AutoGain: Gain Function Adaptation with Submovement Efficiency Optimization

Byungjoo Lee, Mathieu Nancel, Sunjun Kim, Antti Oulasvirta

AutoGain: Gain Function Adaptation with Submovement Efficiency Optimization

Byungjoo Lee[1,2], **Mathieu Nancel**[3,2], **Sunjun Kim**[1,2], **and Antti Oulasvirta**[2]
[1]KAIST, [2]Aalto University, [3] Inria & Univ. Lille, UMR 9189 - CRIStAL, Lille, France
byungjoo.lee@kaist.ac.kr, mathieu.nancel@inria.fr, {sunjun.kim,antti.oulasvirta}@aalto.fi

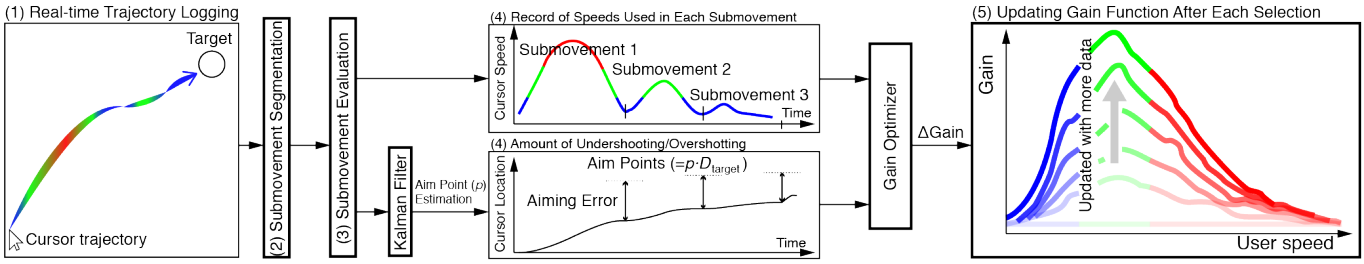

Figure 1. AutoGain adapts transfer function on indirect pointing devices by using a novel submovement-level tracking-and-optimization approach. It gradually updates the control-to-display (CD) gain function based on analysis of speed and error in a user's submovements. It updates the CD gain function trying to minimize expected aiming error for typical submovements.

ABSTRACT

A well-designed control-to-display gain function can improve pointing performance with indirect pointing devices like trackpads. However, the design of gain functions is challenging and mostly based on trial and error. AutoGain is a novel method to individualize a gain function for indirect pointing devices in contexts where cursor trajectories can be tracked. It gradually improves pointing efficiency by using a novel submovement-level tracking+optimization technique that minimizes aiming error (undershooting/overshooting) for each submovement. We first show that AutoGain can produce, from scratch, gain functions with performance comparable to commercial designs, in less than a half-hour of active use. Second, we demonstrate AutoGain's applicability to emerging input devices (here, a Leap Motion controller) with no reference gain functions. Third, a one-month longitudinal study of normal computer use with AutoGain showed performance improvements from participants' default functions.

Author Keywords

Pointing; Submovement; CD gain functions; Pointer acceleration; Human Performance; Pointing facilitation.

CCS Concepts

•**Human-centered computing** → **Pointing devices;**

INTRODUCTION

Control-to-display (CD) function is a pointing facilitation technique used in the vast majority of indirect input devices like mice and trackpads. A CD function determines the speed of a control point in display space (e.g. a cursor) from the user's motion speed in control space (e.g. a mouse speed). A gain function (f_{CD}) computes a scalar factor (or "gain") from the instantaneous input speed (v_{in}), that is multiplied to that input speed to obtain the speed of the control point (v_{out}): $v_{out} = f_{CD}(v_{in}) \times v_{in}$. The type and design of a gain function can affect the kinematics [18, 19] and performance [3, 5, 23, 30] of pointing. *Speed-dependent* CD gain functions are interesting to performance-oriented users, such as gamers and information workers. They affect which limbs are used [1, 33], reduce unnecessary clutching [14]. and allow users to avoid constraining high movement speeds that otherwise might lead to high error [5]. Some studies show improvements in selection time of up to 24% with speed-dependent gain functions [3], compared to constant gains.

Prior to this work, speed-dependent gain functions have been designed based on either trial and error [3] or heuristic iteration [23, 31] (see Related Work). While methods exist for constant functions [5], no automated or computer-assisted method exists for the general, speed-dependent case. Calibration-free methods exist, e.g. [10], but they are known to not scale well [22]. One challenge is that the design space is large: in principle, any continuous function is valid. A function must strike a balance between high and low gains that control the trade-off between the speed and accuracy of pointing. A high CD gain reduces the time to approach a distant target, but it may hamper precise positioning on top of a smaller target. Low gains increase precision but possibly at the cost of slower approaches. One also has to solve how to adapt the function:

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

How to infer data from a user and set the right speed while ensuring stable performance given that users also learn?

This paper investigates a novel computational approach to adapting control-to-display (CD) gain functions to individuals. *AutoGain* addresses the two challenges (Figure 1). It iteratively adapts the CD gain function after each target acquisition trial, using the estimated accuracy of its submovements. The submovement optimization approach is informed by a theory of motor control. It tries to minimize aiming error (overshooting/undershooting). This is calculated with respect to the user's inferred aim point for each submovement within the last target acquisition (see bottom of Figure 1). The gain function is discretized, with input speeds "binned" like in a histogram, and the gain of each bin is modified based on the speeds used in the previous pointing act. Unlike previous methods, the method requires no human supervision and minimal initialization: after setting the overall speed of convergence, it starts from any function and adapts it based on automated observations and principles. This makes it deployable for devices for which no manually designed or reference functions exist.

To sum up, AutoGain is the first method to automatically optimize a gain function from a user's actual motion data, using minimal assumptions about the shape of that function. In the rest of the paper, we detail the method, and report the results of three evaluative studies.

RELATED WORK

Despite extended early work on force-to-motion functions for isometric input (e.g. [26]), and although virtually every study using an isotonic indirect pointing device uses some form of speed-based transfer function, the design and adaptation of the latter remains a relatively little-studied topic [3, 23, 31].

Their simplest form is fixed gains, i.e. constant ratios between input and output movement velocities. Casiez et al. [5] presented a principle to choose usable ranges of fixed gain values. A minimum gain CD_{min} should allow the user to acquire the most distant targets without clutching, and a maximum gain CD_{max} should allow accessing each individual pixel. Nancel et al. later proposed a formulation that relaxes these constraints [24], allowing for some clutching and arbitrarily small targets.

Most commercial gain functions for mice and trackpads are not constant, but little research has been conducted to guide the design of such (speed-dependent) gain functions. It is generally agreed that gain functions can perform better than constant gains, even though that is usually derived from comparing one gain value against one or more gain functions, e.g. [3, 16]. In effect, systematic comparisons are scarce. Casiez and colleagues [5] compared 6 gain values and 6 gain function settings, and report a borderline-significant effect ($p = .065$) with an improvement of 3.3% in selection time between the best settings of fixed vs. non-constant gains.

Casiez and Roussel [3] reverse-engineered gain functions in existing operating systems. They found that all commercially deployed functions share some features, namely monotonic increase in the beginning and comparable maximum outputs. They also found differences, especially in minima, continuity, and shape of the functions.

Nancel and colleagues [22, 23] proposed a generic gain function based on the generalized logistic curve. It expresses four features of gain functions: the asymptotic minimum and maximum output gains, the abscissa (input velocity) of the curve's inflexion point, and its slope at that inflexion point. This function was successfully applied to translation- and rotation-based input channels, and later on to different input devices and interactive environments [12, 20]. However, tuning the parameters of the function is ad hoc: initial values are based on heuristics derived from [5], and suitability is left to designer's judgment. Earlier approaches combined simple sine-based CD gain functions with absolute position control [10, 16, 11], with an implicit velocity-based transition.

