Pitch Pipe: An Automatic Low-pass Filter Calibration Technique for Pointing Tasks

Eugene M. Taranta II [*]
University of Central Florida

Seng Lee Koh [†]
University of Central Florida

Brian M. Williamson [‡]
University of Central Florida

Kevin P. Pfeil [§]
University of Central Florida

Corey R. Pittman [¶]
University of Central Florida

Joseph J. LaViola Jr. [‖]
University of Central Florida

Abstract

Practitioners use low-pass filters to improve the quality of noisy input device signals whose optimal parameters depend on application-specific precision and latency requirements, as well as situational human and environmental factors. Two common calibration approaches are to learn optimal filter parameters from training data, or interactively tune via trial and error until satisfaction ensues, both having major drawbacks. We propose a novel, automatic custom calibration technique for pointing tasks called Pitch Pipe that in three straightforward steps is able to determine appropriate parameters for a given filter, and is therefore suitable for deployment into unknown environments. Specifically, we estimate noise and user speed, and then select those parameters that best meet system requirements. In a widely deployed Fitts' task user study, we show that Pitch Pipe-tuned filters perform on par with their manually calibrated counterparts, demonstrating that one may use our automatic approach for custom calibration.

Keywords: Low-pass filter, automatic calibration, pointing, 1€

Index Terms: Human-centered computing—Human computer interaction—Interaction techniques—Pointing

1 Introduction

Humans interact with computers through input devices whose signals are perturbed by thermal noise, interference, precision errors, and other types of corruption-causing phenomena that affect performance [23, 24]. An augmented reality interface designed for selection and manipulation or that uses a pointing device may lack precision due to noise-induced jitter, and a virtual reality, full-body gesture interface may lack discriminatory power due to inflated feature variance. For these and other reasons, practitioners often utilize low-pass filters to improve signal quality before evaluating embedded content. Such filters attenuate high frequency noise while preserving low band information, as is appropriate given that humans interact through low frequency motion [42]. However, noise is not one's only concern, as variability in motion similarly impacts signal quality, and over aggressive filtering may lead to lost information.

More specifically, in calibrating a low-pass filter, one must consider the impact of both precision and lag on performance [5, 23]. When a signal is corrupted by noise, repeated measures of a motionless input device yields a varying response that is realized as an imprecise, jittery pointer. Aggressive filtering can attenuate noise and improve precision, but doing so introduces latency, which causes one's cursor to lag behind its true position. Due to the detrimental effect of noise and latency on throughput, practitioners must strike a balance between speed and accuracy while simultaneously considering their unique application requirements.

In a lab setting, one can achieve acceptable performance by using trial and error calibration. However, additional considerations influence parameter selection when conducting user studies or deploying software into unknown environments. For example, one may deploy their software across multiple platforms via a transreality system where one must support a variety of input devices [21]. Noise levels can vary between input devices as well as individuals based on sensor, software, and environmental factors. A computer vision system may have difficulty tracking certain skin tones or clothing colors under various lighting conditions and backgrounds. An individual with a degenerative disease such as Parkinson's may inadvertently add additional noise to the input signal. Unlike noise, latency depends on speed, where fast moving individuals will experience lag different from those who move slowly.

One way to account for these variabilities is to allow for automatic, custom filter calibration. Common practices, however, are not automated. Such techniques include off-line parameter sweeps over representative training data [18, 41], subjective interactive feedback [5], guesswork through trial and error, or synchronization through multiple sensors [38], each of which have their own drawbacks, thus motivating the need for a better solution. Our answer is Pitch Pipe, a straightforward approach to online, automatic low-pass filter parameter selection, comprising only three steps:

1. Estimate noise using power spectral density analysis
2. Estimate maximum speed by asking one to complete a representative task
3. Select suitable filter parameters based on noise and speed measurements as well as precision and lag requirements

In this paper, we not only describe our Pitch Pipe system, but demonstrate its effectiveness through an online, in-the-wild Mechanical Turk Fitts' task test using a state-of-the-art low-pass filter [5]. Results show that Pitch Pipe chooses parameters that perform on par with those selected by participants, and automatic calibration is significantly faster. Finally, we also provide Pitch Pipe reference code that can be found at `http://github.com/ISUE/PitchPipe`.

2 Related Work

Low-pass filters are used throughout a variety of domains, including HCI, especially for tracking and smoothing user input. For example, Xiao et al. [41] focused on jitter reduction to improve motion estimates; Dabra et al. [7] smoothed Kinect sensor motion input; and Feit et al. [8] used filtering to track eye movement. Filtering has been used in augmented reality system by Liang et al. [20] and Zhu et al. [43], as well as in virtual reality [36]. These selected examples are only a few among many.

In all cases, one has to make some kind of decision about how to calibrate their filter. Options fall into three broad categories:

[*]e-mail: etaranta@gmail.com
[†]e-mail: ksenglee@knights.ucf.edu
[‡]e-mail: brian.m.williamson@knights.ucf.edu
[§]e-mail: kevin.pfeil@knights.ucf.edu
[¶]e-mail: cpittman@knights.ucf.edu
[‖]e-mail: jjl@cs.ucf.edu

manual or interactive tuning, machine learning based calibration, and adaptive techniques. Manual tuning is common, where one uses trial and error to fine tune parameters according to preference and taste. This approach can be effective and easy for low noise levels, but may require expert knowledge or prior experience depending on task and filter complexity. One practical example involves the 1€ filter [5] that comprises two parameters, enabling one to balance jitter and latency, for which the authors propose a manual calibration procedure. The filter is very effective, but can be enhanced with an automatic calibration procedure.

2.1 Machine Learning Techniques

Researchers have explored a variety of machine learning techniques for parameter selection. Saha et al. [28] and Borah et al. [3] used particle swarm optimization; Horvath et al. [11] utilized a genetic algorithm; Russo [27] used a histogram of edge gradients for image denoising; Harshcer et al. [10] used a gradient descent method to optimize physical screw turns for their filter; Chen et al. [6] combined a median filter with an automated parameter tuning technique.

Grid searches (or parameter sweeps) are especially common. In this approach, each parameter's range is divided into a set of regularly spaced intervals that when taken collectively form a grid pattern. Thereafter, one iteratively walks over the grid in order to exhaustively evaluate each parameter combination. LaViola [18], for example, used a grid search to tune the double exponential moving average and Kalman filters to minimize the root mean squared error for position and orientation data.

