N-euro Predictor: A Neural Network Approach for Smoothing and Predicting Motion Trajectory

QIJIA SHAO[†*], Columbia University, USA
JIAN WANG[*], Snap Inc., USA
BING ZHOU, Snap Inc., USA
VU AN TRAN, Snap Inc., USA
GURUNANDAN KRISHNAN, Snap Inc., USA
SHREE NAYAR, Snap Inc., USA

Jitter and lag severely impact the smoothness and responsiveness of user experience on vision-based human-display interactive systems such as phones, TVs, and VR/AR. Current manually-tuned filters for smoothing and predicting motion trajectory struggle to effectively address both issues, especially for applications that have a large range of movement speed. To overcome this, we introduce *N-euro*, a residual-learning-based neural network predictor that can simultaneously reduce jitter and lag while maintaining low computational overhead. Compared to the fine-tuned existing filters, *N-euro* improves prediction performance by 36% and smoothing performance by 42%. We fabricated a Fish Tank VR system and an AR mirror system and conducted a user experience study (n=34) with the real-time implementation of *N-euro*. Our results indicate that the *N-euro* predictor brings a statistically significant improvement in user experience. With its validated effectiveness and usability, we expect this approach to bring a better user experience to various vision-based interactive systems.

CCS Concepts: • **Human-centered computing → Interactive systems and tools**.

Additional Key Words and Phrases: vision-based interactions, motion-to-photon latency, motion prediction, neural network, perceived jitter and lag

ACM Reference Format:
Qijia Shao[†*], Jian Wang[*], Bing Zhou, Vu An Tran, Gurunandan Krishnan, and Shree Nayar. 2023. N-euro Predictor: A Neural Network Approach for Smoothing and Predicting Motion Trajectory. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 3, Article 120 (September 2023), 25 pages. https://doi.org/10.1145/3610884

1 INTRODUCTION

Interacting with displays is becoming a dominant way for humans to process information in our daily lives. Various kinds of displays are ubiquitous around us, from small screen displays (*e.g.*, cell phones, touchpads) to bigger screen displays (*e.g.*, computers, TVs) and to emerging wearable devices (*e.g.*, VR/AR headset). Researchers have

[†]Work done during internship at Snap Research.
[*]Co-primary authors

Authors' addresses: Qijia Shao[†*], Columbia University, New York City, NY, USA, qijia@cs.columbia.edu; Jian Wang[*], Snap Inc., New York City, NY, USA, jwang4@snapchat.com; Bing Zhou, Snap Inc., USA; Vu An Tran, Snap Inc., USA; Gurunandan Krishnan, Snap Inc., USA; Shree Nayar, Snap Inc., USA.

2474-9567/2023/9-ART120 $15.00
https://doi.org/10.1145/3610884

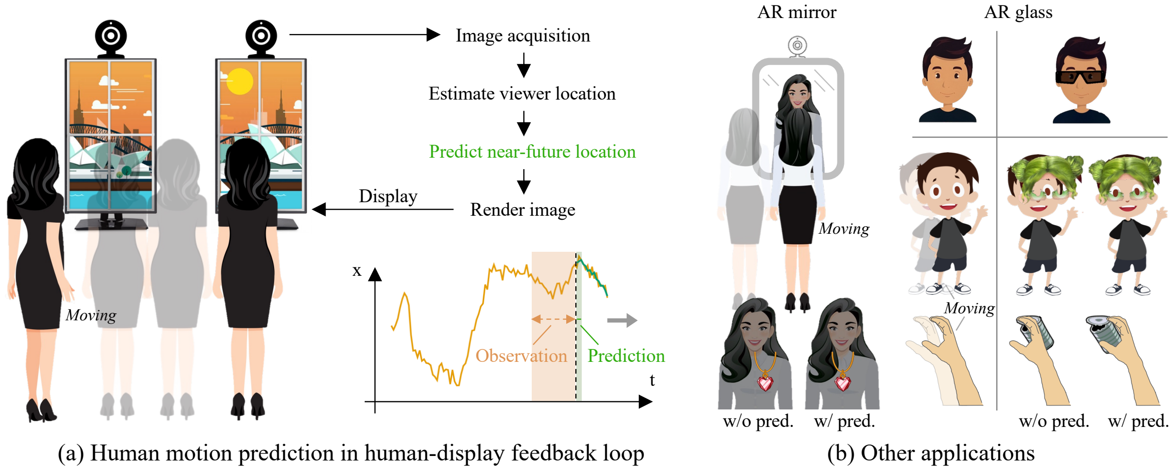


Fig. 1. By mitigating both jitter and lag, *N-euro* enhances the overall user experience. (a) The Digital Window application is utilized as an illustrative example, simulating a real window using a display, to convey the underlying concept of our approach. (b) Other application scenarios are showcased, highlighting the results obtained both without and with the integration of our prediction technique.

been exploring different interaction techniques, either touch-based or touchless, for more intuitive human-display interactions [12, 17, 23, 44, 72].

This paper focuses on vision-based interactive systems (*i.e.*, using cameras as the sensor), particularly those requiring the tracking of human motion trajectories. In vision-based interactive systems, response time is a crucial factor, which reflects how quickly the system can respond to user inputs. The entire sensing-rendering loop involves camera image acquisition (including both exposure and readout), image processing for obtaining user input localization, and corresponding scene rendering on display. The response time is the time from the sensing to the corresponding rendering on display. Depending on the complexity and implementation details of the systems, response time can vary from 10s of ms to 100s of ms. Therefore, by the time the corresponding view rendering is finished, the user may have already moved to another location. Slow response time results in a lag between the user's motion and the perceived pixel changes on the display, significantly impacting the user experience. In VR/AR, this is referred to as Motion-to-Photon (MTP) latency, and a large MTP latency can cause a loss of performance and even provoke cybersickness [73]. Another important factor influencing the user experience of vision-based interactive systems is *jitter*. Jitter could arise from the noise introduced by both the hardware system and the tracking algorithms employed. Specifically, spatial jitter manifests as trembling or instability of perceived pixels on the display, consequently impeding users from seamlessly viewing and interacting with the display content [3].

Prior research has explored the prediction for the endpoints of mouse movements [8, 45, 84], gaze fixation and viewport [6, 24], finger/hand movements [25, 33, 35, 36, 56, 77], and head movements [31, 32, 66]. Most of these works have primarily focused on achieving accurate predictions of final outcomes, such as reaching a target, and their evaluation metrics have centered around prediction accuracy. However, in the context of vision-based interactive systems, the smoothness of the entire motion trajectory, from the starting position to the end position, significantly influences the overall user experience. On the other spectrum, common ways in signal processing to reduce jitter or lag is by adding either a smoothing filter (*e.g.*, moving average filter [38]) or a predictive filter (*e.g.*, Kalman filter [9, 48, 79, 82], double-exponential filter [47]). However, smoothing filters face the *jitter-lag tradeoff* (*i.e.*, less jitter uses a longer looking back window of the historical signals and causes more lag, and less lag has to use a short window and results in more jitter). Predictive filters encounter a similar tradeoff, as they need to calculate speed or acceleration from noisy data. Using a longer window provides smoother results but at the expense of timely predictions, while a shorter window offers quicker responses but with increased jitter. One euro filter [14] is a well-known and widely adopted filter for striking a dynamic balance between the jitter and filter-introduced lag, but it does not make predictions and cannot handle scenarios with high initial system lag.

The aim of this paper is to simultaneously reduce both jitter and lag to enhance the user experience in vision-based interactive systems. To achieve this goal, the system must render the scene based on a smoothed user's location at the time of rendering completion, rather than the user's location derived at the moment of image capture. This is accomplished through a neural network (NN)-based predictor that takes recent historical data as input and outputs near-future predictions. Designing such an NN-based predictor faces challenges on three fronts. First, human motion is non-deterministic and could be arbitrary, making it challenging to predict accurately. Second, the performance of NN-based algorithms is typically highly dependent on the training datasets, which may have low generalization capabilities for different noise levels, motion types, and individuals. Third, an NN-based predictor itself could introduce high computational overhead in terms of both memory size and inference time, potentially leading to even more lag.

We tackle these challenges in the following manner. (1) Although human motion during interaction is non-deterministic, it is confined within certain limits (*e.g.*, movement speed and acceleration are bounded and follow a joint distribution). Human movement speed and acceleration cannot change abruptly. A change in direction must occur through deceleration to zero in the original direction, followed by acceleration in the new direction. By collecting human movement data under different conditions, we propose a network that can learn the overarching constraints of human motion and the short-term dynamics from the data, and predict the near future (2) Instead of feeding raw sensor data, images/videos, directly into the network, we first abstract human motion during the interaction as univariate time series. This way, the inputs to the neural network have higher information density and reduce the domain gap between different people and environments. (3) We prune the network to contain the minimum number of parameters while retaining the ability to predict and smooth motion trajectories, and it can perform inferences within 1 ms.

We compare the proposed N-euro predictor with existing traditional filters, quantitatively showing that *N-euro* predictor provides more accurate predictions while maintaining the highest level of smoothness. To further validate the practical improvement in user experience, we conducted two user studies. In the first study, we implemented a "Digital Window" system that utilized a display to simulate a real window, creating a Fish Tank Virtual Reality (FTVR) setting. N-euro was evaluated against various baseline algorithms, and the results revealed that our N-euro predictor significantly outperformed the alternatives in terms of user experience. By effectively reducing both perceived lag and jitter, our system provided the best user experience for FTVR interactions. In the second study, we employed the "AR-mirror" system to investigate AR interactions. Despite slightly higher perceived jitter compared to the smoothest baseline, participants consistently preferred our *N-euro* due to its substantial reduction in lag. This tradeoff between jitter and lag was well-received by users, underscoring the practical benefits of our approach in AR interaction scenarios.

We summarize our core contributions as follows.

- We developed an NN-based predictor, *N-euro*[1], using a residual-learning approach to address two main issues (jitter and lag) influencing the human-display interaction user experience. The *N-euro* predictor maintains a small network size and minimal computational overhead.

[1] The name "N-euro" is an homage to the One Euro filter [14] and stands as an abbreviation for "Neuro" network-based predictor.

- We quantitatively demonstrate the performance of *N-euro* predictor by comparing it against commonly used smoothing and prediction filters. *N-euro* predictor outperforms all baseline filters by a significant margin, especially for applications involving a wide range of movement speeds and relatively low tracking frequencies. Specifically, the *N-euro* improved the prediction performance by 36% and the smoothing performance by 42% on average.
- We developed two vision-based human-display interactive systems, Digital Window (VR) and AR mirror, to assess the real-world impact of the *N-euro* predictor on the actual user experience. Repeated measures ANOVAs and the post-hoc Tukey tests with 34 participants show a statistically significant improvement of the *N-euro* predictor in reducing jitter and lag. The results demonstrate the favorable impact of the *N-euro* predictor in enhancing the user experience in real VR and AR systems.

2 RELATED WORK

Our work mainly intersects with three areas of research: smoothing and predictive filters, human motion prediction for latency compensation, and time series forecasting.