In contrast to previous studies considering how the magnitude of input movement is being transferred, several studies have investigated other aspects of control-to-display relationships, such as angular deviation [30], coordinate disturbance [19], proximity to target [2] and movement direction [18]. Results include accurate and faster drawing [19], reduced overshoot [18], and better experience for motor impaired users [30].

To sum up, speed-dependent gain functions (a.k.a. acceleration) are expected to be superior to constant gains. However, the design of successful gain functions for any setup has been limited to hand-tuned approaches. Research suggests that there is value in analyzing pointing kinematics in their design.

OVERVIEW OF TECHNIQUE

AutoGain is designed under the assumption that *the aiming error of individual pointing submovements is an indication of the inadequacy of the gain function*: if a submovement ends before its intended aim point (undershoot), then the gain function was too low and should be increased; if it ends further than its intended aim point (overshoot), then the gain function was too high and should be decreased. In both cases, the gain change should be proportional to the amount of aiming error. AutoGain applies this principle in a speed-dependent manner: it treats gain functions as series of gains associated to discrete intervals of input speeds, and only alters a gain when the corresponding "speed bin" was used in a pointing task. This procedure ensures it does not change the function erratically. The principle is summarized in Figure 2.

AutoGain subscribes to a local optimization scheme that assumes that there exists a case-specific optimum gain function (or several optimal gain functions with different trade-offs) for a user performing pointing tasks of certain difficulty and scale ranges, with a given pointing device. It also assumes that this function is reachable from an initial constant or speed-dependent gain function, by means of a sequence of local updates to the function. Moreover, when updates are based on repeated observations of the user, a reasonable estimate of the optimum function can be obtained despite several sources of variability in the human motor system.

To estimate the optimum, AutoGain builds on two assumptions in earlier theories of human motor control:

- **Submovement decomposition**: An aimed movement can be divided into one or more submovements [7, 21, 28] using local accelerations and decelerations in the speed profile.

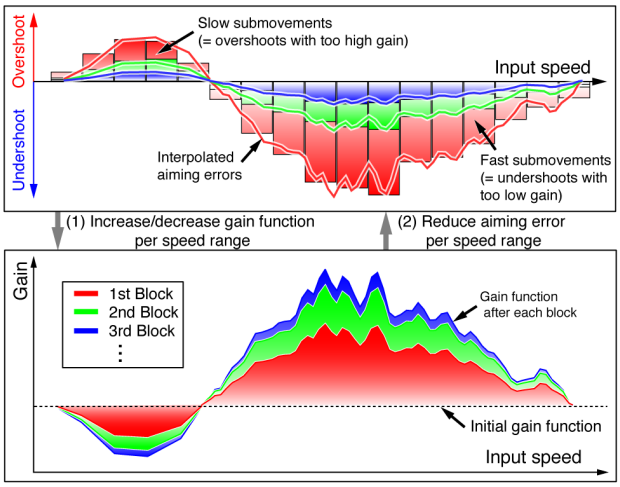

Figure 2. AutoGain improves a discrete gain function by updating gains locally per speed range. It segments a movement into submovements to update a profile of aiming errors, which it tries to reduce by adapting the gain response. Here, it is shown how AutoGain fixes too-low gains at high input speeds, and too-high gains at low input speeds.

- **Implicit aim point**: A submovement i has an implicit aim point located at a fraction (p_i) of the remaining distance to the target ($D_{target,i}$) at its beginning [7]:

$$D_{aim,i} = p_i \cdot D_{target,i} \tag{1}$$

  Due to stochastic noise in the motor system, submovement endpoints are distributed around the center of the aim point [21]. With respect to the aim point, the aiming error R_i of a submovement i is defined as:

$$R_i = (D_{aim,i} - D_{c,i}) \tag{2}$$

  With $D_{c,i}$ the distance moved during the submovement i (projected, see Figure 3). Negative errors indicate overshooting, positive errors indicate undershooting.

These assumptions exploit two well-known theories of aimed movements. In Crossman and Goodeve's *deterministic iterative corrections model* [7], aimed movements are modeled as series of ballistic, open-loop submovements aimed at a constant fraction (p) of the remaining distance to the target. However, because empirical data showed large variations in the duration and aim point of these submovements, an extension of the idea was proposed. Meyer's *stochastic optimized submovement model* [21] assumes that neuromotor noise causes the primary submovement to either undershoot or overshoot

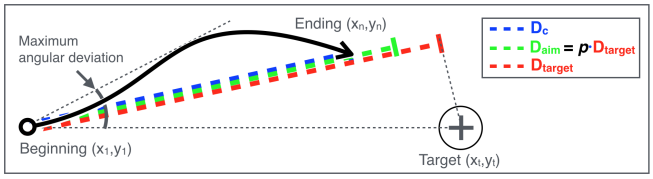

Figure 3. AutoGain assumes that each submovement has an implicit aim point (D_{aim}) located at a fraction (p) to the remaining target distance (D_{target}). In this figure, a submovement is undershooting, which is marked with a positive aiming error $R = (D_{aim} - D_c)$.

the target, which requires a corrective submovement to finally reach the target center.

IMPLEMENTATION

AutoGain updates the gain function after every target selection. The updating procedure consists in five steps (see Figure 1). First, it records a pointing trajectory from onset until the target selection. Second, it segments the trajectory into submovements using kinematic criteria. Third, it filters submovements out based on their trajectory and dynamic properties. Fourth, it computes the amount of aiming error (undershooting/overshooting) for each submovement, relative to its estimated aim point, as well as the input speeds that were used in that submovement. Fifth, it updates the gain function around these input speeds, using the amplitude and nature of these errors, following a local optimization scheme.

Step 1: Real-Time Trajectory Logging

AutoGain logs two different time vectors in real-time, for each input event t: (1) the raw input stream (dx_t, dy_t) (in counts) from the input device, and (2) the cursor trajectory $(x_{c,t}, y_{c,t})$ (in pixels). The raw input stream is used to obtain records of movement speeds in submovements. The cursor trajectory is used to segment submovements. We express the relationship between inputs and outputs as follows:

$$\begin{gathered} v_t = C_{in}\sqrt{dx_t^2 + dy_t^2} \\ (x_{c,t+1}, y_{c,t+1}) = (x_{c,t}, y_{c,t}) + C_{out} \cdot C_{in} \cdot G[v_t] \cdot (dx_t, dy_t) \\ \text{with } C_{in} = \text{Freq}_{in}/\text{Res}_{in} \text{ and } C_{out} = \text{Freq}_{in}/\text{Res}_{out} \end{gathered} \tag{3}$$

C_{in} is a factor converting unit of raw input (count) to m/s, which is determined from the input device's resolution (Res_{in}, here in points per millimeters) and update frequency (Freq_{in}, here in number of events per millisecond) [3]. C_{out} uses the opposite principle to convert transformed input movements in m/s into pixels translations, using the same update frequency Freq_{in} and the resolution of the display (Res_{out}, in pixels per millimeter). AutoGain treats gain functions as arrays of gains associated to discretized intervals of input speed. Gains values ($G[v_t]$) are therefore interpolated when v_t is not one of the discretized input values.

Step 2: Submovement Segmentation

From the beginning of a movement to the selection of the target, the coordinates are segmented into submovements based on local extrema in the cursor speed profile [8], using the `Persistence1D` [17] algorithm. It first smooths the speed profile (red curve in Figure 4) using a Gaussian kernel filter ($\sigma = 3$), then returns all pairs of minima and maxima that exceed a pre-defined persistence value (0.2). In most cases, resolution of raw inputs are available in integer steps so we set value of persistence less than 1.0 to ensure enough sensitivity on smaller submovements even after the smoothing process.

After identifying the local minima and maxima in the speed profile, each neighboring minimum-maximum-minimum triplet is considered to be a possible submovement. As in [8], we only consider submovements from the highest local maximum (① in Figure 4), which is assumed to be the initial

ballistic movement to the target. This helps exclude possible non-aiming movements during the trial.