An issue with machine learning based approaches is that one requires both training and ground truth data to evaluate their filter. The data may also require cleaning or other preparation work, a process that typically requires domain knowledge. These issues make it difficult for one to support custom calibration because of the large amounts of data one needs to collect in order to cover all scenarios, which is not a problem for Pitch Pipe. Although we internally use a grid search to find the optimal parameter set, we evaluate synthetic signals, eliminating the need for massive data collection or preparation.

2.2 Adaptive Filters

Another popular tuning approach involves the use of adaptive filters. Given a first primary input signal and a second reference input signal, an adaptive filter adjusts parameter weights until differences measured by an error function are minimized. Research in adaptive filtering primarily focuses on specialized filters [1, 2, 17, 30] and optimizations for fast convergence [12, 22, 34], but such filters have also found their way into HCI applications. For instance, Bulling et al. [4] applied a filter to remove unwanted noise found during eye tracking with an EOG device; Lee et al. [19] devised an adaptive filter for an ECG device; and Kaluri et al. [16] used an adaptive filter in the context of gesture recognition. Pitch Pipe, unlike an adaptive filter, does not require a second signal, but instead uses internal synthetic signals to calibrate a filter.

2.3 Complexity

Straightforward techniques that work well are appreciated and widely adopted by the HCI community, such as $-family custom gesture recognizers [35, 40] and double exponential smoothing [18]. Since Pitch Pipe leverages easy to use DSP techniques, we believe HCI practitioners will also use Pitch Pipe to simplify their own calibration efforts, as well as enable customize calibration in their deployments.

3 Pitch Pipe

Pitch Pipe is a custom low-pass filter calibration technique that finds optimal parameters based on context specific information. As such, Pitch Pipe requires three inputs: a signal from which to derive relevant characteristics, a low-pass filter to calibrate, and an application specific criteria to optimize for, namely precision and lag. From the input device signal, Pitch Pipe extracts noise and maximum user speed estimates, which we use to generate synthetic noise and edge patterns. Thereafter, Pitch Pipe performs a grid search over the filter's parameter space, evaluating its performance on the stated synthetic data. Pitch Pipe finally outputs the parameter set that best matches the application specific criteria. As such, Pitch Pipe internally comprises three steps that are to estimate noise, estimate maximum user speed, and optimize the parameter set, each of which are discussed in throughout the remainder of this section.

Step 1: Estimate Noise Variance. Frequencies void of motion information that would otherwise carry zero power are nevertheless corrupted by noise. When noise is white Gaussian, one can estimate its variance σ^2 using power spectral density (PSD) analysis. Specifically, we perform a discrete Fourier transform (DFT) on select frequencies, calculate power, and average together results over time. Those unfamiliar with digital signal processing (DSP) concepts and who desire a deeper understanding of what follows can refer to Smith's [29] accessible introductory treatment. For efficient processing, we use a constant time, one second long sliding window DFT (SDFT) [13, 14] that incrementally computes frequency k's Fourier coefficient as each new input device sample arrives. The SDFT is defined as:

$$S_k(t) = e^{2\pi i k \frac{1}{N}} \left[S_k(t-1) + x_t - x_{t-N} \right], \tag{1}$$

where x_t is the input device sample at time index t, N is the sampling rate, i is the imaginary unit, and k is the monitored frequency such that integer $k \in [0, N/2]$.

We should not, however, use Fourier coefficient $S_k(t)$ directly in our analysis because of spectral leakage. Human motion occurs over a continuum of frequencies, not just on input device harmonics, and so non-integer frequencies discretely sampled "leak" into neighboring bins. A user unable to hold perfectly still while the system estimates noise will inflate the variance estimate. We solve this problem by employing a Hann window, which one can compute with three SDFTs [13] as follows:

$$\mathbb{S}_k(t) = \frac{1}{2} S_k(t) - \frac{1}{4} S_{k-1}(t) - \frac{1}{4} S_{k+1}(t). \tag{2}$$

With the corrected Fourier coefficient $\mathbb{S}_k$ in hand, we then use Welch's method [15, 37] to estimate PSD ϕ_k for frequency k:

$$\hat{\phi}_k = \hat{\sigma}^2 = \frac{1}{W(T-N)} \sum_{t=N}^{T} |\mathbb{S}_k(t)|^2, \tag{3}$$

where W is the Hann window normalization factor given by:

$$W = \sum_{n=0}^{N-1} \left(\frac{1}{2} \left[1 - \cos\left(\frac{2\pi n}{N-1} \right) \right] \right)^2. \tag{4}$$

Notice that the PSD estimate $\hat{\phi}_k$ begins after collecting N samples, the time needed to prime a one second SDFT window when the sampling rate is N.

Since the PSD $\hat{\phi}_k$ estimate also yields noise variance ($\mathbb{E}[\phi_k] = \sigma^2$ [32]), we only need to monitor some frequency k until its estimate converges. Or to improve convergence speed, we can monitor and average together multiple frequencies. One practical problem is that it may be difficult for users to hold still during the calibration process as they adjust their postures or sway; thus Pitch Pipe implementations ought to allow for low frequency movement. Since most human motion falls below 10 Hz [42], we recommend that one utilizes all frequencies $k \in [11, N/2]$.

Step 2: Estimate Maximum Speed. User speed impacts lag,

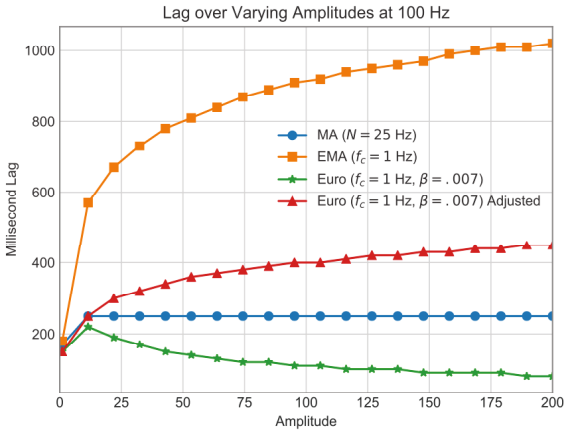

Figure 1: Response time in milliseconds over varying edge ampltidues for different low-pass filters, illustrating the impact of user speed on lag.

though the exact relationship is filter and parameter dependent. We illustrate four examples in Figure 1, where it can be seen that lag depends on the delta between two input samples, i.e., movement amplitude. Specifically, we measure how long it takes for each filtered response to reach ground truth. Not shown, yet intuitive, is that the response curves also vary with parameter selection. Therefore, in calibrating a filter we must estimate a user's maximum speed. Two ways of doing this are to apply a-priori knowledge regarding the application domain, or alternatively measure the user engaged in a representative task. As an example, in our user study, we ask each participant to complete one point and click task round. During this time we record the distance between input samples, discard outliers, and take the largest of what remains as an estimate of maximum user speed.