Smoothing and Predictive Filters. In signal processing, the moving average filter [10, 64, 80] is a commonly used method for smoothing noisy signals. However, it requires a moving window of length *n*, resulting in a lag of up to *n* times the sampling period. The double exponential smoothing filter [26, 81] mitigates the lag issue by assigning exponentially decreasing weight to older data points. The well-known Kalman filter and its variations [9, 48] require prior information about the system generating the signal, which is not always available and hard to assume for human motion. Double exponential and Kalman filters could be used as predictive filters, which means they can predict future point locations. LaViola [47] has shown that a double exponential filter can provide the same level of prediction accuracy as Kalman and extended Kalman filters but with significantly less computational overhead. During the tuning process, we discovered that traditional predictive filters also face the jitter-lag tradeoff, failing to provide optimal results in both metrics. The one euro filter [14] is a well-adopted smoothing filter in the tracking system, which is a first-order low pass filter with an adaptive cutoff frequency based on the speed. It uses a low cutoff frequency at low speed for less jitter and an increased cutoff frequency at high speed for reducing the filter-introduced lag, but it cannot reduce the initial system lag. Furthermore, all the traditional filters mentioned above require hyperparameter tuning for different scenarios. In recent years, neural networks have proven to be more advanced than traditional hand-crafted filters in various fields, such as computer vision [60] and audio processing [63]. The advantage of neural networks is that they can be designed by learning from data and have many parameters to extract features, allowing them to adapt to complex scenarios automatically. With this insight, we designed a neural network-based filter which indeed shows advantages over traditional filters.

Human Motion Prediction for Latency Compensation. Since most interactive systems have a lag between user input and corresponding interface reaction, using prediction to compensate for such lag has been explored in many works in the HCI community. We categorize them based on if the final interactive systems are touch-based or touchless. (1) Touch-based. Heuristics-based/regression-based prediction is a common way of predicting for reducing the perceived delay [8, 84, 87]. Xia et al. [84] proposed to predict touch events by tracking the path of a user's finger as it approaches the display and predicting the location and time of landing. They fit a parabola on the hovering trajectory and linearly extrapolate to the next positions. The Delphian Desktop predicts the user pointing target based on estimating the movement direction and peak velocity [8]. Other rationales for endpoint prediction include inverse control theory [89], and normative kinematic laws [45, 61]. Prediction for the touch-based interactive systems could leverage the human body, arm, and hand information, which is usually not available for touchless interactive systems. TurboTouch predictor [56] leveraged finite-time derivative estimation together with a post-smoothing of the prediction, which demonstrated great performance on both accuracy and smoothness on direct touch tasks. It, however, requires relatively complex parameter tuning for both the general parameters and the optimization-based parameters. Henze et al. [36] proposed to use neural networks (LSTM model) for predicting user inputs on touchscreens. This serves as a reasonable baseline for our work and we compared our proposed filter with it in our evaluation. (2) Touchless. The most representative touchless interactive systems are AR and VR systems, and there is a great amount of work on reducing the motion-to-photon latency by prediction [31, 32, 46, 66, 83]. LaViola mentioned two ways used for head tracking in Oculus VR headsets [46]: assuming either the currently measured angular velocity or the acceleration will remain constant over the latency interval. Other head tracking work leveraged a linear combination of the past values [32], kalman filter [31], and LSTM-based neural network [66]. Hand motion, as the primary method for inputs in AR/VR systems, also plays an important role. Several recent works explored hand motion prediction in AR/VR [25, 33, 77]. Most recently, Gamage et al. [25] proposed a kinematics-based regressive model for continuously predicting ballistic hand movements that work across users and activities. Our work builds upon those works but differs in that we aim to achieve prediction and smoothing at the same time with a tiny overhead.

Time Series Forecasting. There are mainly two categories of approaches to the time series forecasting problem: classic statistical methods and learning-based methods. *(1) Classic statistical methods.* Double exponential smoothing, autoregressive integrated moving average (ARIMA) models, and their extensions are well-known classic time series forecasting methods. ARIMA models have been used for predicting primary energy demand [21] and stock prices [7, 54]. *(2) Learning-based methods.* Recently, neural network-based time series forecasting methods have gained popularity and have been shown to outperform classic statistical methods [15, 41, 49, 58, 69]. N-BEATS [58] was the first to demonstrate that purely deep learning-based architecture can achieve state-of-the-art results in univariate time series forecasting. Temporal Fusion Transformer [49] uses recurrent layers for local processing and self-attention layers to learn temporal relationships, resulting in improved prediction performance at multiple future time steps. It is important to note that general time series forecasting methods do not consider the smoothness of the predicted time series or the running time delay, which are crucial factors in our applications.

3 N-EURO PREDICTOR

The *N-euro* predictor is a neural network-based solution for predicting a smoothed near-future human motion trajectory. Previous work in human dynamics prediction leverages the videos/images as inputs [42, 85, 86], which contains rich information about the motion context (*e.g.*, environment, object). This input format, however, decreases the information density (*e.g.*, many pixels in the image are not useful for prediction) and enlarges the domain gap (*e.g.*, pixel-level statistics are quite different). Therefore it makes it harder for neural networks to learn to predict only the trajectory and generalize well to different scenarios and users outside the training sets, especially with a small network and limited training data.

Instead of using raw sensor data, we first abstract the human motion data captured by sensors into several univariate time series. Human motion can be represented as a group of moving points, with each point's trajectory depicted as the 3D locations over time, which we treat as three separate univariate time series. The learning task can be described as: given T frames of the 3D locations of a certain point on human body $\mathbf{x} = [L_1, ..., L_T] \in R^{3\times T}$ extracted from a k-fps tracking data as inputs, *N-euro* predicts this point's next N frames 3D locations $\mathbf{y} = [L_{T+0}, ..., L_{T+K-1}] \in R^{3\times K}$. Here $K = 1$ represents only smoothing but no prediction.

The goals of the network design are to achieve accurate prediction of the next motion positions, to produce smooth predicted trajectories, and to maintain a small network structure with low inference time for deployment on mobile devices. We next outline how each component of the network has been designed to meet these objectives.

3.1 Supervision for Prediction and Smoothing

Most NN-based time series forecasting techniques are trained in a self-supervised manner [15, 39, 58], where previous time series data is used as the training data and the next time series is used as the label. This approach is typically used when the smoothness of the curve is not a concern, such as with stock prices, which contain little noise. In human motion tracking, however, the time series data is often noisy, due to both the hardware and tracking algorithms used. Therefore, in order to achieve both prediction and smoothing, the smoothed future time series are needed as the ground truth labels.

Pseudo Ground Truth Labeling. Obtaining completely noise-free tracking data is practically difficult. Similar to the previous work [47], we explore the use of pseudo-ground truth labels as a surrogate. The pseudo labels are computed by running a double-sided moving average filter offline with a window size of $2k + 1$ to the raw time series with length n,

$$f(t) = \frac{1}{2k+1} \sum_{j=-k}^{k} L_{t+j}, \tag{1}$$

where $t \in [k, n-k-1]$. The larger value of k, the flatter and smoother the frame will be. Flatter means it is more biased (*i.e.*, deviated more from the original signal), especially near peaks and troughs. So when tuning k, there is a tradeoff between decreasing the variance (larger k) and decreasing bias (smaller k) [38]. The best value of k usually is determined empirically by gradually increasing it and observing the smoothed trajectory until it meets the desired goals. Although the pseudo labels are not as accurate as actual ground truth labels, the easier labeling process makes it a practical solution for various interactive systems, without requiring a high-precision, low-noise tracking system. We will further discuss this point in Section 6.

3.2 Architecture and Loss

Our *N-euro* model structure is largely based on N-BEATS [58], a time series forecasting model leveraging backward and forward residual links. The model consists of multiple stacks, each of which computes backcasting and forecasting weights for pre-defined basis functions and passes the backcasting residual to the next stack. The final forecasting result is the sum of outputs from all stacks. The design of N-BEATS is thought to adapt to local data observations. When *N-euro* observes a small portion of data locally, it can perform self-supervised learning to achieve accurate backcasting. An accurate backcasting boosts forecasting performance.

There are two main differences between *N-euro* and N-BEATS. First, N-BEATS is trained in a self-supervised manner without considering the smoothness of the time series, while we use pseudo ground truth labels and modify the loss function to penalize less smoothed predictions. Second, the original N-BEATS is designed for general predictions in a wide array of target domains, leading to a very deep network with large network size. To fulfill our tiny network size requirement, we leveraged the intrinsic constraints of human movements, pruning the network to preserve only the necessary components while keeping the smoothing and prediction performance.

Architecture. As illustrated in Fig. 2, the whole *N-euro* consists of two blocks. The input to the first block is an $I \times T$ time series, where I (1 or 3) and T represent the spatial and temporal dimensions, respectively. The input segment $\mathbf{x}_1$ is a moving window of historical data with length T that ends with the most recent measurement.

The first block starts by processing the input segment through a fully connected layer with ReLU nonlinearities [55] to extract motion dynamics features $\boldsymbol{\theta}_1$. These features are then sent to both the forward and backward layers to generate a backcast $\widehat{\mathbf{x}_1}$ and a forecast $\widehat{\mathbf{y}_1}$. The backcast $\widehat{\mathbf{x}_1}$ represents the best estimate of the inputs $\mathbf{x}_1$, while the forecast $\widehat{\mathbf{y}_1}$ represents the forecast generated by this layer based on the information captured in $\widehat{\mathbf{x}_1}$. The input to the next block is the residual of the first block, $\mathbf{x}_2 = \mathbf{x}_1$ - $\widehat{\mathbf{x}_1}$. The second block has the same structure as the first block and produces a second forecast element $\widehat{\mathbf{y}_2}$ based on the residual information. In

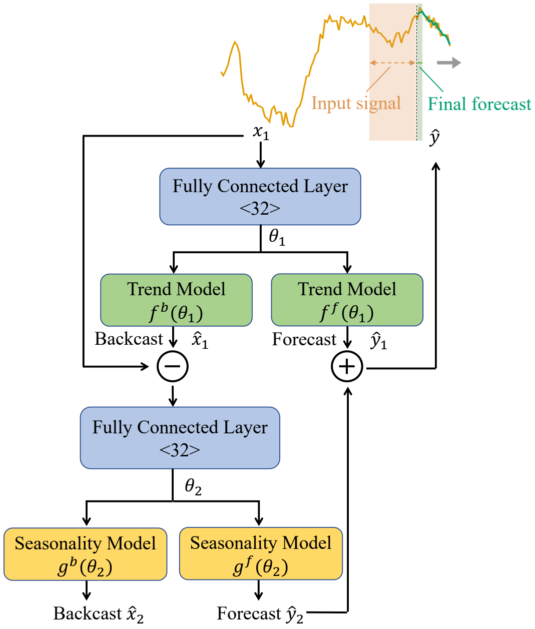

Input signal
Final forecast
x_1
$\hat{y}$
Fully Connected Layer <32>
θ_1
Trend Model $f^b(\theta_1)$
Trend Model $f^f(\theta_1)$
Backcast $\hat{x}_1$
Forecast $\hat{y}_1$
Fully Connected Layer <32>
θ_2
Seasonality Model $g^b(\theta_2)$
Seasonality Model $g^f(\theta_2)$
Backcast $\hat{x}_2$
Forecast $\hat{y}_2$

Fig. 2. Neural network architecture of *N-euro*. The input signal is analyzed sequentially by the fully connected layers with two trend models and two seasonality models [58]. The final prediction is a fusion of two partial predictions, representing the general moving trend and the periodic feature of human motion, respectively. The whole model size is 5 kB with only 1.2 k trainable parameters.

the N-BEATS paper [58], these two blocks together form a basic building block. The authors suggest stacking multiple (*e.g.*, 30 in the original paper) such building blocks for better performance. In a follow-up paper by the same authors [59], it is shown that stacking these building blocks can help learn global characteristics across different tasks. However, since our task involves predicting only human movement, we decided to remove the stacking of building blocks, resulting in a significant reduction in model size. We observed that human movement speed and acceleration cannot change abruptly. Changes in direction must occur through deceleration to zero in the original direction, followed by acceleration in the new direction. Additionally, human walking presents periodic up-and-down bobbing [52]. Based on these observations, we use the trend model [58] for the first block to model the general moving trend (*e.g.*, speed, direction) of the motion, and the seasonality model [58] for the second block to model the periodic feature of human motion. By sequentially separating and analyzing human motion, we can more easily identify the motion trend and then learn the periodic motion features. The final forecast $\widehat{\mathbf{y}}$ is the sum of the two partial forecasts, $\widehat{\mathbf{y}} = \widehat{\mathbf{y}_1} + \widehat{\mathbf{y}_2}$.