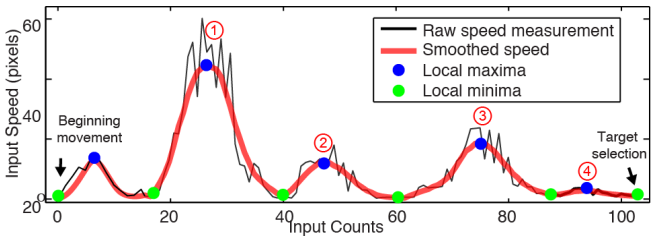

Figure 4. For each trial, `Persistence1D` identifies submovements as minimum-maximum-minimum triplets in the smoothed input speed profile, then excludes the ones preceding the highest maximum (here ①).

Step 3: Submovement Evaluation

AutoGain then identifies some unwanted characteristics of the submovements' trajectories: *unaimed*, *interrupted*, and *non-ballistic*. The goal of this classification is to filter out submovements that are likely to introduce noise in the updates of the gain function, or of the aim point (see Table 1).

We define *unaimed* submovements using two tajectory properties: (1) the maximum angular deviation, and (2) the amount of overshoot (see Figure 3). Maximum angular deviation is defined as the maximum angle between the line joining the first and last points of the submovement's trajectory, and the line joining the first and any other point of the submovement's trajectory. The amount of overshoot is defined as: $\max(D_c - D_{target}, 0)$. Any submovement that satisfies at least one of the following conditions is marked as *unaimed*:

- Maximum angular deviation $> 45°$.
- Amount of overshoot $> 0.5 \cdot D_{target}$.

These thresholds were obtained through trial and error and can be set more or less conservatively for the designer's purposes.

Submovements are considered *interrupted* when they fall distinctly short (less than halfway) from the remaining distance to the target, or when they are classified as clutching movements. Clutching can be determined with more or less certainty depending on the input device. Some devices allow straightforward detection of clutching by providing touch-down and touch-up events (e.g., styluses). Otherwise, predefined temporal thresholds can be used to detect the resetting of the end-effector, by measuring the time between two consecutive sensor events. In our implementation with a trackpad, we categorize a submovement as clutching when any interval of sensor events exceeds a predefined temporal threshold (130 ms). The last submovement of a task was never considered as clutching. For simplicity, submovements that are neither *unaimed* nor *interrupted* are deemed "normal".

Submovements that happen after the second normal submovement are assumed to be *non-ballistic*, i.e. continuously controlled. This assumption is based on the optimal submovement theory [21], which states that in ideal cases two submovements are enough to reach to a target. That classification is used to decide whether AutoGain will include a submovement in future updates of the gain function and the aim point, as reported in Table 1.

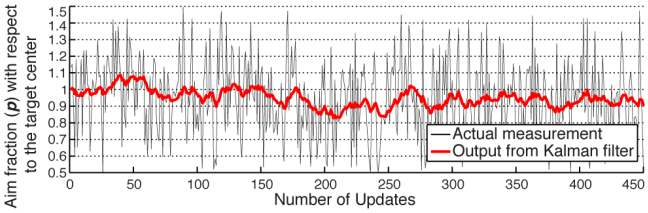

Figure 5. The aim proportion p is continuously updated by a Kalman filter applied after each ballistic submovement from actual measurements. The graph is showing the updates from participant 6 in Experiment 1 with AUTOGAIN condition (trial 1 to 60 in the first BLOCK).

Step 4: Speed Profiles and Aiming Errors

AutoGain measures the aiming error of interrupted and normal submovements. We defined in Equation 2 the aiming error R_i of a submovement i as the (projected) distance remaining to its estimated aim point at the end of the submovement (see Figure 3): $R_i = (D_{aim,i} - D_{c,i})$. In non-ballistic submovements, the aim point is assumed to be the center of the target. In ballistic movements, we assume that the user is aiming at a point located before the target, at a certain proportion p of the remaining distance. The true p value cannot be measured directly and the p_i value observed for each submovement i is contaminated from high measurement noise. Therefore, AutoGain estimates the true p value (assumed to be constant) based on the observed p_i values using a Kalman filter [15]:

$$p \sim p_i = f_{\text{Kalman}}\left((D_{target,i} - D_{c,i})/D_{target,i}, p_{i-1}\right) \quad (4)$$

The measurement noise of the filter was set to 40, the initial process noise and p_i were set to 0.2 and 1.0, respectively. This filtering is based on the assumption of stochastic noise in human movement, whose distribution is centered at the aim point. From the repeated observations, the filter removes stochastic noise similarly to a low pass filter (see Figure 5).

The filter is updated after each *normal+ballistic* submovement. *Interrupted* submovements are treated as aiming for the estimated aim point ($D_{aim} = p \cdot D_{target}$), but falling short due to insufficient gain or poor motor planning, and therefore do not update p. After estimating the aim fraction p_i of the current submovement, the aiming error is calculated using Equation 2: $R_i = (p_i \cdot D_{target,i} - D_{c,i})$.

AutoGain also records the input speeds used in every event of each submovement, in the form of a boolean array $S_i[V]$. The range of possible input speeds is binned into J speed intervals ($V_j = [v_j, v_{j+1}[, j \in [0, J-1]$) of constant width $w = v_{j+1} - v_j$. The corresponding array entry is false unless at least one event in the corresponding submovement had an instantaneous speed

Type of Submovement	Ballistic ($D_{aim} = pD_{target}$)	Non-ballistic ($D_{aim} = D_{target}$)
Normal	Update gains & p	Update gains only
Interrupted	Update gains only	
Unaimed	No update	

Table 1. Depending on the type of submovements, details of gain and aim point updates changes. The general objective of this classification is to exclude submovements with insufficient quality (noisy and unaimed movements) in order to facilitate the convergence of AutoGain.

v within that interval:

$$S_i[V_j] = \begin{cases} 1, & \text{if } \exists v : v_j \leq v < v_{j+1} \text{in submovement } i \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

R_i and $S_i[V_j]$ are used in the gain optimization process.

Step 5: Gain Update

The principle behind AutoGain, after each trial, is to increase (or decrease) the gain function when submovements in that trial undershot (or overshot) their estimated aim points, for the input speeds that were used during these submovement. AutoGain considers a gain function as a series of gain values associated to binned intervals of speeds input, using the same partitioning as in $S_i[V_j]$ (Equation 5). For each j-th speed interval, the gain function for the next trial $G_{t+1}[V_j]$ is updated from the gain function of the current trial $G_t[V_j]$ as follows:

$$G_{t+1}[V_j] = G_t[V_j] + \sum_{i=1}^{N_t} \Delta_i[V_j] \quad (6)$$

With N_t the number of submovements in the trial t, and Δ_i the corrections calculated from each submovement's aiming error. The amplitude of gain change $\Delta_i[V_j]$ for each speed bin is calculated using the amount of aiming error R_i, multiplied by a constant C that defines the overall rate of gain change. *Unaimed* submovements are excluded from the updates due to their poor aiming quality.

As stated earlier, only the speed bins that have been used in the submovements of the previous pointing task will be updated ($S_i[V_j]$ in Equation 5). However, a given speed interval can be used by more than one submovement within the same pointing task. To avoid over-favouring the speeds most commonly used, AutoGain prioritizes submovements in descending chronological order, on the principle that submovements that occurred earlier in a trial can be sufficiently represented by their faster speed components. The gain change of each speed bin V_j is therefore calculated in reverse order from the last submovement N_t, allowing only one change per bin (Equation 7). $\Delta_i[V_j]$ is calculated as expressed in Equation 8.

$$I_i[V_j] = \prod_{k=(i+1)}^{N_t} (1 - S_k[V_j]) \quad (7)$$

$$\Delta_i[V_j] = C \cdot R_i \cdot S_i[V_j] \cdot I_i[V_j] \quad (8)$$

Change Rate C

In effect, the AutoGain optimization process involves two interdependent active components: (1) gain optimization and (2) human skill acquisition. The system adapts the gain function to the user movements, then the user adapts his movements to the changes in the gain function, and so on. It is therefore crucial that the updates to the gain function occur fast enough to ensure a realistically short calibration process, but slow enough to allow the user to adapt to them.