Step 3: Select Filter Parameters. Once noise variance and maximum speed are known, we proceed to calibrate the filter, but in order to do so, we must understand how a given parameter set affects precision and lag. We choose to estimate precision through Monte Carlo simulation—to generate a normally distributed random sample with variance equal to that of the noise level measured in Step 1 (for instance, by using the Box-Muller transform [25]). Thereafter, we pass the random sample through a filter configured with the specified parameter set and measure the filtered sample's variance, which is taken as an estimate of the filter's precision. For filters with a constant phase shift, we could instead perform a Fourier transform on the edge response, convolve with constant variance, and measure the resulting power; however this method only works in a limited number of cases. Thus, our choice to use simulation in estimating precision was driven by our desire for simplicity and generality.

Next, to estimate lag with the same level of simplicity, we measure a given filter's edge response as discussed above. That is, we generate a synthetic input signal where the first sample is zero and all remaining samples are set to the estimated maximum speed found in Step 2. This signal is then passed through the filter until its output reaches the desired response within a level of tolerance, and we treat this time to convergence as an estimate of lag.

One may also choose to deal with ringing and overshooting, which is a filter-dependent issue. Double exponential smoothing [18], for example, tends to overshoot during rapid deceleration and so when estimating lag, to reach the desired response also means to wait until the system state is settled. Another example is that the 1€ filter momentarily rings after an abrupt stop, depending on how its parameters are set. One can account for these effects in the lag estimate by including the time it takes for the filter's response to stabilize. The effect of this can be seen in the adjusted 1€ filter's response time in Figure 1.

Now that we have a way to estimate precision and lag, we calibrate the filter. Pitch Pipe uses a simple grid search over the parameter space to find those settings that meet application specific requirements. First, given a target level of precision, we find all parameter sets that meet the requirement. From those selected, we then choose that option which minimizes latency. If Pitch Pipe is unable to find a solution, then we raise the target precision and continue as such until a solution is identified, which is not a problem we encountered. A second constraint that an application may provide is maximum lag. If the optimal solution's latency is below this second threshold, then Pitch Pipe can search for a tighter target precision. In this way, we can balance jitter and lag, but in a way that always prioritizes precision.

4 Evaluation

Inspired by Pavlovych and Stuerzlinger [23] who studied the impact of jitter and latency on pointing tasks of varying difficulty, we similarly evaluated Pitch Pipe using a web-based tool conforming to the ISO 9241-9 [31] international standard. We choose to use this well known approach because it allows us to evaluate the impact of various calibration techniques on throughput and accuracy using a standard methodology. In our evaluation, we specifically hypothesize the following:

- **H1**: Pitch Pipe will reduce the amount of time spent tuning parameters
- **H2**: Pitch Pipe will perform as well as manual calibration in throughout and accuracy

In the remainder of this section, we discuss our Pitch Pipe implementation for 1€ [5] (a popular, state-of-the-art low-pass filter), as well as our user study design and protocol.

4.1 Low-pass Filter and Pitch Pipe Effectuation

For logistical reasons, we decided to conduct our user study using a single filter, for which there are a number of commonly used options to choose from, such as the moving average, exponential moving average, double exponential moving average, and Kalman filters [18, 29]. We chose to evaluate Pitch Pitch using the 1€ filter [5] because of its high precision and responsiveness. Unlike other options, 1€ responds to changes in velocity, where at faster speeds, 1€ less aggressively filters so as to reduce lag. Formally, we have:

$$\boldsymbol{y}_t = \alpha_t \boldsymbol{x}_t + (1-\alpha_t)\,\boldsymbol{y}_{t-1} \tag{5}$$

$$\alpha_t = 1 - \exp\left\{-2\pi \frac{f_{c_t}}{f_s}\right\} \tag{6}$$

$$f_{c_t} = f_{c_{min}} + \beta\,|\dot{\boldsymbol{y}}_t|\,, \tag{7}$$

where $\boldsymbol{y}_t$ is the filtered response at time t, $\boldsymbol{x}_t$ is the raw input device sample, f_s is the sampling frequency, and α_t is a smoothing parameterbased on the dynamic cutoff frequency f_{c_t}. Per Equation 7, this latter parameter has a minimum cutoff $f_{c_{min}}$ that increases linearly with speed—the absolute value of the smoothed derivate $\dot{\boldsymbol{y}}_t$ (see [5] more information). As defined, there are two free parameters one must tune: the minimum cutoff $f_{c_{min}}$ and slope β parameters.

Since our evaluation is web-based, we implemented Pitch Pipe in JavaScript. One issue with using an interpreted language and software deployed over unknown systems is uncertainty in computational power. Recall that we use Monte Carlo simulation to estimate the precision of a given parameter set, which may be too slow for some platforms. Therefore, we decided to precompute a lookup table over varying noise levels, minimum cutoff frequencies $f_{c_{min}}$, and slopes β, for which we then use straightforward trilinear

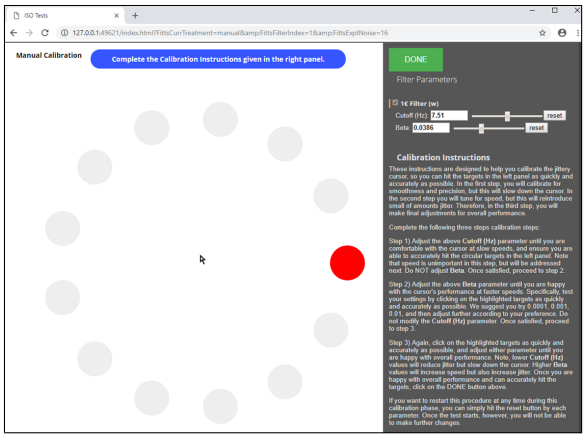

Figure 2: Web-based interface for user study. An example Fitts' task ring is shown in the left white panel, and manual calibration controls are shown in the right panel. A participant is required to click on the highlighted (red) target as quickly and acccuractely as possible, as the taret moves around the ring.

interpolation to estimate precision. For lag estimation, we measure the edge response as previously described.