Loss Function. *N-euro* predictor aims to minimize the position errors of the prediction while penalizing the less smoothed prediction during the training. The position error is defined as the Euclidean distance between the prediction and the pseudo-ground truth,

$$L_{pos} = \sum_{t=0}^{T-1} \sum_{i=0}^{2} |Y_{i,t} - G_{i,t}|, \tag{2}$$

where $Y_{i,t}$ is the predicted position and $G_{i,t}$ is the corresponding pseudo ground truth position. The acceleration error [42, 53] is widely used for evaluating the smoothness of a motion trajectory and is defined as

$$L_{acc} = \sum_{t=0}^{T-1} \sum_{i=0}^{2} |Y''_{i,t} - G''_{i,t}|, \tag{3}$$

where $Y''_{i,t}$ is the predicted acceleration computed by taking the second derivative of the predicted positions, and $G''_{i,t}$ is the corresponding pseudo ground truth acceleration. Our final loss function is the weighted summation of these two loss terms,

$$L = L_{pos} + \alpha \cdot L_{acc}, \tag{4}$$

where α is the parameter controlling the emphasis on prediction accuracy and prediction smoothness. The larger α is, the smoother the final prediction, but the prediction accuracy would be lower, and vice versa.

4 EXPERIMENTS

In this section, we evaluate the performance of our proposed *N-euro* predictor through experiments. We first describe the experiment setups for data collection and the ground truth labeling. We then introduce several human motion prediction and smoothing baselines for comparison. After that, we show the performance in terms of reducing jitter, lag, as well as computational overhead.

4.1 Data Collection

Apparatus. For our experiments, we use a pair of cameras to track the 3D locations of the user's head in the display's coordinate system. Specifically, we track the center of two eyes of the user. The whole tracking comprises landmark detection, stereo matching for refining the correspondence, and triangulation. More details on the implementation of the tracking system will be described in Section 5 for "Digital Window".

Participants and Procedure. We recruited 11 participants (4 female, 7 male) with a wide coverage of the age range (min = 20, max = 57, mean = 31.2). The participants were asked to move freely in a 3m × 4m rectangle area labeled on the floor while keeping their faces toward the screen. We also encourage the participants to try different moving speeds and acceleration at their own paces (*e.g.*, running, fast/slow walking, sudden stop). We instructed them to do lower speed movement in the first half of the experiment and higher speed movement in the second half of the experiment, so we could easily divide the data into two-speed ranges for further analysis. Each participant performed those motions for 5 minutes while the tracking system is running at 160Hz. In total, we got around 528k (11 × 5 × 60 × 160) data frames.

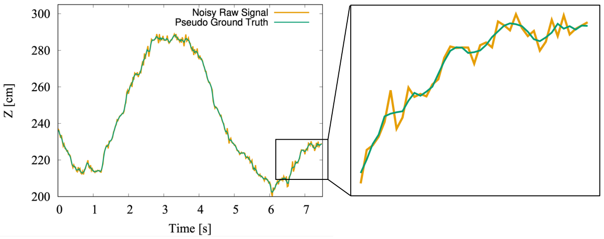


Fig. 3. Double-sided filtering for pseudo ground truth

Data and Labeling. In the experiments, we choose an observation window length of 300 ms after a hyperparameter search, with a prediction length of 50 ms, and a 1-frame moving step. It's important to note that the observation and prediction lengths may vary for different applications. The optimal observation length can be determined by conducting a hyperparameter search after deciding the prediction length based on the specific application. As an empirical suggestion, it is recommended to explore the observation length range of three to eight times the prediction length. To assess performance at a lower tracking rate, we downsampled the 160 Hz signal to 40 Hz. As described in Section 3, we used a double-sided moving average with a window size of 3 to create the pseudo ground truth for the 160 Hz signal. We choose 3 to balance smoothing and accuracy. To create the pseudo ground truth for the 40 Hz data, we first applied a double-sided filter with a length of 5 to the 160 Hz data, then downsampled the data to 40 Hz, and finally applied a double-sided filter with a length of 3 to the 40 Hz data. This approach further enhances the quality of the pseudo-ground truth. Fig. 3 demonstrates the smoothing effect of the double-sided moving average filter. The smoothed trajectory aligns well with the raw signal, indicating low bias, and there is no delay as the filter uses information from both past and future data.

4.2 Baselines and Implementation

We compare our *N-euro* predictor with the following four baselines on two metrics.

Naive Filter. The simplest approach to prediction is to use the current data points as the future prediction.

Double-Exponential Filter. LaViola [47] proposed to use the double exponential filter as a predictive algorithm for AR/VR applications, proving that it provides as accurate predictions as Kalman and extended Kalman filter-based predictors but with much less computation overhead. We implemented a double-exponential predictor as instructed in [47].

One Euro Filter. One euro filter [14] is a first-order low-pass filter with an adaptive cutoff frequency for smoothing the noisy input signals in interactive systems. One euro filter is deployed in many real-time tracking frameworks (*e.g.*, Google MediaPipe face tracking [43]) due to its simplicity and effectiveness. We used the implementation from Nicolas Roussel on the one euro filter website [67].

Long Short-Term Memory (LSTM). LSTM is designed to be able to learn the important parts from the historical sequence and forget the less important ones. LSTM [37] as a powerful recurrent neural network approach, has been leveraged to solve similar problems on sequential data (*e.g.*, head prediction [66], touch screen inputs prediction [36], machine reading [16], anomaly detection [51, 68]). It serves as a reasonable NN-based baseline for our work. We set the number of layers and hidden states to 5 and 256, respectively, resulting in a model size of 1.45 MB with 346 k trainable parameters (>200 times larger than our proposed *N-euro* filter).

Implementation. We used PyTorch [62] to implement all deep learning models (LSTM and ours) and trained the models using a Macbook Pro 2021. The batch size is set to 32. For all models, we used an Adam optimizer with a dynamic learning rate schedule which started from 0.0005 and repeatedly reduced to half once the validation accuracy did not improve after 5 epochs. We set the epoch number to 100, with an early stop number of 30. The final performance of the NN models is computed on the testing set. The whole training time is around 15 minutes.

Evaluation Metrics. We use the Mean Absolute Error (MAE) and the Mean Absolute Acceleration Error (MAAE [42, 53]) between the prediction and the pseudo ground truth as the evaluation metrics for the prediction accuracy and smoothness, respectively. Parameter tuning is essential for traditional filters' prediction/smoothing performance. We perform a brute-force search for the hyperparameters of the double-exponential filter (α and β) and the one euro filter ($f_{c_{min}}$ and β) to find the best parameter settings for the testing set. The best parameter is defined as those resulting in minimal MAE+MAAE for the double exponential filter and minimal MAAE for the one euro filter (minimizing MAE only would result in a worse double exponential filter). It is worth noting that since one euro filter is not designed for prediction, it will for sure give a larger MAE when we compare its outputs to the near-future data points. But it represents the state-of-art smoothing method, so we mainly compare the MAAE metric between it and the proposed *N-euro* predictor.

4.3 Results

We assess the performance of *N-euro* predictor through a leave-one-user-out evaluation protocol to match the close-to-practice usage of the model (*i.e.*, *N-euro* does not need to be re-trained on each new user). So each time, we train the *N-euro* predictor on 10 users' data and test on the left-out user's data. Then we report the average results across the 11 data divisions. As we apply a sliding window approach, for each test set, the prediction results contain the whole trajectory except the first 300ms.

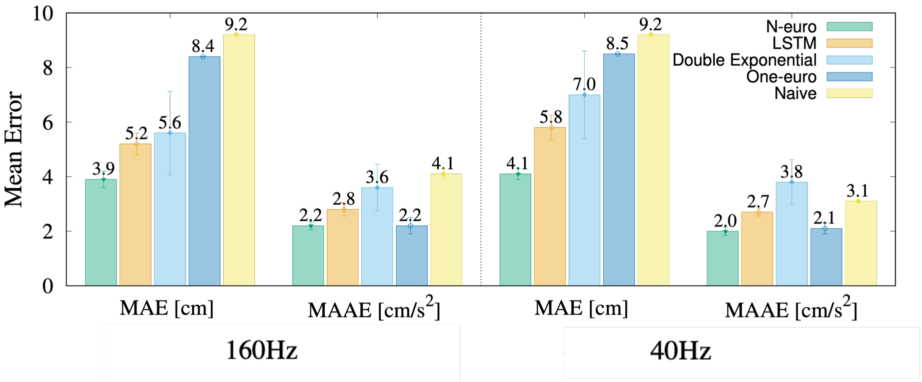
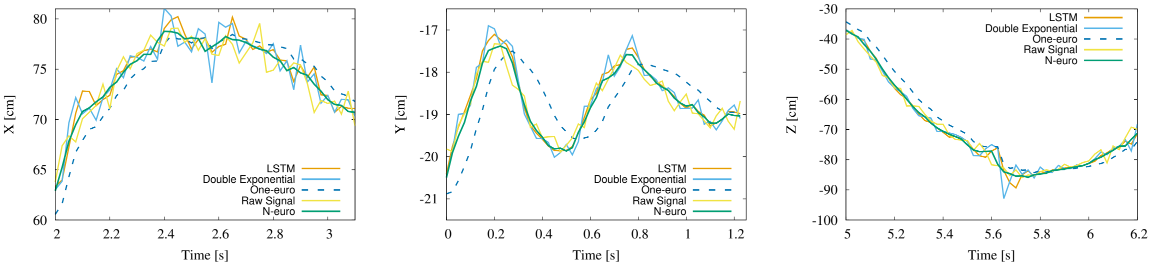


Fig. 4. Overall performance of different methods across users and different speeds. Under both sensing frame rates of 160Hz and 40 Hz, *N-euro* predictor achieved the least MAE (best accuracy) and the least MAAE (best smoothness on par with the fine-tuned one euro filter). 160Hz could provide more fine-grained information on motion dynamics (*e.g.*, speed, acceleration), thus leading to slightly better accuracy in prediction.



(a) Predicted Trajectory (X-axis)

(b) Predicted Trajectory (Y-axis)

(c) Predicted Trajectory (Z-axis)

Fig. 5. *N-euro* gives the best prediction among all (best viewed when zoomed in) on the predicted trajectories on X, Y, and Z axis. X and Z axes are the directions that participants were moving along freely. Y is the vertical axis, where the up-and-down bobbing of human walking happens. Here we can observe that the one-euro filter has an obvious lag (it is actually the current location after smoothing) as it cannot predict future locations, while other filters are showing the predicted future location.

Overall Performance. The overall performance of different prediction methods was evaluated in Fig. 4. The *N-euro* achieved better accuracy and smoothness compared to the baseline LSTM network. Compared to the double exponential filter, the *N-euro* predictor reduced prediction error by 36% and improved smoothness by

42%. The prediction error of the naive filter and the one euro filter was calculated using the current frame as the prediction. Fig. 4 shows that the *N-euro* predictor had similar MAAE (mean absolute average error) to the fine-tuned one euro filter in terms of signal smoothness (and of course much lower MAE in terms of lag). Fig. 5 compares a portion of the predicted trajectories from the *N-euro* predictor and all baselines on all three axes. The raw signal was shifted ahead by 300 ms to represent the target prediction. The *N-euro* provided the best smoothness among all methods and was closest to the raw signal, especially at turning points, validating that the proposed *N-euro* predictor effectively reduces both jitter and lag simultaneously.