This is implemented in AutoGain through the parameter C in Equation 8. If C is too large, the user will not have enough time to follow up the updates in the gain function and the system might become unstable. If C is too small, it will take too much time for the performance to converge. Overall, the effect of C on convergence speed will depend on the characteristics of the interactive system, like the scale difference between the input and output resolutions.

C expresses the relationship between the amount of gain change δ_g that should occur after M submovements of average aiming error μ_R, for a given speed bin:

$$C = \delta_g / (M \cdot \mu_R) \quad (9)$$

In applications of AutoGain, we recommend to start with an initial C value of 5×10^{-5} (in mm^{-1}), then increase or decrease the value by matching the convergence rate, observed through pilot testing, to the intended time pressure imposed on the optimizer and user.

STUDY 1: TRACKPAD GAIN FUNCTION

We first assess how AutoGain fares in a laptop+trackpad setup. We are particularly interested in whether it converges to a stable function, how long it takes, and how that resulting function compares to an established baseline (macOS function).

Participants

We recruited 11 paid participants (4 females) aged 22 to 38 (μ=27.4 years old, σ =5.3) from the local university. They were all regular MacBook trackpad users (μ=4.5 years, σ=2.2) and used a trackpad 5 hours a day on average (σ=3.1). Only one participant was left-handed, but typically used her right hand to control the trackpad. We also asked their current trackpad setting in Mac OS X (one of 10 positions on a slider, the higher the faster): 4 participants used the 4th position (OS default), 2 used the 6th, 4 used the 7th, and 1 used the 8th.

Design and Dependent Variables

The experiment followed a within-subject design with one independent variable: *gain condition* = {REFERENCE, AUTOGAIN}. In the REFERENCE condition, we replicated each participant's everyday pointer acceleration setting (see above) using the `libpointing` [3] library. Note that this REFERENCE condition is a strong baseline. It has evolved over many years of iteration and has been used for several months by participants. In the AUTOGAIN condition, the gain function was initially set to a constant gain of 1 ($G[v_t] = 1 \frac{m.s^{-1}}{m.s^{-1}}$). This function was then updated after each trial using the AutoGain method.

Participants completed 800 trials for each *gain condition*. We counterbalanced the order of conditions across the participants. For performance analysis, we blocked those trials into 10 *block* conditions of 80 trials with randomized Fitts's Indexes of Difficulty (ID). We used three dependent variables to compare REFERENCE and AUTOGAIN: *trial completion time*, *error rates*, and subjective assessments using the NASA-TLX [13].

Task and Procedure

The task consisted in selecting circular targets on a laptop screen using the embedded trackpad in the two different *gain condition*s. Participants were instructed to perform the tasks as quickly and as accurately as possible. A trial ended at the first click, regardless of selection errors. In the AUTOGAIN condition, participants were informed that changes in the gain

function could occur, but not whether those changes would be positive or negative. While this might bias participants, we thought it preferable to random reactions from (noticeable) gain function changes. They were also warned that pointing might feel slow or awkward in the beginning, and instructed to try to perform normally regardless. Participants were allowed use either tapping or pressing the touchpad to select the targets.

The target was a red disk presented on a black background. To obtain a comprehensive picture of the observed effects, we randomized the target diameter ($[2, 11.5]$ mm), the orientation between two consecutive targets ($[0, 2\pi]$ rad), and the ID of each task ($[2, 5.5]$ bits). We used the following randomization process after each successful click (x, y):

1 : $ID \leftarrow$ random(2, 5.5);
2 : **do:**
3 : $W_c \leftarrow$ random(2, 11.5);
4 : $x_c \leftarrow$ random(0, screen.width);
5 : $y_c \leftarrow$ random(0, screen.height);
6 : $ID_c \leftarrow \log_2\left(1 + \frac{\text{dist}[(x,y),(x_c,y_c)]}{W_c}\right)$;
7 : **until** $|ID_c - ID| < 0.1$

This ensured uniform distributions of IDs and orientations, providing comparable performance datasets.

Participants sat on a regular office chair that they could adjust, and used a laptop placed on a desk. They first filled a preliminary questionnaire about their regular MacBook trackpad usage and proceeded to the task. Participants were instructed to take a break every 80 selections and answered a NASA-TLX form about their performance since the last break. The experiment lasted about one hour per participant.

Apparatus

We ran the experiment on a MacBook Pro laptop (2012 version) running Mac OS X 10.11 with a integrated trackpad. The size of the display was 35.8 cm × 24.7 cm (1280 × 800 pixels). The experiment was coded in C++. The optimization speed parameter C in Eq. (6-8) was set to 6.4×10^{-5} mm^{-1} after pilot tests. The REFERENCE functions were obtained from `libpointing` [3], which was also used for cursor coordinate calculations in this *gain condition*. We used `libpointing`'s 'subpixel' option that transfers the remainder of the last calculated (floating) cursor coordinates to the next time step. We implemented the same subpixel mechanism in AutoGain, and discretized the range of input speeds into bins of 0.0079 m/s as REFERENCE condition. We set the processing noise parameter of the Kalman filter at 0.2, and the sensor noise parameter at 40.0. The refresh rate of the display was 60 fps.

Results

Data Processing

We excluded one participant from our dataset, who displayed irregular initial performance in both *gain condition*s as well as inconsistent aiming behavior (high variance in parameter p, Eq. (1)). Her ballistic movements were also notably slower: while the other participants' input was 96% higher in the first BLOCK of AUTOGAIN than of REFERENCE (due to the lower initial gain function), hers was only 15% higher in AUTOGAIN for that block. As a result, while AutoGain did improve her performance over time, the improvement was markedly slower. We exclude that participant from the following analyses (N=10), and take note that movement consistency can play an important role in the improvement rate of AutoGain.

We analyzed 8,000 trials per *gain condition*, corresponding to 19,766 submovements in the REFERENCE condition (2.47 per trial) and 19,828 submovements in the AUTOGAIN condition (2.48 per trial). In the REFERENCE condition, 4,349 submovements (22.0% of all submovements, 0.54 times per trial on average) were evaluated as *interrupted*, 4,774 submovements (24.1% of all submovements, 0.60 times per trial) were evaluated as *unaimed* submovements, and 6,031 submovements (30.5% of all submovements, 0.75 times per trial) were evaluated as *non-ballistic*. In the AUTOGAIN condition, 4,275 submovements (21.6%, 0.53 times per trial) were *interrupted* submovements, 3,577 (18.0 %) were evaluated as *unaimed* submovements, and 6,769 submovements (34.1% of all submovements, 0.75 times per trial) were evaluated as *non-ballistic*. The overall average of aim point (p) estimated from Kalman filter was 0.94 (σ =0.033) for AUTOGAIN condition. The completion time and error rates are averaged by blocks of 80 trials before analysis. The error rate is defined as the percentage of trials that misselected the target among all trials.

We used two-way repeated-measure ANOVAs for the following analysis. Although each participant used their personal gain setting in REFERENCE condition, the gain setting had no significant effect on trial completion time ($F(1,7)=0.686$, $p=0.435$). The order of gain conditions also had no significant effect on overall trial completion time ($F(1,7)=0.045$, $p=0.838$). So we exclude the above factors from the analysis.

Error Rates

The mean error rate was 9.3% (σ=2.2) for REFERENCE and 9.3% (σ=2.8) for AUTOGAIN. We attribute that overall high error rate to the higher number of small targets generated by the target randomization scheme, as has been reported in previous studies [29, 32]. We found no significant effect of *gain condition* ($F(1,9)=0.001$, $p=0.984$) and *block* ($F(1,9)=1.261$, $p=0.29$) on error rate. We also found no interaction effect between *block* and *gain condition* on error rate ($F(9,81)=1.02$, $p=0.432$). We conclude that the participants maintained comparable accuracy throughout the experiment.