Our application-specific requirements are designed to maximize throughput. Since the smallest target in our evaluation is fourteen pixels, we set our application-specific target precision to 3.5 pixels. This decision was informed by prior work [23], where we find that if jitter is within approximately 25% of the target size, its impact on accuracy is minimal. We further set lag to 50 ms. Pitch Pipe prioritizes accuracy and selects the least lag parameter set that falls within the specified precision requirement. If there are multiple options under the target lag, then the most precise variant is selected, which in our testing only occurs with the lowest noise condition.

4.2 Apparatus

We developed a JavaScript web-based application conforming to ISO 9241-9 [23] that we deployed through Amazon Mechanical Turk (MTurk)[1]. For a given noise level, participants were asked to complete a series of Fitts' tasks over varying complexities and calibration modes. Specifically, we presented fourteen evenly spaces targets around a ring specified by amplitude (diameter) and width (target size) as shown in Figure 2. Participants were asked to click on the highlighted target as quickly and accurately as possible, where after each click, the highlighted target moved to the opposite side of the ring, which continued on until all targets were visited. Our software recorded position and time stamp information for each click, in order to calculate movement time, accuracy, and throughput.

4.3 Protocol

Participants first read a description of the study, after which we collected standard demographic information. Those using a track pad or who had an inadequate screen resolutions ($<$ 900 px height) were not allowed to continue. We then presented a Fitts' task test block comprising twelve rounds of varying task complexity, where the second six rounds duplicate the first six in random order, and the first are discarded as practice. We use this first test block to establish baseline performance, as the signal is noise free, and to cull individuals who are unable to perform the tasks (those who miss half or more of the targets in any round, which is well above expectation [23]). After establishing baseline performance, a participant completes three more blocks but with simulated noise added to the signal. Each block tests a different calibration mode: unfiltered, manual, and Pitch Pipe.

[1] https://www.mturk.com/

To support manual calibration, we included a panel with detailed instructions and two parameter controls, one for minimum cutoff fc_{min} and another for slope β, which we presented as the Error and Speed control parameters, respectively. Participants were allowed to practice and modify the parameters until they were happy with overall performance. For automatic calibration, the system adjusted these same parameters using Pitch Pipe.

4.3.1 User Feedback

Using a 7-point Likert scale, we asked each participant to rate their satisfaction with the cursor's speed, accuracy, ease of completing the tasks, and overall performance. Additionally, we asked each user to rate their perceived ease and confidence in manual calibration, and to select their overall preference.

4.4 Design

Our experiment was a mixed, four factor repeated measures design. The four nominal factors follow, where the first factor is between-subjects and remaining are within:

- **Jitter**: ±4, ±8, ±12, and ±16 pixels
- **Calibration**: Baseline, Unfiltered, Manual, and Pitch Pipe
- Target **Width**: 14, 32 or 71 px
- Target **Amplitude**: 372 or 589 px

Width and amplitude were uniformly spaced to create a task difficulty range typical of those found in a desktop environment, and the jitter levels follow those measured across various HCI hardware configurations [23]. The dependent variables were throughput and accuracy. In this work, we treat misses as erasures [9] (lost information) and calculate throughput as follows:

$$TP = \frac{(1-\varepsilon)\log_2\left(\frac{A}{W}+1\right)}{MT}, \tag{8}$$

where ε is the miss rate for a single round of targets on an A pixel diameter ring comprising W pixel width targets, and MT is the average movement time between clicks.

We recruited a total of 80 participants through MTurk, with 20 individuals assigned to each jitter level. There were 47 males and 33 females, with an average age of 36.5 ($\sigma = 9.7$). Each participant was compensated $4 for approximately 40 minutes of time.

5 Results

5.1 Quantitative Results

We measured participant speed during calibration and found large variability ($\mu = 88$ pixels per s, $\sigma = 32$), which may be due to user proclivities, OS mouse transfer functions and other complex system interaction effects. This result supports the notion that an automatic calibration system requires user specific information and static, off-line, parameters may not fit variations in-the-wild. We also show the distribution of the minimum cutoff fc_{min} and slope β parameters in Table 1 over the varying jitter levels. One will notice manual calibration results in high variability of parameter selection. With respect to Pitch Pipe, we see a consistent parameter selection behavior that gracefully becomes more restrictive as noise increases. Finally, the mean of all noise variance estimates (not shown) are within 2.5% (σ=2.0%) of ground truth.

We also recorded the amount of time each subject spent in the manual and automatic calibration phases. A pairwise t-test, assuming unequal variance, revealed a significant difference between averages ($t_{79} = 1.99$, $p < .0001$). We observed that Pitch Pipe calibration was consistently low ($\mu = 45.03$ s, $\sigma = 28.79$) while manual calibration took a variable amount of time that was higher on average ($\mu = 200.29$ s, $\sigma = 178.42$).

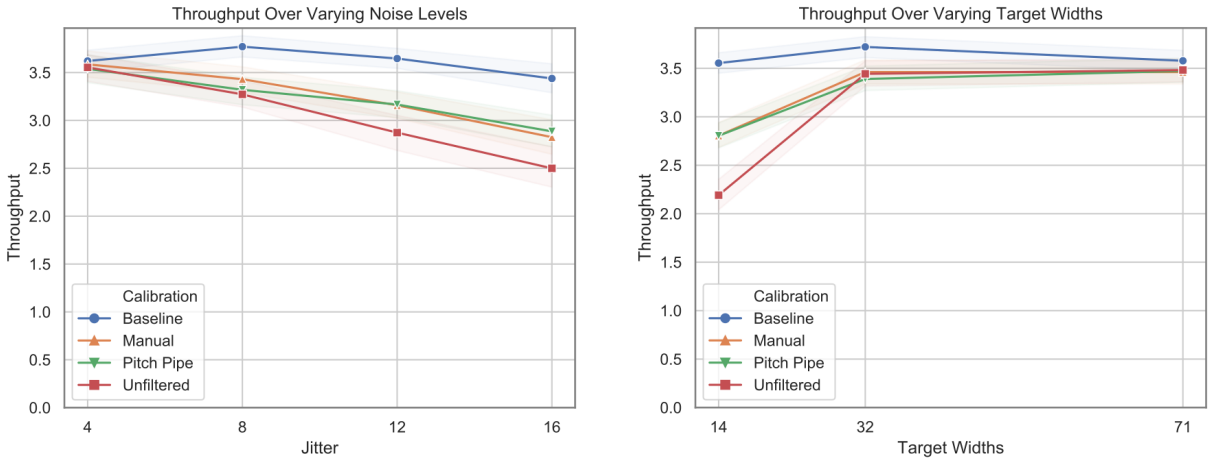

Figure 3: Throughput with 95% confidence interval bands over varying noise levels (left) and target widths (right).