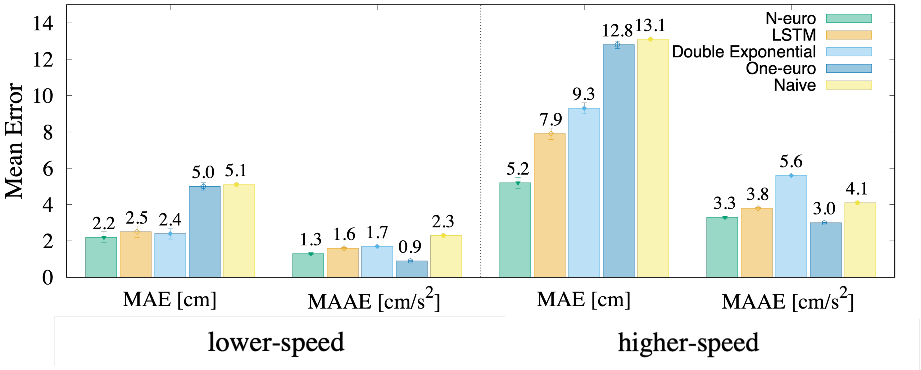


Fig. 6. Impact of movement speed. When predicting lower-speed movements, the baseline filters achieved almost comparable results to *N-euro* predictor, but as the movement speed increased, *N-euro*significantly outperformed the baseline filters. Note that the hyperparameters of the baseline filters were tuned separately for lower-speed and higher-speed data, whereas the neural network (*N-euro*) used the same parameters for both types of data.

Effect of Movement Speed. As previously stated, we collected data from movements at varying speeds, including low-speed movements (*e.g.*, walking) and higher-speed movements (*e.g.*, running, jumping, and squatting), with average speeds of 0.9 m/s and 3.5 m/s, respectively. The overall speed ranges from 0.5 m/s to 6 m/s. As traditional filters need different hyperparameters for different speeds, we performed a brute-force search for both lower and higher-speed movements. The results showed that different hyperparameters were indeed needed for each speed scenario to achieve the best MAE+MAAE on the testing sets. In contrast, the *N-euro* and LSTM filters were trained on the training data without any further fine-tuning on the testing sets. Figure 6 illustrates the impact of movement speed. At lower speeds, the traditional filters with optimal hyperparameters could perform similarly to the *N-euro* and the LSTM filter. However, as speed increased, even with the best hyperparameters, traditional filters were unable to maintain their performance, but the *N-euro* outperformed all baselines in terms of prediction accuracy.

To further illustrate the prediction ability, Fig. 7 shows the prediction trajectories of both a traditional predictive filter (double exponential) and the *N-euro* . Both filters demonstrated larger prediction errors at direction-changing points (*e.g.*, red dashed circles), which are due to the time required for filters to change the motion dynamics (*e.g.*, speed, acceleration) accordingly from historical data. At lower speeds, the change in motion dynamics was small. The double exponential filter could handle this scenario with a similar performance as *N-euro*. However, at high speeds, the better adaptation ability of the *N-euro* resulted in a much lower prediction error.

Impact of Sensing Frequency. We assessed the impact of sensing frequency by training the *N-euro* with data collected at two different frequencies: 160Hz and 40Hz (downsampled from 160Hz). The observation length was fixed at 300ms and the prediction length at 50ms, which correspond to 40/8 frames for the 160Hz sensing

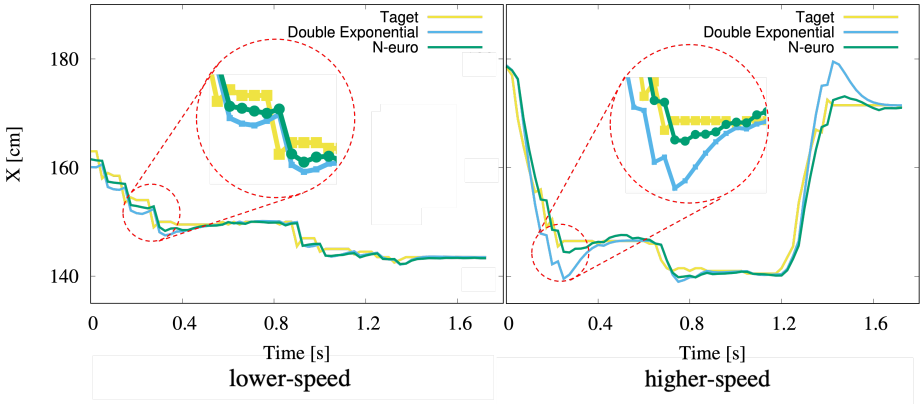
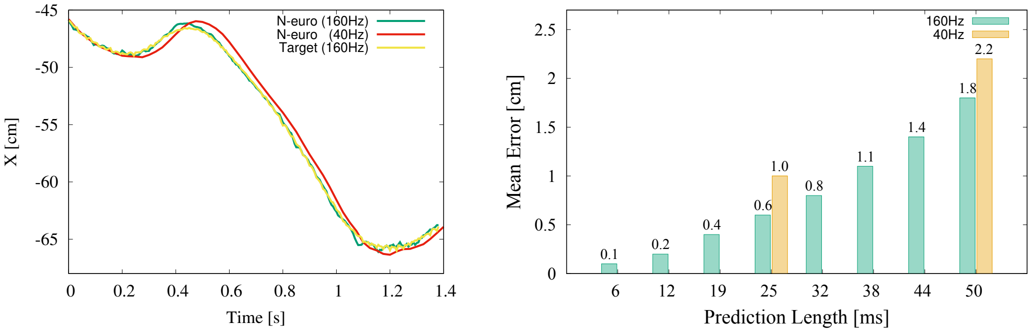


Fig. 7. Comparison between traditional predictive filter (double exponential) and *N-euro*. At the direction turning points (*e.g.*, red dashed circle), the double exponential filter has more "over-anticipate" than *N-euro*. At lower speeds, the difference is not significant. While with higher-speed movements, *N-euro* achieved much less prediction error at those turning points than the fine-tuned double exponential filter.



(a) Impact of sensing frequency

(b) Impact of prediction length

Fig. 8. (a) The predicted trajectories with a prediction length of 50ms using different sensing frequencies are shown. The results indicate that using a higher sensing frequency of 160Hz leads to lower prediction error (closer to the target) compared to a lower frequency of 40Hz, particularly at turning points. The prediction with 160Hz appears to have more jitter due to its capturing of three times more points (thus more fluctuates between data points; higher initial MAAE). (b) The longer the prediction length is, the higher the prediction error will be.

frequency and 10/2 frames for the 40Hz sensing frequency. As depicted in Fig. 8(a), the prediction with a 160Hz signal demonstrates a lower error (closer to the target). The prediction with 160Hz appears more jitter (higher MAAE) due to its capturing of three times more points (thus more fluctuates between data points; higher initial MAAE). From Fig. 4, we observe that prediction with 160 Hz reduces more jitter (a reduced MAAE of 1.9 cm/s^2 from the naive filter) compared to the prediction with 40Hz (a reduced MAAE of 1.1 cm/s^2 from the naive filter).

In Fig. 8(a), the predicted trajectories further highlight the phenomenon in the speed experiments where deviation from the actual trajectory is most significant at direction-changing/turning points. The higher sensing rate results in more data points to learn the movement dynamics (this reveals the same underlying rationale with lower movement speeds), making it easier for the neural network to learn the changes in speed and acceleration. This, in turn, results in a shorter time for the prediction to catch up with direction changes, leading to the observed less “over-anticipate” in the prediction compared with 40Hz inputs. This analysis suggests intuition on why the neural network could predict the motion: it is still fitting a curve from the historical data but in a more complicated and powerful way, allowing for quicker adaptation to direction changes than the traditional methods. This also implies that the “over-anticipate” issue is inherent (adaptation takes time anyway) and will be discussed further in Section 6.

Impact of Prediction Length. Fig. 8(b) illustrated that the larger the prediction length, the larger the prediction error will be. This is because human motion is essentially arbitrary, and the longer it is into the future, the less likely it will be related to the past. This plot also validated that with the same prediction duration, prediction with 160 Hz as inputs is more accurate than prediction with 40Hz signals.

Summary. The proximity of adjacent sensing data points reflects the human movement speed and sensing frequency. Based on the above experiments, when the distance between adjacent sensing data points is short, all filters tend to perform better, and the difference between *N-euro* and traditional filter is minor. However, as the distance grows (indicating a higher movement speed or lower sensing frequency), the *N-euro* predictor significantly outperforms traditional filters under the condition that they have been fine-tuned on the testing set. In practice, fine-tuning for finding the best parameters cannot be achieved easily in real-time and traditional filters are typically set to a middle ground between low and high speed, which leads to subpar performance at either end. Given that most human-display interactive applications involve a wide range of movement speeds and relatively low tracking frequencies, the advantages of the *N-euro* become more pronounced.

4.3.1 Generalization to Other Motions without re-training.

We evaluated the generalization ability of the *N-euro* filter to other types of human motion without retraining the model on new data. Specifically, we tested its performance on human hand motion in free space and cursor movements. *N-euro* filter works reasonably well in both scenarios.

Human Hand Motion. Hand motion is another common way for human-display interactions [40, 50, 65, 71, 78]. To assess the performance of the *N-euro* predictor on human hand motion, we collected data from one participant who wore an AR headset to capture hand motion videos at 40 FPS while performing daily activities (such as waving, drinking, and boxing). We processed the videos using Mediapipe [30] to obtain the trajectories of the center of the palm. We used the model trained on 40 Hz head movement data to predict 50 ms into the future of hand movements. The average prediction error for waving and drinking was 1.1 cm, and for boxing was 27.6 cm.

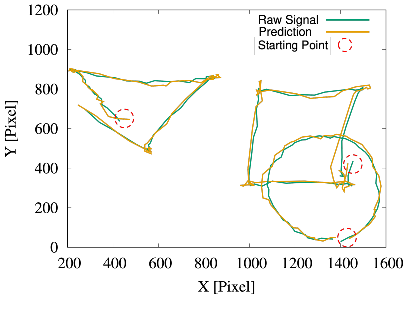


Fig. 9. Predicted cursor trajectories of different shapes.

Table 1. Running time of different methods

Method	Running Time [ms]
Double Exponential Filter	0.05
One Euro Filter	0.01
LSTM	2
N-euro	0.95

Cursor Movement. We collected cursor trajectory data using a Python application with the PyAutoGUI library [75]. One participant drew different shapes (*e.g.*, circle, square, triangle) using the cursor, and the data was collected at 40 FPS. After collecting the cursor trajectories, we added Gaussian noise with a 0 mean and 5-pixel variance. We then used the *N-euro* predictor, which was trained on the 40 Hz head movement data, to predict the cursor movements 50ms into the future. As shown in Fig. 9, the predicted trajectories were slightly off at the edges of the triangle and square, but overall the accuracy was not bad. The average prediction error was 12.4 pixels, which was computed by comparing to the initial cursor trajectory before adding the noise (actual ground truth).

The average moving speed in the testing dataset for cursor movements was around 520 pixel/s, which corresponds to a moving distance of 26 pixels in the 50 ms prediction length. The resulting prediction error percentage was 47%, a value similar to that obtained for head movement prediction using the *N-euro* predictor (42%) and the double exponential filter (46%). However, boxing, being a fast-moving and direction-changing motion with an average speed of 9.2 m/s, has a much higher average speed than the highest moving speed (6 m/s) in the *N-euro* predictor training set. Consequently, the prediction error percentage for boxing was much higher at 60%. This suggests that *N-euro* predictor's performance is affected by the similarity of motion dynamics such as speed and acceleration between the training and testing scenarios. In cases where there is a significant difference in dynamics, it may be necessary to collect new training data to achieve optimal results.