Trial Completion Time

We found a significant effect of of *gain condition* on trial completion time ($F(1,9)=40.68$, $p<0.001$). The average completion time was 813.5 ms (σ=30.7) for REFERENCE and 869.1 ms (σ=26.4) for AUTOGAIN. This result is expected: in the AUTOGAIN condition, participants started from a slow gain of 1 $\frac{m.s^{-1}}{m.s^{-1}}$. However, the difference was only about 50 ms. This is on a similar level with a difference reported between OS X function and Windows in a previous study (= 50 ms) [3]. Also, it was quickly improved after few blocks (see Figure 6).

The interaction effect between *block* and *gain condition* on trial completion time was significant ($F(9,81)=10.24$, $p<0.001$). Pairwise comparison showed that the differences in trial completion time between *gain condition*s become not significant from BLOCK 7 onward ($p=0.35$, see Figure 6). The

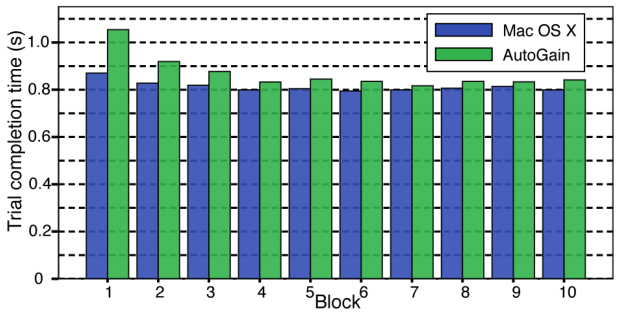

Figure 6. Study 1: The average difference in trial completion time between AUTOGAIN and REFERENCE was 56 ms. The difference became non-significant from BLOCK 7 (p=0.35) onward.

converged trial completion time of AutoGain after BLOCK 7 remained slightly higher than the baseline by 26.9 ms on average (n.s.). Completion times after BLOCK 7 were as follows:

Block #	REFERENCE (ms)	AUTOGAIN (ms)	p
7	799.35 (σ=34.8)	816.95 (σ=26.4)	.35
8	806.44 (σ=28.8)	835.34 (σ=27.8)	.065
9	814.28 (σ=38.9)	833.54 (σ=32.4)	.141
10	799.32 (σ=30.2)	841.31 (σ=32.4)	.137

Workload Metrics

For simplicity we report Raw TLX values [13] (Figure 7). We found a significant effect of *gain condition*s on mental demand (F(1,9)=6.40, p=0.032), physical demand (F(1,9)=21.69, p=0.001), and effort (F(1,9)=9.30, p=0.014). However, from the interaction effect between *block* and *gain condition*, these differences pertained to the beginning half of the study and became non-significant in the later half. (Table 2). Differences were not significant for temporal demand (F(1,9)=2.97, p=0.12), performance (F(1,9)=1.04, p=0.37), and frustration (F(1,9)=4.47, p=0.064).

Measure	REFERENCE	AUTOGAIN	Insignificant from
Mental demand	4.7 (σ=0.9)	6.4 (σ=1.3)	BLOCK 6 (p=.062)
Physical demand	5.2 (σ=0.8)	7.9 (σ=1.2)	BLOCK 7 (p=.063)
Effort	7.4 (σ=1.3)	9.0 (σ=1.4)	BLOCK 6 (p=.133)

Table 2. Mean and standard deviation for overall TLX measure in both *gain condition*s (lower is better). In every measure, the difference became insignificant after a certain BLOCK.

Summary

In 30 minutes of intensive use, AutoGain produced a well-performing gain function starting from scratch. User performance and workload ratings were comparable to that of Mac OS X trackpad function, which has been hand-tuned since 1994 (1st MacBook trackpad) and replicated the participants' daily settings. The data lend evidence to the main goal of AutoGain which is to reduce aiming errors as a means to improve pointing performance. Figure 8 shows that these errors are indeed being reduced over time. Performance between AutoGain and the baseline became similar at the same time as aiming errors, around BLOCK 7.

Figure 9 shows the gain functions obtained by AUTOGAIN alongside the Mac OS X functions used in REFERENCE (top),

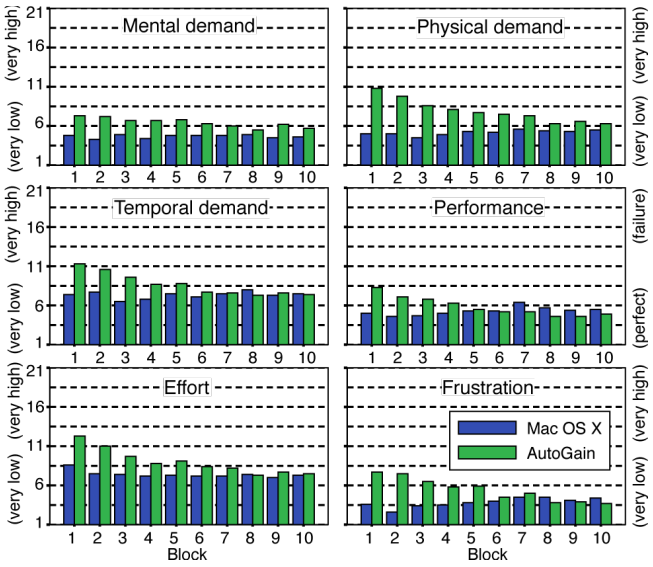

Figure 7. The difference in subjective ratings between REFERENCE and AUTOGAIN becomes insignificant after 6 to 7 blocks in the trial. Users reported performance being higher in AUTOGAIN than in REFERENCE after BLOCK 7.

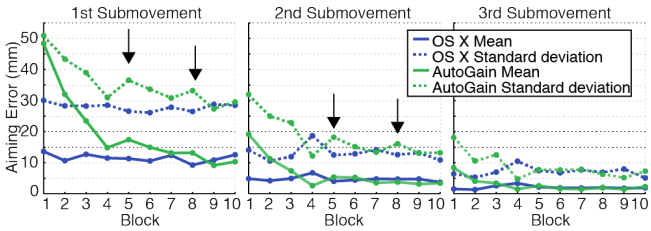

Figure 8. AutoGain gradually reduced aiming error in submovements to levels similar to the REFERENCE (Mac OS X). Fluctuating dynamics in aiming error were observed in BLOCKs 5 and 8 (black arrows).

and illustrates how the input speeds used in AUTOGAIN and REFERENCE became similar by the end of the experiment (bottom), especially between 0 and 0.3 m/s. This range also corresponds to the regions where the functions of AUTOGAIN and REFERENCE were most similar, both in value and slope.

The much steeper downward slope of the AutoGain-produced function after 0.3 m/s reflects the smaller number of input events featuring these input speeds: AutoGain updates the gain corresponding to an input speed interval only when this interval was used in the previous pointing task. However, the comparable completion times obtained between the two *gain condition*s tend to indicate that the gains above commonly used speeds is of lesser importance for overall performance.

STUDY 2: GAIN FUNCTION FOR AN EMERGING DEVICE

The objective of this study is to assess AutoGain's ability to produce usable gain functions on input devices that were not primarily designed for cursor control—in this case, a Leap Motion controller. Indirect pointing with this device has already been explored, e.g. in [20], using a hand-tuned sigmoid gain function based on [23]. We are interested to see if AutoGain stabilizes to a gain function with satisfying performance and