Table 1: Distribution of parameters for each calibration method across varying jitter levels, reported as mean (standard deviation).

	$f_{c_{min}}$		β	
Jitter	PP	Manual	PP	Manual
4	.107 (.033)	4.82 (4.70)	.03 (.007)	.034 (.029)
8	.138 (.117)	4.03 (4.11)	.029 (.007)	.042 (.035)
12	.462 (.163)	2.85 (4.09)	.013 (.002)	.026 (.027)
16	.223 (.051)	2.53 (3.74)	.005 (.0005)	.015 (.018)

Table 2: ANOVA Results

Factor	**Throughput**	**Miss Ratio**
Tuning	$F_{3,1865} = 53.76, p < .0001$	$F_{3,237} = 254.1, p < .0001$
Jitter	$F_{3,79} = 4.866, p < .004$	$F_{3,79} = 12.02, p < .0001$
Width	$F_{2,1865} = 191.5, p < .0001$	$F_{2,158} = 793.8, p < .0001$
T x J	$F_{9,1865} = 5.85, p < .0001$	$F_{9,237} = 35.54, p < .0001$
T x W	$F_{6,1865} = 75.81, p < .0001$	$F_{6,474} = 145.144, p < .0001$
J x W	$F_{6,1865} = 6.612, p < .0001$	$F_{6,158} = 26.76, p < .0001$
T x J x W	$F_{18,1865} = 6.346, p < .0001$	$F_{18,474} = 20.943, p < .0001$

5.2 Throughput and Accuracy

As mentioned in the experimental design we captured throughput and miss ratio data from each participant. We analyzed these results using 3-way repeated measures ANOVA, with calibration, jitter, and width being the independent variables, and miss ratio and throughput, the dependent variables.

5.2.1 Throughput

We ran a three factor within-subjects ANOVA on the throughput, shown in Table 2, and found statistical significance on the calibration method with width and jitter. We further found statistical significance across all interactions between calibration, width and jitter. For our post-hoc analysis we ran a linear hypothesis, shown in Table 3, and found no statistical significance between the Pitch Pipe and manual calibration methods, written as tuning to save space in the tables. All other permutations showed statistical significance on the throughput.

Our post-hoc analysis shows that as noise level increases, unfiltered throughput falls below the filtered methods (Figure 3). Baseline achieved the highest throughput across all conditions, which is expected since no jitter was introduced. Over the varying target widths, as shown in Figure 3, unfiltered throughput approaches the filtered methods after target width increases, which is expected because jitter is less problematic on larger targets and low-pass filters introduce small amounts of latency.

Table 3: Post-hoc Analysis

	Throughput		**Miss Ratio**	
Tuning Permutation	Z score	P Value	T score	P Value
Pitch Pipe - Manual	1.449	.469	1.359	.5263
Pitch Pipe - Unfiltered	-8.828	$< .0001$	-17.752	$< .0001$
Pitch Pipe - Baseline	-7.522	$< .0001$	-9.416	$< .0001$
Manual - Unfiltered	-7.379	$< .0001$	-16.393	$< .0001$
Manual - Baseline	-8.971	$< .0001$	-10.775	$< .0001$
Baseline - Unfiltered	-16.35	$< .0001$	-27.168	$< .0001$

Interactions were present between the calibration method, width, and jitter factors, however, post-hoc analysis shows there was no difference between manual and Pitch Pipe calibration across these conditions.

5.2.2 Accuracy

We ran a three factor within subjects ANOVA on the miss ratio (results in Table 2) and found statistical significance on the calibration method, width, and jitter factors. All permutations of interactions showed statistical significance. We performed a post-hoc analysis, shown in Table 3, which saw no significant difference in the miss ratio between Pitch Pipe and manual calibration. Every other permutation of tuning showed significant difference.

Post-hoc analysis shows that as the noise increases, the miss ratio for the unfiltered method has a linear increase (Figure 4). The manual and Pitch Pipe calibration methods showed consistent miss ratios as jitter increased, demonstrating the effectiveness of low-pass filtering. In Figure 4, we see that the miss ratio increases drastically as the target widths grow smaller, and all methods converge for the largest target width used. Both calibration methods remain nearly identical and show improvement over the unfiltered response. Post-hoc testing confirms that each calibration interaction follows a unique trend, except for manual and Pitch Pipe calibration, which follow similar trends.

This data reveals that while the filtering does not achieve baseline performance, there is a significant improvement in accuracy over using raw, unfiltered data. More importantly, the automated calibration method introduced in this paper was able to meet the

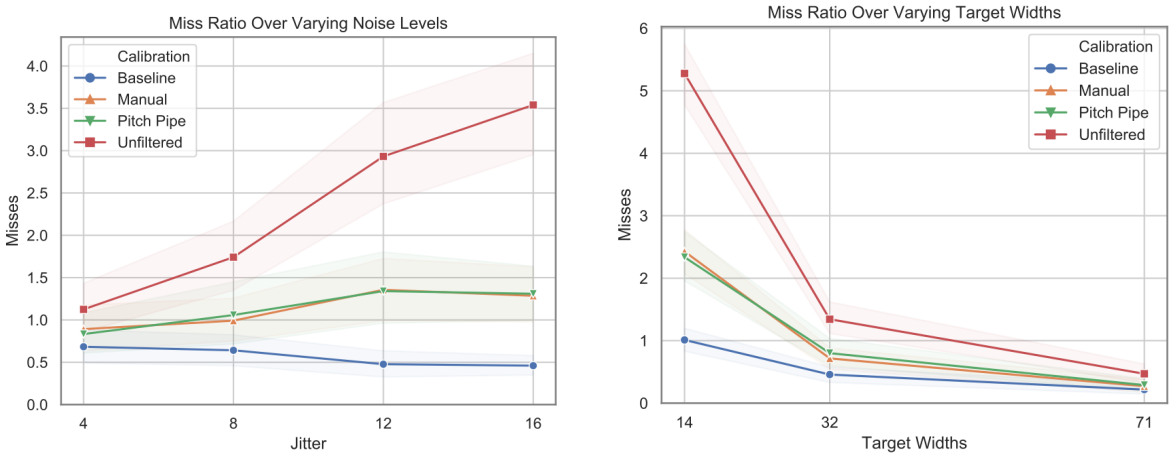

Figure 4: Miss ratio with 95% confidence interval bands over varying noise levels (left) and target widths (right).