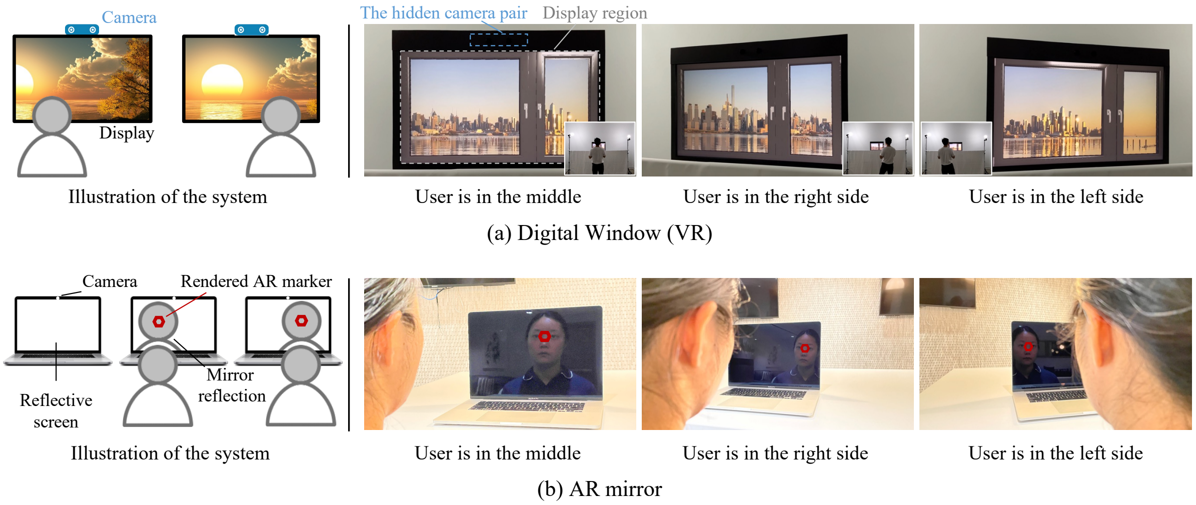
4.3.2 Computation Overhead. The whole model size of *N-euro* is 5 kB with 1.2 k trainable parameters. To demonstrate the computation/inference speed for each method, we took 100 random test samples from each method and computed the overall average running times on a MacBook Pro 2021. Table **??** shows the computation overhead for each method.We observe that although the *N-euro* predictor runs slower than traditional filters, its running time remains under 1 ms. This level of efficiency is well-suited for real-time human-display interactions, ensuring seamless and responsive user experiences. Further insights on computation overhead are discussed in Section 6.

5 USER STUDY

After validating *N-euro* from the algorithmic perspective, we then conducted a user study to showcase the capabilities of *N-euro* for improving the actual user experience in different human-display interactions. We implemented two real-time interactive systems. As shown in Fig. 10(a), the first system – Digital Window, is an Fish Tank Virtual Reality (FTVR) system that uses a desktop display to mimic an actual window by head-tracked rendering [27, 34, 88]. The second system illustrated by Fig. 10(b) is an AR mirror that always renders an add-on marker to the center of the user's eyes while the user is moving. Each system has four different versions, which correspond to four different ways of motion prediction and smoothing. Each user would play with all four versions of the two systems. During the user study, we measured participants' perceived jitter, lag and collected qualitative feedback on the overall user experience. This study was approved by our Institutional Review Board.

5.1 Apparatus and Participants

In both devices, we track the 3D locations of the center of two eyes of the user and define the origin of the display's coordinate system as the center of the display and the three axes as shown in Fig. 11(a). Note that we use camera(s) to track the user and get the result in the camera's coordinate system, and thus we need to convert the results in the display's coordinate system. The calibration of the two coordinate systems can be done with a mirror [28]. Since the movement range of the user in Digital Window is much larger than that in the AR Mirror, it also requires a higher tracking accuracy. Therefore, we leveraged a pair of cameras in the Digital Window system and a single web camera in the AR mirror system. Note that both devices do not need the gaze direction of the user. The following introduces our implementation details on the hardware, tracking algorithm, and rendering methods for each system.



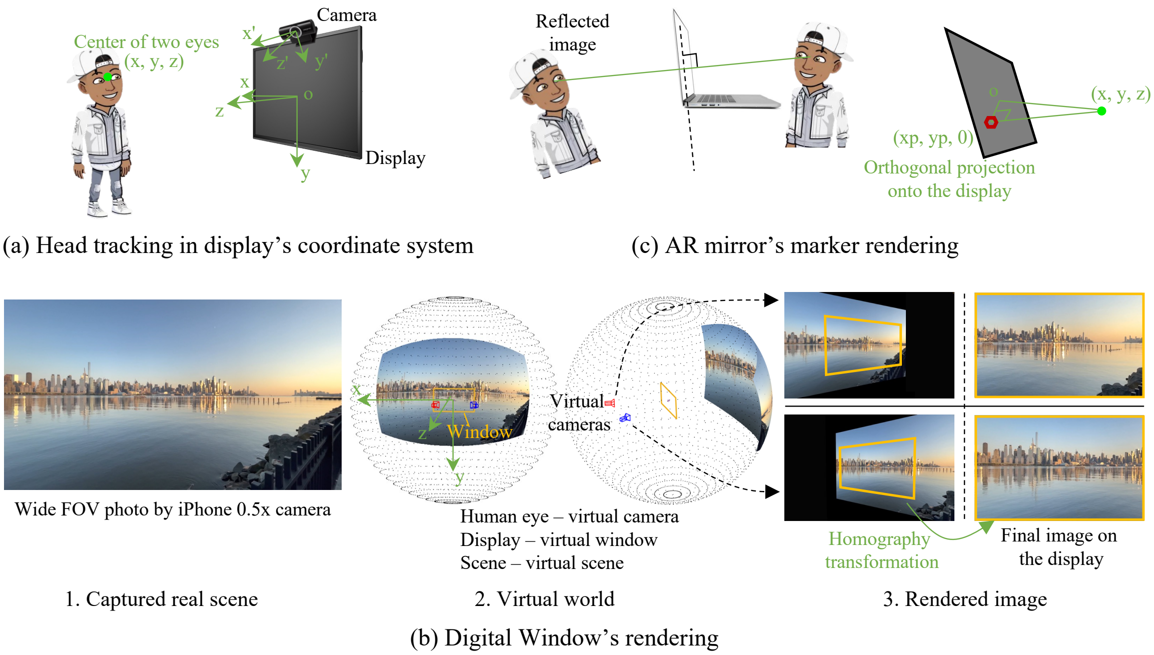
(a) Digital Window (VR)

(b) AR mirror

Fig. 10. The two systems used to evaluate our proposed predictor in the user study are: (a) Digital Window - a display that mimics a real window by showing scenes based on the user's position. The left image shows the system, and the right shows our implementation with the big images being what the user sees (captured by a camera attached to the face) and the insets showing the user's location. (b) AR Mirror - an AR marker is attached to the mirror-reflected face while the user moves. The left image shows the system using a reflective screen as the mirror. The right image shows our implementation (captured by a camera attached to the side of the face; for illustration, the marker is drawn in the camera's perspective, not the user's eye perspective used in the user study).

Digital Window (VR). The tracking system uses a pair of Basler cameras [11] with wide angle lens [76], and a micro-controller Teensy 3.2 [2] for synchronization. The display is a 27-inch Acer Nitro XV273K [1]. All the computation is done on a Dell desktop with NVIDIA GeForce RTX 3090 GPU [19]. The tracking is composed of landmark detection, stereo matching for refining the correspondence, and triangulation and is implemented in Python with the OpenCV libraries [13]. The tracking result is sent to an HTML webpage application through WebSocket. The webpage application renders scenes according to the current location of the user and is implemented in Three.js (JavaScript and WebGL). The tracking is running at 40 FPS and the display's refresh rate is 120Hz. We next talk about the rendering details for mimicking a real window.

Since we are using a fixed-size window of 27 inches, we need to render the scenery such that it will be the same as what a user will see through a 27-inch real window. To do this, we first capture a wide field-of-view (FOV) photo of the iPhone's 0.5x camera (as shown in Fig. 11 (b.1)). Then, in the rendering system, we place a virtual window around the origin (Fig. 11 (b.2), yellow rectangle) which has the same size as the display. We convert the photo from a perspective projection to a spherical representation and attach it to a big sphere as the texture. The sphere's center is aligned with the virtual window's center, and the texture is directly in front of the window (see Fig. 11 (b.2)). We place a virtual camera at the same location as the user. In order to have more pixels in the rendered window's region, we point the camera to the center of the window. Fig. 11 (b.3) shows the rendered images with the yellow window; finally, we do a homography transformation to convert the window region to be rectangular and show it on the display ((b.3)). We validated these steps by making sure that when we put the display at the same location where the photo was captured, the rendered scene coincides with the real scene; that is, the display looks transparent. To make the digital window more realistic, we added a 3d window



(a) Head tracking in display's coordinate system

(c) AR mirror's marker rendering

(b) Digital Window's rendering

Fig. 11. The implementation details of the two systems. (a) The 3d location of the center of two eyes is tracked in the display's coordinate system. Gaze direction is not needed. (b) Digital Window's rendering system places the virtual camera, virtual window, and virtual scene following the real locations of the user, display, and scene such that the rendered scene is realistic and mimics what a user will see through a real window. (c) AR mirror renders the marker at the projection point of the face center onto the display such that the user will see it at their reflected face's center.

frame and virtual indoor lighting similar to real lighting, *e.g.*, the brown window shown in Fig. 10 (a); we also hid the cameras with black cardboard as shown in Fig. 10 (a) right blue dotted line box; finally, we added sound which is compatible with the scene,*e.g.*, adding the sound of waves to the scene of Fig. 11 (b).

AR Mirror. The system is implemented with an M1 Pro MacBook Pro [4]. The MacBook's screen is reflective, and the user can clearly see their reflection when the screen is dark (see Fig. 10 (b) right). We render an AR marker on the reflected face's center from the user's eye's perspective. This is achieved by tracking the 3d locations of the center of two eyes in the display's coordinate system, then projecting it to the display screen, and then drawing the marker on the projection point (see Fig. 11 (c)). The tracking is done by MacBook's embedded web camera and implemented by Python code and OpenCV. The 3d location estimation is by a Facemesh algorithm and known pupillary distance. For the Facemesh algorithm, the input is a single face image and the camera's intrinsic parameters (including focal length and center points), the output is 3d mesh of the face in the camera's coordinate system with an unknown scale, and it can be implemented by 3D Morphable Face Models method [5, 22, 29]. The unknown scale can be solved by the known pupillary distance. The rendering is simply adding a circle to the tracked locations.

Participants. We recruited 34 participants (14 female, 20 male; 13 in the age group of 18 – 24, 19 in the age group of 25 – 34, and 2 in the age group of 34 – 45). No participant reported prior injuries to their eyes. Each participant received a compensation of $25 for their time after completing the study.

5.2 Design and Procedure

Baseline and Comparison Filters. We have four versions of the system [2]: 1) Baseline: No filter. The motion-to-photon latency without any filter is around 60 ms for the Digital Window and around 30 ms for the AR mirror. 2) One euro filter: we tuned the one-euro filter for each system to validate smoothing performance. 3) Double exponential filter: we fine-tuned the double exponential predictive filter for comparison. 4) *N-euro*. Both the double exponential filter and our *N-euro* were configured to predict 50 ms and 25 ms into the future for the Digital Window and AR mirror, respectively, reducing perceived lags to an unperceivable level (<10 ms).