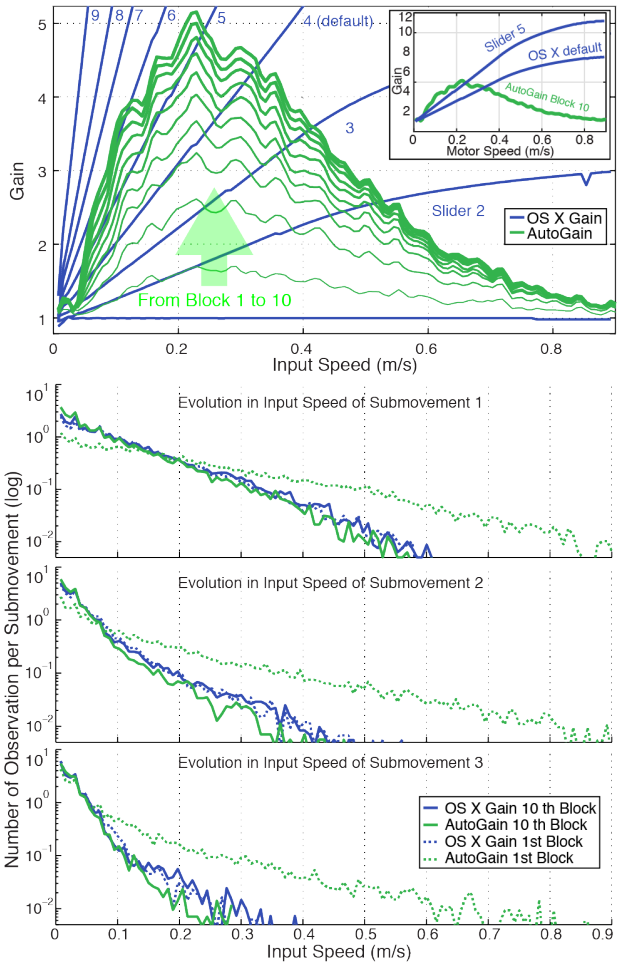

Figure 9. Top: AutoGain gradually updates the gain function until convergence (no more improvements in aiming error). Bottom: the evolution in input speed averaged for all participants. After 10 blocks of trials, participants were utilizing the same speed intervals with AutoGain as with the OS X function.

user feedback, and how that resulting function fares compared to the state-of-the-art function for this device.

Participants

We recruited 10 paid participants (3 females and 7 males; 23 years old in average, σ=3.33) from a local university. None of them had experience with Leap Motion. One used his left hand to control the cursor. Four had corrected-to-normal vision.

Design

This experiment design is similar to Study 1: it followed a within-subject design with one independent variable, *gain condition*: AUTOGAIN and SIGMOID. We counterbalanced the order of conditions across the participants. Each participants was to perform 800 trials for each gain condition, which we blocked into 10 *block*s of 80 trials.

The initial gain function for AUTOGAIN was a one-to-one constant ($G(v) = 1\ \frac{m.s^{-1}}{m.s^{-1}}$) between the speeds of the index finger and of the cursor. The SIGMOID condition used the function $G(v) = G_{min} + (G_{max} - G_{min})/(1 + e^{-\lambda(v - V_{inf})})$ with parameters tuned based on the recommendations in [23].

Task and procedure

The task was the identical to Study 1. Participants used their dominant hand over a Leap Motion tracker to move the cursor, and placed their non-dominant hand over the built-in keyboard to click and clutch. Clicking was performed by pressing the space bar. The cursor only moved when the C key was held pressed; clutching was performed by releasing that key, relocating the finger, and resuming pointing by pressing C again. The finger movements were interpreted as parallel to the display, i.e. on a vertical plane. This ensured forward/up visual compatibility [25] between hand and cursor movements.

During the tasks, participants rested their dominant elbow on a stand aside the desk (at desk height) to reduce fatigue, with the Leap Motion device facing up below their hand and 30 cm below desk height (see Figure 10-left).

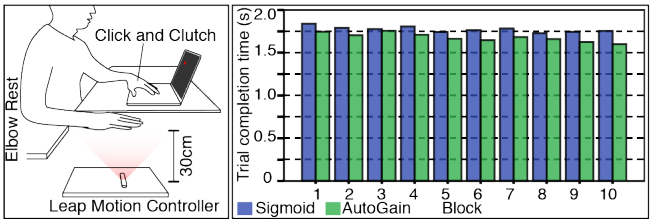

Figure 10. (left) Apparatus and trial completion times in Study 2. (right) Completion times per Condition and Block.

For each participant, we first calibrated the Leap Motion tracker using its inbuilt calibration software before starting the experiment. An experiment assistant then briefly demonstrated the pointing mechanism. Participants sat on a regular office chair that they could adjust, and used a laptop placed on a desk. They first filled a preliminary questionnaire asking about their previous experience with mid-air pointing.

Participants were instructed to take 10 minutes of practice session with an initial constant gain of 1 before the main part of the experiment started. The gain function was not updated during the practice session. During the main session, participants were invited to take breaks every 80 selection tasks. The experiment lasted about 90 minutes per participant.

Apparatus

The experiment was coded in C++ and run on the same computer as in Study 1. The cursor was controlled using mid-air hand movements tracked by a Leap Motion controller, (software version 2.3.1+31549). We used the Leap Motion in "Robust tracking mode" and smoothened its raw input using the `1 € filter` [4]. Following Casiez *et* al.'s tuning guidelines, we used 10^{-5} for minimum cutoff frequency and 0.05 for beta parameter.

For AUTOGAIN, the change rate parameter C was set to 3.6×10^{-5} mm^{-1} after pilot tests; this is lower than Study 1 to account for the possible harder learning and increased positional noise, compared to traditional trackpad input. We again set the processing noise parameter of the Kalman filter

at 0.2, and the sensor noise parameter as 40.0. The refresh rate of the display was 60 fps.

For dynamic tracking of moving hands, the spatial resolution of the Leap Motion controller was reported at around 0.7 mm [27]. Therefore, to avoid being sensitive to noise, we discretized the range of input speeds into bins of 0.06 m/s (1 mm/count, considering Leap Motion's 60Hz frequency).

For SIGMOID, we could not reuse the function parameters from [20] as their use-case was significantly different: participants were standing, the task was performed on a large display, participants had less training (36 trials), clicking and clutching were triggered by hand gestures with inconsistent recognition accuracy, etc. We adapted the tuning recommendations from [23] and obtained the following parameters: $G_{min} = 0.48$, $G_{max} = 2.93$, $V_{inf} = 0.219$, $\lambda = 12.54$.

Results

Data Processing

We gathered 34,193 submovements (4.27 times per trial) in the SIGMOID condition. Among them, 2,829 were *interrupted* (8.27% of all submovements, 0.35 times per trial), 14,657 were *unaimed* (42.9%, 1.83 times per trial), and 14,668 were *non-ballistic* (42.9%, 2.44 times per trial). We gathered 31,969 submovements (4.00 times per trial) in AUTOGAIN. Among them, 2,440 were *interrupted* (7.63% of all submovements, 0.31 times per trial), 14,226 were *unaimed* (44.5%, 1.78 times per trial), and 14,010 were *non-ballistic* (43.8%, 2.24 times per trial). The overall average of aim point (p) estimated from Kalman filter was 1.02 (σ= 0.021). In the following analysis, we used ANOVA and applied Greenhouse-Geisser correction when the assumption of sphericity was violated.

Error Rates

There were no significant effect of *gain condition* (p=0.063, F(1,9)=4.495) or *block* (p=0.559, F(3.23,29.07)=0.717) on error rate. Participants in AUTOGAIN had slightly higher error rates (μ=7.71%, σ=6.7%) than in SIGMOID condition (μ=5.09%, σ=4.4%, not significant).

Trial Completion Time

In Experiment 2, unlike Experiment 1, both gain functions were unfamiliar to the user. Therefore, we first eliminated the learning effect seen in the block and compared the two gain functions. There was a significant effect of *block* (p<0.001, F(9,81)=5.033). Overall average of trial completion time was 1.77 s (σ=0.152 s) in the SIGMOID condition and 1.68 s (σ=0.136 s) in the AUTOGAIN condition.

To test the effect of *block* on trial completion time, we used Helmert contrast , which consists of comparing the mean of each level of a factor (except the last) to the mean of subsequent levels. The effect of *block* on trial completion time becomes insignificant from block 8 (p=0.670). Hence we averaged block 8, 9, and 10 in the following analysis.