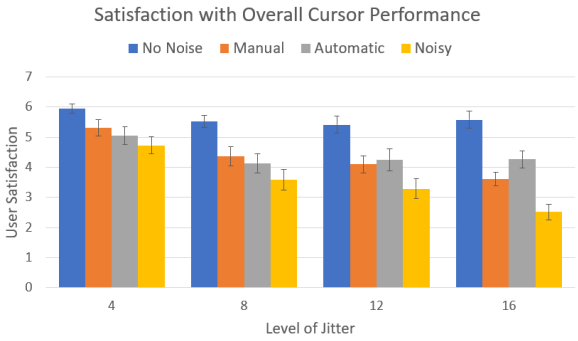

Figure 5: User Satisfaction for different mouse calibrations, ordered by Jitter level. Error bars are 95% confidence intervals.

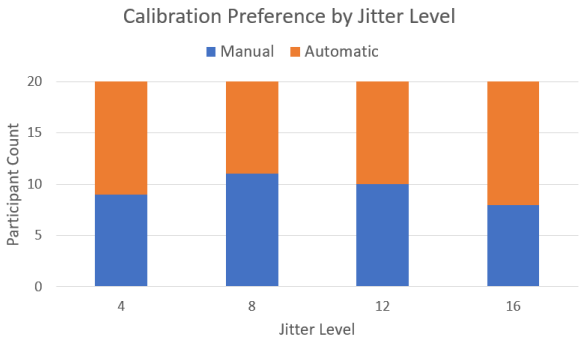

Figure 6: User Calibration Preference by Jitter Level. As jitter increases, users prefer automatic tuning methods.

accuracy of the manual tuning method. Overall, our quantitative data demonstrates that the Pitch Pipe tuning method did not significantly change throughput or miss ratio, making it as effective as manual calibration. Furthermore, we demonstrated a significant reduction in the time taken choosing filter parameters with our method versus manual tuning.

5.3 Qualitative Results

We found that our user satisfaction items were internally valid using Cronbach's Alpha ($\alpha = .918$), so we averaged the values together to form an index. With this value, we used the Aligned Rank Transform tool [39] so we could run a repeated measures ANOVA and check for interaction effects between the jitter and calibration factors. As expected, our participants preferred a mouse which had no noise in it, and they did not like the unfiltered cursor; see Figure 5.

5.3.1 ANOVA Results

We performed a repeated measures ANOVA in order to test for significant differences.

Main Effect of Calibration Mode: An ANOVA revealed a significant main effect of calibration mode on user satisfaction ($F_{3,228} = 81.12$, $p < .001$, $\eta_p^2 = .516$). As expected, post-hoc t-tests show significant differences between *No Noise* and all other conditions, as well as between *Noisy* and all other conditions; interestingly, however, we did not find a significant difference between the automatic and manual calibration modes ($M = 0.08, SD = 1.24, p = .568, t(79) = 0.342$).

Main Effect of Jitter Level: An ANOVA revealed a significant main effect of jitter on user user satisfaction ($F_{3,76} = 418.0$, $p < .001$, $\eta_p^2 = .846$). As expected, post-hoc Mann-Whitney U tests show significant differences between the 4 pixels and all others.

Interaction Effect of Calibration $\times$ Jitter: An ANOVA also revealed a significant interaction effect of small effect size between these two factors ($F_{9,\,228} = 4.408$, $p < .001$, $\eta_p^2 = .148$).

5.3.2 Feedback from Participants

Participants indicated through our questionnaires a near-split decision on the preference between calibration types— 38 chose manual, while 42 selected automatic; see Figure 6. We analyzed the open-ended response questions in order to help understand why they selected their favorite; two authors coded their responses, and we found an acceptable level of inter-rater reliability using Cohen's Kappa ($K = .788$). For split decisions, a third author broke the tie.

We found that a number of participants preferred having full control over their system. As one individual noted, *"I like having control. I want to test it out to find what's best."*, and a number of participants enjoyed being able to tune the cursor to their personal taste. On the other hand, some participants did not feel confidence in their ability to select suitable parameters or would rather not deal with the problem. *"For one it was a lot faster than trying to [manually] figure things out. Also i felt more comfortable with the automatic calibrated settings."*. And finally there were a few participants who had difficulty understanding how to calibrate the filter, despite our

best efforts to provide clear instructions. *"The manual calibration was a little confusing and I would prefer it calibrate automatically."*. These results show the need for an automatic, custom calibration solution in creating a quality user experience for those who do not want to take the time or who do not feel confident in their ability to calibrate their system.

6 Discussion

Applications that track human input for selection, manipulation, navigation, and gesture recognition often use filters. Yet, optimal filter parameters depend on orientation, position, clothing, skin color, physical size, intensity, extension, environmental factors and application objectives, most of which can vary between individuals and runs, impacting signal noise and amplitude in subtle ways. We see this, for example, when tracking motion through depth cameras, where some users are more difficult to track than others, and tuning the system uniquely for each case is cumbersome. Automatic calibration can help save time and instill confidence in users that a system is correctly figured. That is, when software is deployed into an unknown environment, we might expect that non-expert users will have to calibrate the system, many of whom will be unwilling or unsatisfied with manual parameter tuning. For these users, automatic custom calibration software is of high value.

Our solution, Pitch Pipe, begins to address this problem by estimating noise via power spectral analysis and energy by measuring maximum speed on a representative task. We then use these estimates to select those filter parameters that meet application dependent requirements. Since our first iteration of Pitch Pipe specifically addresses pointing tasks, we tune the filter so as to minimize latency (lag) for a given level of precision.

Our evaluation of Pitch Pipe on 2D pointing tasks over varying task complexities highlights a clear benefit in automatic calibration. We first observe that across all conditions Pitch Pipe statistically performs at least as well as manual calibration in both throughout and accuracy. As expected, both approaches improve performance for small and medium sized targets when jitter is beyond negligible ($> \pm 4$ pixels), but Pitch Pipe achieves this without the hassle of a sometimes tricky trial and error parameter selection process. Consequently, automatic calibration is significantly faster on average.