Experiment Design. The whole study contains two sections for the corresponding two systems. In each section, we randomly choose two versions of the system as a trial out of the four setups. So each section consists of 6 trials. In total 6 trails × 2 sections = 12 conditions per participant. The experiment design is within-subject.

Side-effects Metrics. As shown in [57], Root Mean Square Error (RMSE) sometimes cannot capture the side effects caused by the prediction. To compare the perceived user experience among the predictors, we also used 6 most common side effects (wrong distance is hard to observe in our setting) from [57]:

- lateness: The prediction was perceived as late or slow to react to the actual movement.
- jitter: The prediction was perceived as trembling around.
- over-anticipate: The prediction was perceived as too far ahead in time or to overreact to the user input.
- wrong orientation: The prediction was not going in the same direction as the user's movement.
- spring effect: The prediction appeared to "yo-yo" around, possibly in several motions.
- jumps: The prediction appeared to jump away from the finger at times.

Among them, lateness and jitter correspond to the two main effects our *N-euro* design targets at: perceived delay and jitter.

Procedure. In the beginning, participants signed a consent form and received an overview of the user study. Table height was calibrated to place the desktop/display at the center of each participant's head. An instructor then demonstrated how to interact with the systems with different movement directions, distances, speeds, and accelerations and reviewed side effect descriptions. Participants experience two randomly selected versions of the system consecutively in one trial. Since there are 4 versions of the system, this results in a total of 6 trials (all combinations of two versions out of 4 versions) for each participant. Each trial takes 2 minutes (one minute for each version). Immediately after each trial, participants were asked to rate jitter, lag, and overall user experience. The ratings were on a scale of 3 (no/little jitter/lag, best user experience), 2 (noticeable jitter/lag, fine user experience), and 1 (obvious jitter/lag, barely usable/unbearable user experience). Participants also verbally described any side-effects. An interview was conducted at the end to gather participants' overall experiences and suggestions. The study took around 30 minutes per section, with breaks encouraged between sections and trails. Participants were not informed of the filters used, even for the baseline group.

5.3 Qualification Round

The qualification round of our study was designed to evaluate the visual sensitivity of our participants. A simple experiment was conducted after the main user study to ensure that they had the ability to distinguish between different versions of the system. In the latency sensitivity test, the digital window system was set to three different lags: 60 ms (baseline), 120 ms (an additional 60 ms), and 180 ms (an additional 120 ms). For the jitter sensitivity test, Gaussian noise was added to the tracked eye position with zero mean and standard deviation of 0, 0.01, and

[2] We unfortunately could not integrate the TurboTouch predictor [56] in our study due to the complex fine-tuning process, the final user experience of the optimal TurboTouch predictor and the N-euro predictor remains to be compared.

0.04. Participants were asked to rank the lag and jitter in decreasing order. If they were unable to rank them correctly, their evaluations were excluded from the data analysis.

5.4 Results

All 34 participants passed the jitter qualification test. 4 participants failed the latency sensitivity qualification test as they reported that the three versions of the digital window system with different preset delays are similar to them. As a result, their answers were excluded from the following analysis.

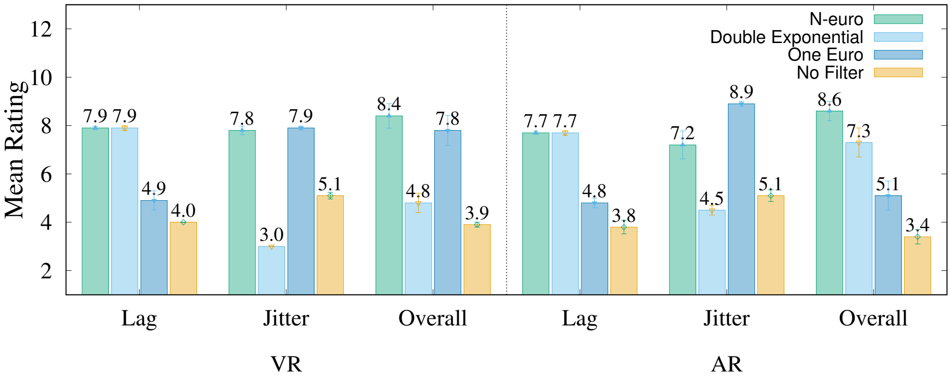


Fig. 12. The *N-euro* predictor was rated as delivering the best overall user experience in both VR and AR settings, performing as well as the double exponential filter in prediction and equally well as the one euro filter in smoothing. In terms of lag performance, the *N-euro* predictor was found to be equal to or better than the double exponential filter and better than the one-euro filter and baseline. Meanwhile, the one euro filter was rated as equal to or better than the *N-euro* predictor in terms of jitter performance, with the baseline and double exponential filter rated as worse.

Perceived Lag and Jitter. Since each participant experienced all combinations of the two randomly selected versions of the system consecutively out of the four versions, each filter was compared to all the other three filters. Therefore, the added score range for each filter is $3 - 9$ for the three metrics: lag, jitter, and overall.

For the Digital Window application, three 4-level one-way repeated measures ANOVAs reveal a significant main effect of different filters on perceived jitter ($F(3, 27) = 251.79, p < 0.001$), perceived lag ($F(3, 27) = 317.22, p < 0.001$) and overall experience ($F(3, 27) = 269.26, p < 0.001$), indicating that the user experience varied across the different versions of the system. Subsequent post-hoc Tukey tests demonstrated that the system with the *N-euro* predictor outperformed the other three versions across all three metrics ($p < 0.001$) , except that for the perceived lag, there is no significant difference ($t(28) = 1.01, p = 0.32$) between the *N-euro* predictor and double exponential predictor; and for the perceived jitter, there is no significant difference ($t(28) = -0.42, p = 0.67$) between N-euro predictor and the one euro filter.

For the AR mirror application, the same three 4-level one-way repeated measures ANOVAs were computed. similar results were obtained. The different filters had significant effects on perceived jitter ($F(3, 27) = 172.22, p < 0.001$), perceived lag ($F(3, 27) = 215.82, p < 0.001$), and overall experience ($F(3, 27) = 252.33, p < 0.001$). The post-hoc Tukey tests indicated that the *N-euro* predictor outperformed the other versions in all metrics (p < 0.001), except for perceived lag ($t(28) = 0.23, p = 0.81$) where there was no significant difference compared to the double exponential predictor; and for perceived jitter, the one euro filter outperformed the *N-euro* predictor ($t(28) = -10.34, p < 0.001$).

These statistical analyses demonstrate that the *N-euro* predictor was able to effectively reduce lag to the same extent as the double exponential filter while at the same time smoothing out jitter as well as the one euro filter, receiving the highest ratings for the overall user experience in both VR and AR settings.

We also calculated the average ratings from all 30 participants and presented the mean scores of each filter for the three metrics: lag, jitter, and overall. As shown in Fig. 12, in the Digital Window/VR setting, the overall user experience ranking was aligned with the jitter performance: one euro filter >= N-euro > baseline > double exponential filter. In the AR mirror setting, however, the overall user experience was more closely aligned with the lag performance: *N-euro* >= double exponential filter > one euro filter >= baseline. This suggests that participants in the FTVR setting, who were farther from the screen, paid more attention to jitter. In contrast, in the AR mirror setting, where the mirrored image was treated as a reference with no delay, participants were more sensitive to lag. Despite the higher levels of jitter in the double exponential filter, more participants preferred it over the one-euro filter in the AR setting. Two participants preferred the one-euro filter over the *N-euro* predictor even if the *N-euro* had less lag and the same jitter level. This was due to the imperfect prediction, especially at direction-changing moments, which we will discuss further in Section 6.

Side-effects. The side effects of lateness and jitter have been discussed through the perceived lag and jitter rankings. In addition to these, the most commonly reported side effect by participants was "wrong orientation" and "over-anticipation". 21 and 12 participants reported that the system using the double exponential filter had the side effects of "wrong orientation" and "over-anticipation", respectively, while 4 and 2 participants reported that *N-euro* had the side effects of "wrong orientation" and "over-anticipation", respectively. Participants noted that the "wrong direction" and "over-anticipation" side effects typically occurred at turning points where they suddenly stopped and changed their moving direction. This result is consistent with our analysis in Section 4 (Fig. 7) which shows that both *N-euro* and the double exponential filter have higher prediction errors at turning points, but *N-euro* predictor outperforms the double exponential filter significantly with much less overshooting. No participants reported "spring effect" or "jumps".

Positive Comments. Many participants provided positive comments about the system. P2 mentioned after trying the AR mirror that *"It's like magic. I can see a very obvious lag between my motion and the rendered circle on the first version. But the last version could accurately follow my motion."* 25 out of 30 participants said they like the Digital Window system and would be willing to have one at home/office. P11 was really impressed: *"Really fun experience! I can see the potential for this technology. It's fun that I feel tired when there is lag in the system and feel 'lightweight' and relaxed when played with one of the versions"*. P25 was happy after the user study: *"I was worried before the experiments that maybe I cannot feel the difference if the difference/improvement is subtle. Cause I do not play video games at all and am not sensitive to tiny motions. After I tried these, oh, I can!"* Some participants also mentioned that they felt less motion sickness when the lag and jitter were smaller: *"as someone who is pretty sensitive to motion sickness, I obviously felt better when I was using the third system (with the proposed predictor)."* The overall user experience rating and the subjective feedback show good practical usability of the *N-euro* predictor and the potential for the proposed Digital Window system.

User Suggestions. Some participants also gave suggestions for system improvements. 10 out of 30 participants mentioned that to mimic a real window experience better, the display should be embedded into a wall or use a projector instead of a display. Another common suggestion is the rendered scenery on the Digital Window should be 3D and the lighting reflected from the objects in the scenery should change accordingly. Over half of the participant thinks the add-on for the AR mirror should be more fascinating (*e.g.*, P9: *"it should be similar to the AR effect of the Snapchat/Instagram lens, so it can be more practical."*). Two people (P2 and P15) mentioned that they can feel the lag when they were quickly changing directions.

6 DISCUSSION

We discuss some of our limitations in this work and possible directions for improvement.

Ground truth. Although pseudo-ground truth labeling makes it easier to apply our *N-euro* to various interactive systems, we acknowledge that one limitation of our work is the lack of actual continuous ground truth in some of the experiments. In these cases, we passed the sensed signal to a double-sided moving average filter to obtain the labels for supervising the training. This means that the real-time prediction performance of our *N-euro* was trained to approximate the performance of an offline double-sided moving average filter, which can see into the future. We conducted simulations on head motion data and the cursor experiment, adding noise to the ground truth to simulate noisy measurements and calculating pseudo-ground truth from the noisy simulated measurements. We trained *N-euro* using both types of ground truth and found similar errors in both settings, indicating the validity of using the double-sided moving average filter result as the pseudo-ground truth. Additionally, our user study results demonstrated that systems with our *N-euro* indeed provided the best actual user experience compared to other filters. Although we have shown that our *N-euro* can achieve good performance using pseudo-ground truth, it is worth noting that actual continuous ground truth remains the best way to supervise training if it is available.

Prediction Errors at Turning Points. Given the non-linear and nondeterministic nature of human motion, uncertainty and errors are inherent in predictions. Although our experiments demonstrated that *N-euro* adapts to motion dynamics faster than other baselines, we observed that the error at turning points is consistently higher than in other places. However, we also found that a higher sensing rate led to lower prediction errors, particularly at turning points. To address this issue, a possible approach is to train a smart upsampling network and then apply *N-euro* to the generated higher-rate data. This two-stage method may outperform a one-stage method that directly uses low-rate data because the upsampling training can be supervised. This approach is in line with recent work [20] and represents an interesting avenue for future research.