For averaged blocks 8, 9, and 10, the effect of *gain condition* on trial completion time was significant (p=0.008, F(1,9)=11.469). The mean trial completion time in the SIGMOID condition was 1.74 s (σ=0.144 s), and 1.62 s (σ=0.1 s) in the AUTOGAIN condition. Means and standard deviations

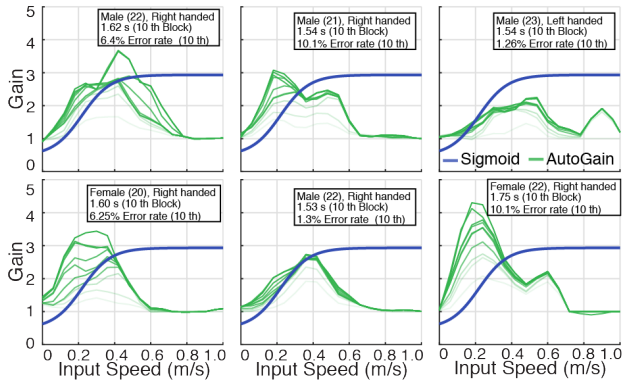

Figure 11. AutoGain adapts to each individual user. One left handed user converged to a very unique gain function, whose trial completion time was lower than other participants.

for each input condition are summarized in the table below, for blocks 8, 9, 10: We also analyzed the error rate again after

Block #	SIGMOID (ms)	AUTOGAIN (ms)
8	1782.72 (σ=137.4)	1660.01 (σ=120.2)
9	1744.58 (σ=171.3)	1625.19 (σ=115.8)
10	1756.56 (σ=135.8)	1599.06 (σ=85.0)

averaging blocks 8, 9, and 10. The effect of *gain condition* on error rate remained not significant (p=0.122, F(1,9)=2.915). The error rate in the SIGMOID condition for averaged blocks 8, 9, and 10 was 5% (σ= 4.9%), and 7.4% (σ=7.8%) in the AUTOGAIN condition.

Discussion

While there is no reference gain function for indirect pointing with Leap Motion, we compared AutoGain to a sigmoid function tuned using recommendations from previous work [23]. AutoGain was able to offer significant improvements, about 6.8%, in completion time when compared to the sigmoid function, without a significant decrease in accuracy. We note that the performance in SIGMOID conditions started already lower than AutoGain from the first block, which suggests a shortcoming of hand-tuning methods, that ultimately rely on subjective impressions rather than objective criteria.

As in the first study, the gain functions produced by AutoGain displayed significant completion time improvement from the initial block (here up to 20.2%), and reasonable completion times (here around 1.6 second). Figure 11 shows the evolution of the gain functions that it produced for a subset of our participants after each block. Each function appears to converge to a particular shape, which might reflect the movement specificities and pointing strategy of individual participants.

STUDY 3: FIELD STUDY WITH A WINDOWS APP

To assess whether AutoGain can improve long-term pointing performance in realistic setups, we ran a one-month longitudinal experiment with two participants, in which it gradually updated the gain function of a desktop OS (Windows 10), starting from each participant's pre-existing gain function. We developed an application (available as an open source project at https://github.com/SunjunKim/AutoGain) that bypasses the system's gain function and replaces it with a custom one.

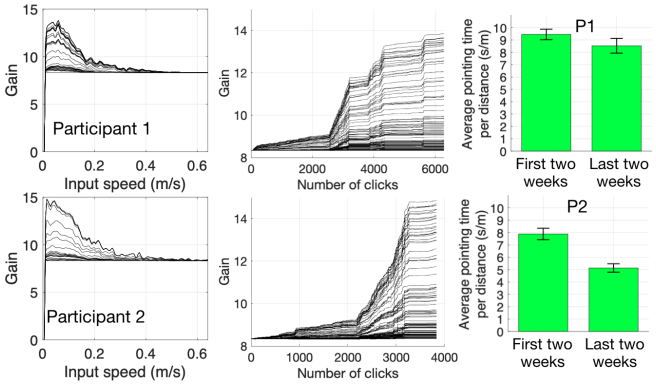

Figure 12. AutoGain deployed as a Windows application for everyday use. Left: evolution of the gain functions (upward), one line per 80 clicks. Middle: evolution of each individual gain bin. Right: Evolution of pointing performance, in s/m to account for uncontrolled tasks. Over a month, participants' pointing time per meter improved by an average of 21.4% by AutoGain. Error bars represent 95% confidence interval.

Method

We conducted a 1-month study with two volunteers (1 female, 1 male, ages 27 and 28). We installed the application on their office desktop computers, both operated by mice. Participants were informed that the behaviour of their cursor behavior might change throughout the study. We gave them no special instructions besides using their computer normally for about a month. The gain function they both usually used was the Windows default function, with the acceleration turned off. The optimization speed parameter C (Eq. 9) was set to 5×10^{-5} mm^{-1}, similar to the previous two user studies.

In everyday situations it is difficult to find out where the center of the target the user clicked was and when the first submovement of the aiming movement for that target began. Several methods have been proposed (see e.g. [6, 9]), but not formally compared. For simplicity, and because AutoGain does not require precise information about target shape or size, we applied two heuristics: (1) the clicked position approximates the target center, and (2) the submovement with the highest speed peak approximates the start of the pointing motion [9].

Results

6,285 (P1) and 3,839 (P2) click events were observed in the collection period. Unlike previous studies, participants barely noticed changes in mouse pointer movement. This is because clicks, and therefore gain function updates, occurred at much lower rates than during a controlled experiment. Also, being an "in-the-wild" study, the percentage of unaimed submovements (30.4 %) that did not lead to an update of the gain function was higher than in Study 1 with the trackpad (18 %).

The gain functions in each computer evolved in a similar fashion (see Fig. 12), with a steep increase around clicks #2500-3500 surrounded by gentler slopes. We hypothesize that participants first got used to slightly higher gains, progressively learning to take advantage of it by using smaller input speeds. This prompted AutoGain to rapidly increase gains up to a point where it no longer measured an aiming advantage. The same process repeated afterwards, with less drastic increases, as users continued to adapt their movements.

To assess the evolution of pointing performance, we compared overall pointing times between the first and last two weeks of the experiment. Since task difficulty can vary greatly, and we cannot reliably assess target size from pointing movement only, we considered pointing time as a function of pointing distance. The two participants took resp. 9.45 (σ=12.13) and 7.89 s/m (σ=10.72) for the first two weeks, but 8.53 (σ=17.15) and 5.14 (σ=7.91) s/m for the last two, i.e. improved by resp. 9.74 % and 34.85 %. Participants reported no difficulty using the mouse while AutoGain was on.

SUMMARY AND LIMITATIONS

We introduced a method inspired by motor control theories to automatically optimize a gain function for a user, using only pointer motion as input and making minimal assumptions about the shape of that function. AutoGain gradually adapts the gain function to minimize aiming error at submovement level. In Study 1, AutoGain produced gain functions for trackpads yielding performance comparable to that of widely used commercial gain functions, from scratch, in about 30 minutes of active use. In Study 2, it produced gain functions for a Leap Motion controller, for which no well-established reference function yet exists, and yielding faster task completion than a sigmoid function from the literature, hand-tuned to the best of our abilities. AutoGain produced notably distinct gain functions for different participants, indicating potential to adapt to individual movements and pointing strategies. Study 3 showed that the obtained functions converge in real use and improve pointing performance over users' default function.

AutoGain opens up exciting possibilities for personalized pointing facilitation techniques, in both existing and new systems. It could be used to accelerate the adoption of novel input devices for which no reference gain function yet exists. Pointing facilitation techniques could adapt independently in different context, e.g. for games or drawing apps. To this end, we publish AutoGain's Windows application as open source. In future work, we plan to apply AutoGain to more varied platforms, such as large displays controlled in mid-air, or high-precision machinery. Exploring a broad range of such systems might help us refine its initialization steps, e.g. the rate at which gain updates occur (C in Eq. 8), which would both accelerate the convergence of the optimization process and, possibly, the efficiency of the resulting gain functions.

Finally, while AutoGain was able to show statistically significant improvements over a baseline (Study 2), more empirical work is needed to disentangle the effects of gain adaptation from users' learning, both of which affect each other.

