," *PLoS computational biology*, vol. 16, no. 2, p. e1007414, 2020.
[46] A. Levant, "Higher-order sliding modes, differentiation and output-feedback control," *International journal of Control*, vol. 76, no. 9-10, pp. 924–941, 2003.

APPENDIX A
HOMOGENEOUS DIFFERENTIATOR

Let a smooth signal $y(t) \in \mathbb{R}$ be available for measurements for all $t \geq 0$, and our goal is to estimate its derivatives.

To this end, we can use the homogeneous finite-time observer from [35], which in our case plays the role of a differentiator.

$$\begin{array}{rcl} \dot{\widehat{z}}_1 & = & \widehat{z}_2 - k_1 \lceil \widehat{z}_1 - y \rfloor^{1+\alpha}, \\ \dot{\widehat{z}}_2 & = & \widehat{z}_3 - k_2 \lceil \widehat{z}_1 - y \rfloor^{1+2\alpha}, \\ & \vdots & \\ \dot{\widehat{z}}_n & = & -k_n \lceil \widehat{z}_1 - y \rfloor^{1+n\alpha}, \end{array} \tag{3}$$

where $\widehat{z}_i$ can be utilized as an estimate of $y^{(i-1)}$ derivative for $1 \leq i \leq n$ (here $y^{(0)} = y$), $\alpha \in \mathbb{R}$ and $k_i \in \mathbb{R}$ for $1 \leq i \leq n$ are tuning parameters. If $\alpha = \frac{n-1}{n}$, then this differentiator becomes the high order sliding-mode differentiator from [46].

For realization, the system (3) has to be discretized using, for example, the explicit Euler method at time instants $t_k = kh$ with a fixed step $h > 0$, and choosing the coefficients k_i in a way that ensures stability. For this purpose, for $n = 4$, taking the characteristic polynomial as $p(s) = (s+\lambda)(s+\lambda\xi)(s+\lambda\xi^2)(s+\lambda\xi^3)$, where $\lambda, \xi > 0$ are tuning parameters, we obtain:

$$k_1 = (1+\xi+\xi^2+\xi^3)\lambda,\ k_2 = (1+\xi+2\xi^2+\xi^3+\xi^4)\xi\lambda^2,$$
$$k_3 = (1+\xi+\xi^2+\xi^3)\xi^3\lambda^3,\ k_4 = \xi^6\lambda^4,$$

hence,

$$\begin{array}{rcl} \widehat{z}_1^{k+1} & = & \widehat{z}_1^k + h\left(\widehat{z}_2^k - k_1 \left\lceil \widehat{z}_1^k - y^k \right\rfloor^{1+\alpha}\right), \\ \widehat{z}_2^{k+1} & = & \widehat{z}_2^k + h\left(\widehat{z}_3^k - k_2 \left\lceil \widehat{z}_1^k - y^k \right\rfloor^{1+2\alpha}\right), \\ \widehat{z}_3^{k+1} & = & \widehat{z}_3^k + h\left(\widehat{z}_4^k - k_3 \left\lceil \widehat{z}_1^k - y^k \right\rfloor^{1+3\alpha}\right), \\ \widehat{z}_4^{k+1} & = & \widehat{z}_4^k + hk_4 \left\lceil \widehat{z}_1^k - y^k \right\rfloor^{1+4\alpha} \end{array}$$

for $k = 0, 1, 2, \ldots$, where $\widehat{z}_i^k = \widehat{z}_i(t_k)$ and $y^k = y(t_k)$ for $1 \leq i \leq 4$, and $\alpha \in [-\frac{1}{4}; 0)$.

APPENDIX B
DREM (DYNAMIC REGRESSOR EXTENSION AND MIXING) PROCEDURE

Consider the linear regression equation:

$$y(t) \quad = \quad \phi(t)^\top \theta,\ \forall t \geq 0,$$

where $y(t) \in \mathbb{R}$ is the measured output signal, $\phi(t) \in \mathbb{R}^n$ is the regressor, and $\theta \in \mathbb{R}^n$ is the vector of unknown constant parameters. The DREM technique, introduced in [40], consists of two steps.

The first step is the *dynamic regressor extension*, where we introduce linear BIBO (bounded-input bounded-output) stable dynamic operators H_i for $1 \leq i \leq n$ and define the vector $Y : \mathbb{R}_+ \to \mathbb{R}^n$ and the matrix $\Phi : \mathbb{R}_+ \to \mathbb{R}^{n \times n}$ by applying the operators on the scalar output y and on the regression vector ϕ:

$$Y = [Y_1 \ldots Y_n]^\top,\ \Phi = [\Phi^1 \ldots \Phi^n]^\top;$$
$$Y_i = H_i(y),\ \Phi^i = H_i(\phi),\ 1 \leq i \leq n.$$

Due to the linearity of the operators H_i and BIBO stability, these signals satisfy the equation:

$$Y = \Phi\theta$$

subject to transient asymptotically converging errors. One of the options of the extension is a *delay operator*

$$H_i(\cdot)(t) := \cdot(t - \tau_i)$$

for some $\tau_i > 0$, $\tau_i \neq \tau_j$ for $1 \leq i \neq j \leq n$.

The second step is *mixing*, whose objective is to derive a set of n scalar equations to separate the estimation of each parameter θ_i. Denoting $\widetilde{Y} = \Phi^* Y$ and $\varphi = \det(\Phi)$, where Φ^* is the adjugate matrix of Φ, we obtain:

$$\widetilde{Y}_i(t) = \varphi(t)\theta_i$$

for all $1 \leq i \leq n$. For this set of scalar equations, the gradient parameter estimation algorithm is applicable to obtain $\widehat{\theta}$ the estimate of θ:

$$\dot{\widehat{\theta}}_i \quad = \quad \gamma_i \varphi(\widetilde{Y}_i - \varphi\widehat{\theta}_i)$$

for $1 \leq i \leq n$, where $\gamma_i > 0$ is a tuning parameter.