Despite little difference in performance, we do observe a user preference dichotomy in that some users prefer to fine tune their filter, while others claim an intelligent system ought to solve this problem for them. Those who prefer manual calibration feel empowered and able to acquire a level of responsiveness suitable to their taste. Remaining users found calibration cumbersome. As one participant wrote *'I didn't realize how well the auto calibration worked until I tried the manual version, I felt like I was just going back and forth with the manual and making the mouse worse rather than better."*

Regardless of preference, we note that automatic parameter selection can be complementary to manual calibration. For example, one can use Pitch Pipe to recommend initial parameter settings that a power user can thereafter modify to fit their personal taste. Even when an application designer chooses to support manual calibration exclusively, techniques discussed herein enables one to present parameters in an alternative, human understandable form. For instance, given a simple moving average filter, the user interface may present a slider that configures window size, but rather than report a somewhat non-obvious integer value, the interface can also report estimated lag in milliseconds and precision in pixels or millimeters, two human relatable measures.

6.1 Limitations and Future Work

One limitation is that our evaluation lacks ecological validity. First, noise profiles may vary with position, movement, and time rather than manifest as pure static white noise. Second, we use a priori knowledge from prior work [23] to set Pitch Pipe's target precision, which may not be appropriate for all applications and gives Pitch Pipe an advantage that is unknown to the participant. Last, there may more efficient mechanisms available for manual tuning, such as to combine mouse and keyboard input so that participants are not forced to maneuver between two panels, thereby reducing the parameter selection time. Despite these limitations, we favored simulation and a simple design in order to test our method in the wild, so that we may reach a large and varied participant pool.

Not all applications are designed for or care about pointing tasks, which is one limitation of our current approach. Gesture recognizers also benefit from filtering so as to reduce variance in their feature and similarity measures. Since a number of techniques involve shape analysis, e.g., [26, 33, 40], constant phase shift may be more important than latency or precision; and an early Pitch Pipe prototype for gesture recognition that optimizes for shape rather than latency shows promise of being able to find optimal parameters at run time.

Perhaps even more compelling is dynamic automatic calibration. Suppose one uses a controller-based pointing device in a virtual environment that has 0.25 degrees noise, mean-to-peak. Their pointer projected onto a virtual surface 2m away will result in approximately 7mm of jitter, and at 10m this error scales to 34mm, which makes precise selection of small targets at a distance quite difficult. An automatic method could adjust filter parameters to accommodate for increases in error as a function of distance. Or a calibrator could react to those objects that are in the neighborhood of a pointer, so that as one approaches a small object, the system reduces jitter, becoming more precise.

7 Conclusion

We presented a novel automatic calibration technique for pointing tasks called Pitch Pipe. In three straightforward steps, our approach is able to tune one's filter to user and application dependent criteria, which greatly simplifies the use and deployment of systems that leverage low-pass filters. Through an Amazon MTurk based user study, we conducted an evaluation of Pitch Pipe over a set of 2D pointing task of varying difficulty, noise levels, and calibration techniques using a state-of-the-art low-pass filter (1€ [5]). We found Pitch Pipe performed as least as well as manual calibration in throughput and accuracy. Our automatic solution further fit the needs of those who lacked confidence in their ability to select good parameters or were unwilling to deal with the hassle of a hands on experience. We believe these findings motivate and show the pragmatic benefits of Pitch Pipe.

Acknowledgments

This work is supported in part by NSF Award IIS-1638060, Lockheed Martin, Office of Naval Research Award ONRBAA15001, and Army RDECOM Award W911QX13C0052. We also thank the anonymous reviewers for their insightful feedback.

References

[1] M. S. Ahmad, O. Kukrer, and A. Hocanin. Recursive inverse adaptive filtering algorithm. *Digital Signal Processing*, 21(4):491–496, 2011.

[2] Z. A. Bhotto and A. Antoniou. A family of shrinkage adaptive-filtering algorithms. *IEEE Transactions on Signal Processing*, 61(7):1689–1697, 2013.

[3] A. D. Borah, A. J. Mondal, D. Muchahary, and A. Majumder. Fir low pass filter design using craziness base particle swarm optimization technique. In *Communications and Signal Processing (ICCSP), 2015 International Conference on*, pp. 0870–0874. IEEE, 2015.

[4] A. Bulling, D. Roggen, and G. Tröster. It's in your eyes: towards context-awareness and mobile hci using wearable eog goggles. In *Proceedings of the 10th international conference on Ubiquitous computing*, pp. 84–93. ACM, 2008.

[5] G. Casiez, N. Roussel, and D. Vogel. 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, pp. 2527–2530. ACM, 2012.

[6] D. Chen, D. Xue, and F. Pan. An improved median filter based on automatic parameter tuning approach. In *Mechatronics and Automation, 2007. ICMA 2007. International Conference on*, pp. 1305–1309. IEEE, 2007.

[7] R. Dabral, A. Mundhada, U. Kusupati, S. Afaque, and A. Jain. Structure-aware and temporally coherent 3d human pose estimation. *arXiv preprint arXiv:1711.09250*, 2017.

[8] A. M. Feit, S. Williams, A. Toledo, A. Paradiso, H. Kulkarni, S. Kane, and M. R. Morris. Toward everyday gaze input: Accuracy and precision of eye tracking and implications for design. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pp. 1118–1130. ACM, 2017.

[9] J. Gori, O. Rioul, and Y. Guiard. Speed-accuracy tradeoff: A formal information-theoretic transmission scheme (fitts). *ACM Trans. Comput.-Hum. Interact.*, 25(5):27:1–27:33, Sept. 2018. doi: 10.1145/3231595

[10] P. Harscher, R. Vahldieck, and S. Amari. Automated filter tuning using generalized low-pass prototype networks and gradient-based parameter extraction. *IEEE transactions on microwave theory and techniques*, 49(12):2532–2538, 2001.

[11] K. Horváth and M. Kuslits. Optimization-based parameter tuning of unscented kalman filter for speed sensorless state estimation of induction machines. In *Electrical and Electronics Engineering (ISEEE), 2017 5th International Symposium on*, pp. 1–7. IEEE, 2017.

[12] F. Huang, J. Zhang, and S. Zhang. Combined-step-size affine projection sign algorithm for robust adaptive filtering in impulsive interference environments. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 63(5):493–497, 2016.