Multi-points Prediction. In Section 3, our approach considers human motion as a collection of moving points, where our method makes predictions by processing the time series of individual points independently. While this approach works well when dealing with rigid bodies, where the relative positions between points remain constant, it may face challenges in scenarios involving non-rigid bodies, such as the human body. In such cases, predicting multiple points independently, like the hand, arm, and body, may lead to results that do not adhere to specific shape or kinematics constraints. To overcome this limitation, an avenue for future research involves exploring ways to incorporate these shape and kinematics constraints into the prediction process. By doing so, we can significantly enhance our ability to predict multiple points simultaneously on non-rigid bodies, thus improving the overall accuracy and coherence of the predictions.

Motion-to-Photon Latency Measurement. In our current user study evaluation, we relied on participant-reported perceived lag as an indicator of the actual user experience. However, for a more comprehensive analysis, it would be beneficial to also report the precise motion-to-photon latency of the system after incorporating various types of filters. There was no standard and easy way to measure the motion-to-photon latency in interactive systems. Most ongoing research work focuses on measuring the motion-to-photon latency in Head-Mounted-display [18, 70, 74]. Systematically measuring motion-to-photon latency within our current user study setting proves to be complex. Obtaining the user's precise location at the moment when rendering is complete presents considerable difficulty. To achieve this, it is necessary to capture the user's motion and the rendered scene simultaneously, either through a single sensor or multiple synchronized sensors. One commonly employed approach involves utilizing a high-speed camera capable of capturing both the display and the user. However, a potential issue arises due to the nature of unrestricted user movement, which may obstruct the camera's view during motion, reflecting the real-life usage scenario. Alternative techniques could be explored in the future. For instance, despite the complexity of calibration and synchronization, employing multiple cameras from different angles or using additional sensors like accelerators or photodiodes might help capture the user's motion accurately without obstructions.

Complexity of Neural Network vs. Traditional Filters. Compared to traditional hand-crafted filters that usually have only $1 - 10$ parameters, the neural network (NN) requires significantly more parameters and computation. This can make it challenging to be used on edge devices with low computational power. Additionally, the NN requires a training process and has a longer running time, whereas traditional filters can be easily implemented (though the hyper-parameter tuning could be tricky) and have an instant running time. Fortunately, the computation cost of *N-euro* is acceptable for middle to high-end platforms such as AR glasses, phones, and laptops. The running time is fast, taking less than 1 ms, and the training can be completed within tens of minutes. Our proposed approach is not intended to replace traditional filters entirely, but rather to explore whether a NN with thousands of parameters can learn human motion dynamics and make better predictions than hand-crafted filters with the best-tuned parameters. This paper marks a preliminary step towards achieving this objective. Future work could focus on reducing the network complexity using common operations like distillation, pruning, and quantization. Additionally, a hybrid approach could combine the strengths of traditional filters and the NN, such as using the NN to estimate speed/acceleration from noisy data and integrating them into a double exponential filter or directly employing the NN to estimate parameters of traditional filters.

7 CONCLUSION

This work introduced a lightweight neural-network-based predictor, *N-euro*, for motion trajectory smoothing and prediction in vision-based interactive interactions. The *N-euro* predictor simultaneously reduces the jitter and lag, which are the two main issues influencing the human-display interaction user experience. Compared to other widely-used smoothing and predictive filters, the experiment result shows *N-euro* predictor achieves the least prediction error. The effectiveness of the *N-euro* filter was further validated through a user study (n=34) with two self-designed interactive systems, Digital Window (VR) and AR mirror. Repeated ANOVAs and the post-hoc Tukey tests indicate that participants preferred the system equipped with the proposed predictor over all other baselines. This simple yet effective predictor has the potential to be widely adopted in various vision-based interactive systems.

ACKNOWLEDGMENTS

We sincerely thank the anonymous reviewers for their insightful comments that helped improve the paper.

REFERENCES

[1] Accer. 2022. NITRO XV3. https://www.acer.com/ac/en/US/content/model/UM.HX3AA.P02. Accessed: 2023-02-13.

[2] adafruit. 2022. Teensy 3.2. https://www.adafruit.com/product/2756. Accessed: 2023-02-13.

[3] Axel Antoine, Mathieu Nancel, Ella Ge, Jingjie Zheng, Navid Zolghadr, and Géry Casiez. 2020. Modeling and reducing spatial jitter caused by asynchronous input and output rates. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 869–881.

[4] Apple. 2021. MacBook Pro. https://www.apple.com/shop/buy-mac/macbook-pro/16-inch-space-gray-10-core-cpu-16-core-gpu-512gb. Accessed: 2023-02-13.

[5] Apple. 2022. Face Tracking with ARKit. https://developer.apple.com/videos/play/tech-talks/601/. Accessed: 2022-08-27.

[6] Elena Arabadzhiyska, Okan Tarhan Tursun, Karol Myszkowski, Hans-Peter Seidel, and Piotr Didyk. 2017. Saccade Landing Position Prediction for Gaze-Contingent Rendering. *ACM Trans. Graph.* 36, 4, Article 50 (jul 2017), 12 pages. https://doi.org/10.1145/3072959.3073642

[7] Adebiyi A Ariyo, Adewumi O Adewumi, and Charles K Ayo. 2014. Stock price prediction using the ARIMA model. In *2014 UKSim-AMSS 16th international conference on computer modelling and simulation*. IEEE, 106–112.

[8] Takeshi Asano, Ehud Sharlin, Yoshifumi Kitamura, Kazuki Takashima, and Fumio Kishino. 2005. Predictive Interaction Using the Delphian Desktop. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology* (Seattle, WA, USA) *(UIST '05)*. Association for Computing Machinery, New York, NY, USA, 133–141. https://doi.org/10.1145/1095034.1095058

[9] Ronald Azuma and Gary Bishop. 1994. Improving static and dynamic registration in an optical see-through HMD. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. 197–204.

[10] C Narendra Babu and B Eswara Reddy. 2014. A moving-average filter based hybrid ARIMA–ANN model for forecasting time series data. *Applied Soft Computing* 23 (2014), 27–38.

[11] Basler. 2022. Basler acA1300-200um camera. https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/aca1300-200um/. Accessed: 2023-02-13.

[12] Andrea Bellucci, Alessio Malizia, Paloma Diaz, and Ignacio Aedo. 2010. Human-Display Interaction Technology: Emerging Remote Interfaces for Pervasive Display Environments. *IEEE Pervasive Computing* 9, 2 (2010), 72–76. https://doi.org/10.1109/MPRV.2010.30

[13] G. Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[14] 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* (Austin, Texas, USA) *(CHI '12)*. Association for Computing Machinery, New York, NY, USA, 2527–2530. https://doi.org/10.1145/2207676.2208639

[15] Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza, Max Mergenthaler, and Artur Dubrawski. 2022. N-hits: Neural hierarchical interpolation for time series forecasting. *arXiv preprint arXiv:2201.12886* (2022).

[16] Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733* (2016).

[17] Pei-Yu (Peggy) Chi, Yang Li, and Björn Hartmann. 2016. Enhancing Cross-Device Interaction Scripting with Interactive Illustrations. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 5482–5493. https://doi.org/10.1145/2858036.2858382

[18] Song-Woo Choi, Siyeong Lee, Min-Woo Seo, and Suk-Ju Kang. 2018. Time sequential motion-to-photon latency measurement system for virtual reality head-mounted displays. *Electronics* 7, 9 (2018), 171.

[19] Dell. 2022. Precision 7920 Tower Workstation. https://www.dell.com/en-us/shop/desktop-computers/precision-7920-tower-workstation/spd/precision-7920-workstation. Accessed: 2023-02-13.

[20] Xingbo Dong, Wanyan Xu, Zhihui Miao, Lan Ma, Chao Zhang, Jiewen Yang, Zhe Jin, Andrew Beng Jin Teoh, and Jiajun Shen. 2022. Abandoning the Bayer-Filter To See in the Dark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 17431–17440.

[21] Volkan Ş Ediger and Sertac Akar. 2007. ARIMA forecasting of primary energy demand by fuel in Turkey. *Energy policy* 35, 3 (2007), 1701–1708.

[22] Bernhard Egger, William AP Smith, Ayush Tewari, Stefanie Wuhrer, Michael Zollhoefer, Thabo Beeler, Florian Bernard, Timo Bolkart, Adam Kortylewski, Sami Romdhani, et al. 2020. 3d morphable face models—past, present, and future. *ACM Transactions on Graphics (TOG)* 39, 5 (2020), 1–38.

[23] Steven Feiner, Blair MacIntyre, Marcus Haupt, and Eliot Solomon. 1993. Windows on the World: 2D Windows for 3D Augmented Reality. In *Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology* (Atlanta, Georgia, USA) *(UIST '93)*. Association for Computing Machinery, New York, NY, USA, 145–155. https://doi.org/10.1145/168642.168657

[24] Xianglong Feng, Viswanathan Swaminathan, and Sheng Wei. 2019. Viewport Prediction for Live 360-Degree Mobile Video Streaming Using User-Content Hybrid Motion Tracking. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 2, Article 43 (jun 2019), 22 pages. https://doi.org/10.1145/3328914

[25] Nisal Menuka Gamage, Deepana Ishtaweera, Martin Weigel, and Anusha Withana. 2021. So Predictable! Continuous 3D Hand Trajectory Prediction in Virtual Reality. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 332–343. https://doi.org/10.1145/3472749.3474753

[26] Everette S Gardner Jr. 2006. Exponential smoothing: The state of the art—Part II. *International journal of forecasting* 22, 4 (2006), 637–666.

[27] William W. Gaver, Gerda Smets, and Kees Overbeeke. 1995. A Virtual Window on Media Space. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '95)*. ACM Press/Addison-Wesley Publishing Co., USA, 257–264. https://doi.org/10.1145/223904.223937

[28] Katia Genovese, Yuxi Chi, and Bing Pan. 2019. Stereo-camera calibration for large-scale DIC measurements with active phase targets and planar mirrors. *Optics express* 27, 6 (2019), 9040–9053.

[29] Google. 2022. MediaPipe Face Mesh. https://google.github.io/mediapipe/solutions/face_mesh.html. Accessed: 2023-02-13.

[30] Google. 2022. MediaPipe Hands. https://google.github.io/mediapipe/solutions/hands.html. Accessed: 2023-02-13.

[31] Serhan Gül, Sebastian Bosse, Dimitri Podborski, Thomas Schierl, and Cornelius Hellge. 2020. Kalman Filter-Based Head Motion Prediction for Cloud-Based Mixed Reality. In *Proceedings of the 28th ACM International Conference on Multimedia* (Seattle, WA, USA) *(MM '20)*. Association for Computing Machinery, New York, NY, USA, 3632–3641. https://doi.org/10.1145/3394171.3413699

[32] Serhan Gül, Dimitri Podborski, Thomas Buchholz, Thomas Schierl, and Cornelius Hellge. 2020. Low-Latency Cloud-Based Volumetric Video Streaming Using Head Motion Prediction. In *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video* (Istanbul, Turkey) *(NOSSDAV '20)*. Association for Computing Machinery, New York, NY, USA, 27–33. https://doi.org/10.1145/3386290.3396933

[33] Markus Hahn, Lars Krüger, and Christian Wöhler. 2008. 3D Action Recognition and Long-Term Prediction of Human Motion. In *Computer Vision Systems*, Antonios Gasteratos, Markus Vincze, and John K. Tsotsos (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 23–32.