ACKNOWLEDGEMENTS

This research was funded by the National Research Foundation of Korea (2020R1A2C4002146), Korea Creative Content Agency (R2019020010), and the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (No 637991)

REFERENCES

[1] Johnny Accot and Shumin Zhai. 2001. Scale effects in steering law tasks. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1–8.

[2] Renaud Blanch, Yves Guiard, and Michel Beaudouin-Lafon. 2004. Semantic Pointing: Improving Target Acquisition with Control-display Ratio Adaptation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. ACM, New York, NY, USA, 519–526. DOI: http://dx.doi.org/10.1145/985692.985758

[3] Géry Casiez and Nicolas Roussel. 2011. 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*. ACM, 603–614.

[4] Géry Casiez, Nicolas Roussel, and Daniel Vogel. 2012. 1 € filter: a simple speed-based low-pass filter for noisy input in interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2527–2530.

[5] GÃl'ry Casiez, Daniel Vogel, Ravin Balakrishnan, and Andy Cockburn. 2008. The Impact of Control-Display Gain on User Performance in Pointing Tasks. *HUMANâĂŞCOMPUTER INTERACTION* 23, 3 (2008), 215–250.

[6] Olivier Chapuis, Renaud Blanch, and Michel Beaudouin-Lafon. 2007. *Fitts' Law in the Wild: A Field Study of Aimed Movements*. Technical Report. https://hal.archives-ouvertes.fr/hal-00612026 LRI Technical Repport Number 1480, Univ. Paris-Sud, 11 pages.

[7] ERFW Crossman and PJ Goodeve. 1983. Feedback control of hand-movement and Fitts' law. *The Quarterly Journal of Experimental Psychology* 35, 2 (1983), 251–278.

[8] Abigail Evans and Jacob Wobbrock. 2012a. Taming Wild Behavior: The Input Observer for Obtaining Text Entry and Mouse Pointing Measures from Everyday Computer Use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 1947–1956. DOI:http://dx.doi.org/10.1145/2207676.2208338

[9] Abigail Evans and Jacob Wobbrock. 2012b. Taming wild behavior: the input observer for obtaining text entry and mouse pointing measures from everyday computer use. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 1947–1956.

[10] Scott Frees, G. Drew Kessler, and Edwin Kay. 2007. PRISM Interaction for Enhancing Control in Immersive Virtual Environments. *ACM Trans. Comput.-Hum. Interact.* 14, 1, Article 2 (May 2007). DOI: http://dx.doi.org/10.1145/1229855.1229857

[11] Luigi Gallo, Mario Ciampi, and Aniello Minutolo. 2010. Smoothed Pointing: A User-Friendly Technique for Precision Enhanced Remote Pointing. In *Proceedings of the 2010 International Conference on Complex, Intelligent and Software Intensive Systems (CISIS '10)*. IEEE Computer Society, Washington, DC, USA, 712–717. DOI: http://dx.doi.org/10.1109/CISIS.2010.76

[12] Faizan Haque, Mathieu Nancel, and Daniel Vogel. 2015. Myopoint: Pointing and Clicking Using Forearm Mounted Electromyography and Inertial Motion Sensors. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 3653–3656. DOI: http://dx.doi.org/10.1145/2702123.2702133

[13] Sandra G Hart. 2006. NASA-task load index (NASA-TLX); 20 years later. In *Proceedings of the human factors and ergonomics society annual meeting*, Vol. 50. Sage Publications, 904–908.

[14] Herbert D Jellinek and Stuart K Card. 1990. Powermice and user performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 213–220.

[15] Rudolph Emil Kalman. 1960. A new approach to linear filtering and prediction problems. *Journal of basic Engineering* 82, 1 (1960), 35–45.

[16] Werner A. König, Jens Gerken, Stefan Dierdorf, and Harald Reiterer. 2009. Adaptive Pointing: Implicit Gain Adaptation for Absolute Pointing Devices. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems (CHI EA '09)*. ACM, New York, NY, USA, 4171–4176. DOI: http://dx.doi.org/10.1145/1520340.1520635

[17] Yeara Kozlov and Tino Weinkauf. 2013. Persistence1D. Code. (23 November 2013). Retrieved June 2, 2016 from https://people.mpi-inf.mpg.de/~weinkauf/notes/persistence1d.html.

[18] Byungjoo Lee and Hyunwoo Bang. 2013. A kinematic analysis of directional effects on mouse control. *Ergonomics* 56, 11 (2013), 1754–1765.

[19] Byungjoo Lee and Hyunwoo Bang. 2015. A Mouse With Two Optical Sensors That Eliminates Coordinate Disturbance During Skilled Strokes. *Human–Computer Interaction* 30, 2 (2015), 122–155.

[20] Mingyu Liu, Mathieu Nancel, and Daniel Vogel. 2015. Gunslinger: Subtle Arms-down Mid-air Interaction. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15)*. ACM, New York, NY, USA, 63–71. DOI: http://dx.doi.org/10.1145/2807442.2807489

[21] David E Meyer, Richard A Abrams, Sylvan Kornblum, Charles E Wright, and JE Keith Smith. 1988. Optimality in human motor performance: ideal control of rapid aimed movements. *Psychological review* 95, 3 (1988), 340.

[22] Mathieu Nancel, Olivier Chapuis, Emmanuel Pietriga, Xing-Dong Yang, Pourang P. Irani, and Michel Beaudouin-Lafon. 2013. High-precision pointing on large wall displays using small handheld devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 831–840. DOI: http://dx.doi.org/10.1145/2470654.2470773

[23] Mathieu Nancel, Emmanuel Pietriga, Olivier Chapuis, and Michel Beaudouin-Lafon. 2015a. Mid-Air Pointing on Ultra-Walls. *ACM Trans. Comput.-Hum. Interact.* 22, 5, Article 21 (Aug. 2015), 62 pages. DOI: http://dx.doi.org/10.1145/2766448

[24] Mathieu Nancel, Emmanuel Pietriga, Olivier Chapuis, and Michel Beaudouin-Lafon. 2015b. Mid-air pointing on ultra-walls. *ACM Transactions on Computer-Human Interaction (TOCHI)* 22, 5 (2015), 21.

[25] James G Phillips, Thomas J Triggs, and James W Meehan. 2005. Forward/up directional incompatibilities during cursor placement within graphical user interfaces. *Ergonomics* 48, 6 (2005), 722–735.

[26] Joseph D Rutledge. 1990. Force-to-motion functions for pointing. In *Proc, INTERACT'90: IFIP Conf. on Human-Computer Interaction*. 701–705.

[27] Frank Weichert, Daniel Bachmann, Bartholomäus Rudak, and Denis Fisseler. 2013. Analysis of the accuracy and robustness of the leap motion controller. *Sensors* 13, 5 (2013), 6380–6393.

[28] Deric Wisleder and Natalia Dounskaia. 2007. The role of different submovement types during pointing to a target. *Experimental Brain Research* 176, 1 (2007), 132–149.

[29] Jacob O Wobbrock, Edward Cutrell, Susumu Harada, and I Scott MacKenzie. 2008. An error model for pointing based on Fitts' law. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 1613–1622.

[30] Jacob O. Wobbrock, James Fogarty, Shih-Yen (Sean) Liu, Shunichi Kimuro, and Susumu Harada. 2009. The Angle Mouse: Target-agnostic Dynamic Gain Adjustment Based on Angular Deviation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 1401–1410. DOI: http://dx.doi.org/10.1145/1518701.1518912

[31] J Yun, YK Lim, KE Kim, and S Song. 2015. Interactivity Crafter: An Interactive Input-Output Transfer. *Archives of Design Research* 28, 3 (2015), 21–37.

[32] Shumin Zhai, Jing Kong, and Xiangshi Ren. 2004. Speed–accuracy tradeoff in FittsâĂŹ law tasksâĂŤon the equivalency of actual and nominal pointing precision. *International journal of human-computer studies* 61, 6 (2004), 823–856.

[33] Shumin Zhai, Paul Milgram, and William Buxton. 1996. The influence of muscle groups on performance of multiple degree-of-freedom input. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 308–315.