[13] E. Jacobsen and R. Lyons. The sliding dft. *IEEE Signal Processing Magazine*, 20(2):74–80, 2003.

[14] E. Jacobsen and R. Lyons. An update to the sliding dft. *IEEE Signal Processing Magazine*, 21(1):110–111, 2004.

[15] O. M. S. Jr. Psd computations using welch's method. *NASA STI/Recon Technical Report N*, 92, 1991.

[16] R. Kaluri and C. Pradeep Reddy. A framework for sign gesture recognition using improved genetic algorithm and adaptive filter. *Cogent Engineering*, 3(1):1251730, 2016.

[17] S. S. Kozat, A. T. Erdogan, A. C. Singer, and A. H. Sayed. Steady-state mse performance analysis of mixture approaches to adaptive filtering. *IEEE Transactions on Signal Processing*, 58(8):4050–4063, 2010.

[18] J. J. LaViola. Double exponential smoothing: an alternative to kalman filter-based predictive tracking. In *Proceedings of the workshop on Virtual environments 2003*, pp. 199–206. ACM, 2003.

[19] J.-W. Lee and G.-K. Lee. Design of an adaptive filter with a dynamic structure for ecg signal processing. *International Journal of Control, Automation, and Systems*, 3(1):137–142, 2005.

[20] H. Liang, J. Yuan, D. Thalmann, and N. M. Thalmann. Ar in hand: Egocentric palm pose tracking and gesture recognition for augmented reality applications. In *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 743–744. ACM, 2015.

[21] K. A. Martin and J. J. Laviola. The transreality interaction platform: Enabling interaction across physical and virtual reality. In *Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2016 IEEE International Conference on*, pp. 177–186. IEEE, 2016.

[22] A. C. Mugdha, F. S. Rawnaque, and M. U. Ahmed. A study of recursive least squares (rls) adaptive filter algorithm in noise removal from ecg signals. In *Informatics, Electronics & Vision (ICIEV), 2015 International Conference on*, pp. 1–6. IEEE, 2015.

[23] A. Pavlovych and W. Stuerzlinger. The tradeoff between spatial jitter and latency in pointing tasks. In *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pp. 187–196. ACM, 2009.

[24] S. Poularakis, G. Tsagkatakis, P. Tsakalides, and I. Katsavounidis. Sparse representations for hand gesture recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 3746–3750. IEEE, 2013.

[25] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in C*, vol. 2. Cambridge University Press, New York, NY, USA, 1996.

[26] D. Rubine. Specifying gestures by example. *SIGGRAPH Computer Graphics*, 25(4):329–337, July 1991.

[27] F. Russo. Technique for image denoising based on adaptive piecewise linear filters and automatic parameter tuning. *IEEE transactions on instrumentation and measurement*, 55(4):1362–1367, 2006.

[28] S. K. Saha, S. Sarkar, R. Kar, D. Mandal, and S. Ghoshal. Digital stable iir low pass filter optimization using particle swarm optimization with improved inertia weight. In *Computer Science and Software Engineering (JCSSE), 2012 International Joint Conference on*, pp. 147–152. IEEE, 2012.

[29] S. W. Smith. *The scientist and engineer's guide to digital signal processing*. California Technical Pub. San Diego, 1997.

[30] S.-W. Sohn, Y.-B. Lim, J.-J. Yun, H. Choi, and H.-D. Bae. A filter bank and a self-tuning adaptive filter for the harmonic and interharmonic estimation in power signals. *IEEE Transactions on Instrumentation and Measurement*, 61(1):64–73, 2012.

[31] R. W. Soukoreff and I. S. MacKenzie. Towards a standard for pointing device evaluation, perspectives on 27 years of fitts' law research in hci. *International Journal of Human-Computer Studies*, 61(6):751 – 789, 2004. Fitts' law 50 years later: applications and contributions from human-computer interaction. doi: 10.1016/j.ijhcs.2004.09.001

[32] P. Stoica and R. L. Moses. *Spectral analysis of signals*, vol. 1. Pearson Prentice Hall Upper Saddle River, NJ, 2005.

[33] E. M. Taranta II, A. Samiei, M. Maghoumi, P. Khaloo, C. R. Pittman, and J. J. LaViola Jr. Jackknife: A reliable recognizer with few samples and many modalities. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pp. 5850–5861. ACM, New York, NY, USA, 2017.

[34] E. Trabes and M. A. Jordan. Self-tuning of a sunlight-deflickering filter for moving scenes underwater. In *Information Processing and Control (RPIC), 2015 XVI Workshop on*, pp. 1–6. IEEE, 2015.

[35] R.-D. Vatavu, L. Anthony, and J. O. Wobbrock. Gestures as point clouds: A $p recognizer for user interface prototypes. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction*, ICMI '12, pp. 273–280. ACM, New York, NY, USA, 2012.

[36] G. F. Welch. History: The use of the kalman filter for human motion tracking in virtual reality. *Presence: Teleoperators and Virtual Environments*, 18(1):72–91, 2009.

[37] P. Welch. The use of fast fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms. *IEEE Transactions on audio and electroacoustics*, 15(2):70–73, 1967.

[38] A. D. Wilson. Sensor-and recognition-based input for interaction. *The Human Computer Interaction Handbook*, pp. 177–200, 2007.

[39] J. O. Wobbrock, L. Findlater, D. Gergle, and J. J. Higgins. The aligned rank transform for nonparametric factorial analyses using only anova procedures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pp. 143–146. ACM, New York, NY, USA, 2011.

[40] J. O. Wobbrock, A. D. Wilson, and Y. Li. Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, pp. 159–168. ACM, New York, NY, USA, 2007.

[41] R. Xiao, C. Harrison, K. D. Willis, I. Poupyrev, and S. E. Hudson. Lumitrack: low cost, high precision, high speed tracking with projected m-sequences. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pp. 3–12. ACM, 2013.

[42] H. Zeng and Y. Zhao. Sensing movement: Microsensors for body motion measurement. *Sensors*, 11(1):638–660, 2011.

[43] Z. Zhu, V. Branzoi, M. Sizintsev, N. Vitovitch, T. Oskiper, R. Villamil, A. Chaudhry, S. Samarasekera, and R. Kumar. Ar-weapon: live augmented reality based first-person shooting system. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*, pp. 618–625. IEEE, 2015.