[34] Chris Harrison and Anind K. Dey. 2008. Lean and Zoom: Proximity-Aware User Interface and Content Magnification. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Florence, Italy) *(CHI '08)*. Association for Computing Machinery, New York, NY, USA, 507–510. https://doi.org/10.1145/1357054.1357135

[35] Rorik Henrikson, Tovi Grossman, Sean Trowbridge, Daniel Wigdor, and Hrvoje Benko. 2020. Head-Coupled Kinematic Template Matching: A Prediction Model for Ray Pointing in VR. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3313831.3376489

[36] Niels Henze, Sven Mayer, Huy Viet Le, and Valentin Schwind. 2017. Improving Software-Reduced Touchscreen Latency. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services* (Vienna, Austria) *(MobileHCI '17)*. Association for Computing Machinery, New York, NY, USA, Article 107, 8 pages. https://doi.org/10.1145/3098279.3122150

[37] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[38] Rob J. Hyndman. 2011. *Moving Averages*. Springer Berlin Heidelberg, Berlin, Heidelberg, 866–869. https://doi.org/10.1007/978-3-642-04898-2_380

[39] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. 2020. A survey on contrastive self-supervised learning. *Technologies* 9, 1 (2020), 2.

[40] Vimal Kakaraparthi, Qijia Shao, Charles J. Carver, Tien Pham, Nam Bui, Phuc Nguyen, Xia Zhou, and Tam Vu. 2021. FaceSense: Sensing Face Touch with an Ear-Worn System. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 3, Article 110 (sep 2021), 27 pages. https://doi.org/10.1145/3478129

[41] Kelvin Kan, François-Xavier Aubet, Tim Januschowski, Youngsuk Park, Konstantinos Benidis, Lars Ruthotto, and Jan Gasthaus. 2022. Multivariate Quantile Function Forecaster. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 10603–10621.

[42] Angjoo Kanazawa, Jason Y Zhang, Panna Felsen, and Jitendra Malik. 2019. Learning 3d human dynamics from video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5614–5623.

[43] Yury Kartynnik, Artsiom Ablavatski, Ivan Grishchenko, and Matthias Grundmann. 2019. Real-time facial surface geometry from monocular video on mobile GPUs. *arXiv preprint arXiv:1907.06724* (2019).

[44] Pin-Sung Ku, Qijia Shao, Te-Yen Wu, Jun Gong, Ziyan Zhu, Xia Zhou, and Xing-Dong Yang. 2020. ThreadSense: Locating Touch on an Extremely Thin Interactive Thread. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376779

[45] Edward Lank, Yi-Chun Nikko Cheng, and Jaime Ruiz. 2007. Endpoint Prediction Using Motion Kinematics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '07)*. Association for Computing Machinery, New York, NY, USA, 637–646. https://doi.org/10.1145/1240624.1240724

[46] Steven M LaValle, Anna Yershova, Max Katsev, and Michael Antonov. 2014. Head tracking for the Oculus Rift. In *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 187–194.

[47] Joseph J. LaViola. 2003. Double Exponential Smoothing: An Alternative to Kalman Filter-Based Predictive Tracking. In *Proceedings of the Workshop on Virtual Environments 2003* (Zurich, Switzerland) *(EGVE '03)*. Association for Computing Machinery, New York, NY, USA, 199–206. https://doi.org/10.1145/769953.769976

[48] Jiandong Liang, Chris Shaw, and Mark Green. 1991. On temporal-spatial realism in the virtual reality environment. In *Proceedings of the 4th annual ACM symposium on User interface software and technology*. 19–25.

[49] Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. 2021. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting* 37, 4 (2021), 1748–1764.

[50] Ruibo Liu, Qijia Shao, Siqi Wang, Christina Ru, Devin Balkcom, and Xia Zhou. 2019. Reconstructing Human Joint Motion with Computational Fabrics. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 1, Article 19 (mar 2019), 26 pages. https://doi.org/10.1145/3314406

[51] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, Puneet Agarwal, et al. 2015. Long short term memory networks for anomaly detection in time series. In *Proceedings*, Vol. 89. 89–94.

[52] Firas Massaad, Thierry M Lejeune, and Christine Detrembleur. 2007. The up and down bobbing of human walking: a compromise between muscle work and efficiency. *The Journal of physiology* 582, 2 (2007), 789–799.

[53] Alejandro Melendez-Calderon, Camila Shirota, and Sivakumar Balasubramanian. 2021. Estimating movement smoothness from inertial measurement units. *Frontiers in Bioengineering and Biotechnology* 8 (2021), 558771.

[54] Prapanna Mondal, Labani Shit, and Saptarsi Goswami. 2014. Study of effectiveness of time series modeling (ARIMA) in forecasting stock prices. *International Journal of Computer Science, Engineering and Applications* 4, 2 (2014), 13.

[55] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning* (Haifa, Israel) *(ICML'10)*. Omnipress, Madison, WI, USA,

807–814.

[56] Mathieu Nancel, Stanislav Aranovskiy, Rosane Ushirobira, Denis Efimov, Sebastien Poulmane, Nicolas Roussel, and Géry Casiez. 2018. Next-Point Prediction for Direct Touch Using Finite-Time Derivative Estimation. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18)*. Association for Computing Machinery, New York, NY, USA, 793–807. https://doi.org/10.1145/3242587.3242646

[57] Mathieu Nancel, Daniel Vogel, Bruno De Araujo, Ricardo Jota, and Géry Casiez. 2016. Next-Point Prediction Metrics for Perceived Spatial Errors. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 271–285. https://doi.org/10.1145/2984511.2984590

[58] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2020. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*.

[59] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2021. Meta-learning framework with applications to zero-shot time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 9242–9250.

[60] Niall O'Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. 2020. Deep learning vs. traditional computer vision. In *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Volume 1 1*. Springer, 128–144.

[61] Phillip T. Pasqual and Jacob O. Wobbrock. 2014. Mouse Pointing Endpoint Prediction Using Kinematic Template Matching. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 743–752. https://doi.org/10.1145/2556288.2557406

[62] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[63] Hendrik Purwins, Bo Li, Tuomas Virtanen, Jan Schlüter, Shuo-Yiin Chang, and Tara Sainath. 2019. Deep learning for audio signal processing. *IEEE Journal of Selected Topics in Signal Processing* 13, 2 (2019), 206–219.

[64] Aistis Raudys, Vaidotas Lenčiauskas, and Edmundas Malčius. 2013. Moving averages for financial data smoothing. In *International conference on information and software technologies*. Springer, 34–45.

[65] J.M. Rehg and T. Kanade. 1994. DigitEyes: vision-based hand tracking for human-computer interaction. In *Proceedings of 1994 IEEE Workshop on Motion of Non-rigid and Articulated Objects*. 16–22. https://doi.org/10.1109/MNRAO.1994.346260

[66] Miguel Fabián Romero Rondón, Lucile Sassatelli, Ramón Aparicio Pardo, and Frédéric Precioso. 2020. Track: a Multi-Modal Deep Architecture for Head Motion Prediction in 360° Videos. In *2020 IEEE International Conference on Image Processing (ICIP)*. 2586–2590. https://doi.org/10.1109/ICIP40778.2020.9191331

[67] Nicolas Roussel. 2022. One Euro Filter Implementation. https://cristal.univ-lille.fr/~casiez/1euro/. Accessed: 2023-02-13.

[68] Hasim Sak, Andrew W Senior, and Françoise Beaufays. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. (2014).

[69] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting* 36, 3 (2020), 1181–1191.

[70] Min-Woo Seo, Song-Woo Choi, Sang-Lyn Lee, Eui-Yeol Oh, Jong-Sang Baek, and Suk-Ju Kang. 2017. Photosensor-based latency measurement system for head-mounted displays. *Sensors* 17, 5 (2017), 1112.

[71] Qijia Shao, Amy Sniffen, Julien Blanchet, Megan E. Hillis, Xinyu Shi, Themistoklis K. Haris, Jason Liu, Jason Lamberton, Melissa Malzkuhn, Lorna C. Quandt, James Mahoney, David J. M. Kraemer, Xia Zhou, and Devin Balkcom. 2020. Teaching American Sign Language in Mixed Reality. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 4, Article 152 (dec 2020), 27 pages. https://doi.org/10.1145/3432211

[72] Alexa F. Siu, Eric J. Gonzalez, Shenli Yuan, Jason B. Ginsberg, and Sean Follmer. 2018. ShapeShift: 2D Spatial Manipulation and Self-Actuation of Tabletop Shape Displays for Tangible and Haptic Interaction. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3173865

[73] Jan-Philipp Stauffert, Florian Niebling, and Marc Erich Latoschik. 2020. Latency and cybersickness: impact, causes, and measures. A review. *Frontiers in Virtual Reality* 1 (2020), 582204.

[74] Jan-Philipp Stauffert, Florian Niebling, and Marc Erich Latoschik. 2020. Latency and cybersickness: Impact, causes, and measures. A review. *Frontiers in Virtual Reality* 1 (2020), 582204.

[75] Al Sweigart. 2019. PyAutoGUI. https://pyautogui.readthedocs.io/en/latest/. Accessed: 2022-07-27.

[76] Thorlabs. 2022. MVL5WA. https://www.thorlabs.com/thorproduct.cfm?partnumber=MVL5WA. Accessed: 2023-02-13.

[77] Tran Huy Vu, Archan Misra, Quentin Roy, Kenny Choo Tsu Wei, and Youngki Lee. 2018. Smartwatch-Based Early Gesture Detection 8 Trajectory Tracking for Interactive Gesture-Driven Applications. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 1, Article 39

(mar 2018), 27 pages. https://doi.org/10.1145/3191771

[78] Robert Y. Wang and Jovan Popović. 2009. Real-Time Hand-Tracking with a Color Glove. *ACM Trans. Graph.* 28, 3, Article 63 (jul 2009), 8 pages. https://doi.org/10.1145/1531326.1531369

[79] Greg Welch, Gary Bishop, et al. 1995. An introduction to the Kalman filter. (1995).

[80] Matt Whitlock, Ethan Harnner, Jed R Brubaker, Shaun Kane, and Danielle Albers Szafir. 2018. Interacting with distant objects in augmented reality. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 41–48.

[81] Wikipedia. 2023. Exponential smoothing. https://en.wikipedia.org/wiki/Exponential_smoothing. Accessed: 2022-11-03.

[82] Andrew D Wilson. 2007. Sensor-and recognition-based input for interaction. In *The Human-Computer Interaction Handbook*. CRC Press, 203–226.

[83] Jiann-Rong Wu and Ming Ouhyoung. 2000. On latency compensation and its effects on head-motion trajectories in virtual environments. *The visual computer* 16, 2 (2000), 79–90.

[84] Haijun Xia, Ricardo Jota, Benjamin McCanny, Zhe Yu, Clifton Forlines, Karan Singh, and Daniel Wigdor. 2014. Zero-Latency Tapping: Using Hover Information to Predict Touch Locations and Eliminate Touchdown Latency. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA) *(UIST '14)*. Association for Computing Machinery, New York, NY, USA, 205–214. https://doi.org/10.1145/2642918.2647348

[85] Ye Yuan and Kris Kitani. 2019. Ego-pose estimation and forecasting as real-time pd control. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 10082–10092.

[86] Jason Y Zhang, Panna Felsen, Angjoo Kanazawa, and Jitendra Malik. 2019. Predicting 3d human dynamics from video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7114–7123.

[87] Weidong Zhao, David A Stevens, Aleksandar Uzelac, Hrvoje Benko, and John L Miller. 2017. Prediction-based touch contact tracking. US Patent 9,542,092.

[88] Dingyun Zhu, Tom Gedeon, and Ken Taylor. 2011. Exploring Camera Viewpoint Control Models for a Multi-Tasking Setting in Teleoperation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 53–62. https://doi.org/10.1145/1978942.1978952

[89] Brian Ziebart, Anind Dey, and J. Andrew Bagnell. 2012. Probabilistic Pointing Target Prediction via Inverse Optimal Control. In *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces* (Lisbon, Portugal) *(IUI '12)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/2166966.2166968