FLOW: Filtering and LSTM-based Optimization for Web Browser Interactions

Conference Paper · February 2024

DOI: 10.1145/3629606.3629611

Dongyijie Primo Pan
Hong Kong University of Science and Technology
2 PUBLICATIONS **0** CITATIONS

SEE PROFILE

Long Ling
Tongji University
2 PUBLICATIONS **1** CITATION

SEE PROFILE

FLOW: Filtering and LSTM-based Optimization for Web Browser Interactions

Dongyijie Pan
primojaypan@cuc.edu.cn
School of Animation and Digital Arts, Communication University of China
Beijing, Chaoyang, China

Zhengnan Li
lzhengnan389@gmail.com
School of Animation and Digital Arts, Communication University of China
Beijing, Chaoyang, China

Long Ling
lucyling0224@gmail.com
College of Design and Innovation, Tongji University
Shanghai, China

Guangzheng Fei
gzfei@cuc.edu.cn
School of Animation and Digital Arts, Communication University of China
Beijing, Chaoyang, China

ABSTRACT

Web browsers have become an indispensable tool for human-computer interactions across various platforms. However, interactive performance (e.g the immediacy of interaction, the accuracy of interaction data) in browser-based apps can be hindered by hardware limitations and network delays, resulting in unwanted jitter and input signal delays that negatively impact the user experience. To tackle these challenges, we propose a novel approach that leverages effective filtering algorithms and lightweight machine learning models. By integrating the 1 euro filter and Long Short-Term Memory (LSTM) algorithm, we successfully optimize the accuracy of Mediapipe-based gesture interactions and TUIO-based tangible interactions within web browsers. Our findings demonstrate that this combined approach not only enhances input signal accuracy but also ensures seamless and highly responsive web browser interactions.

CCS CONCEPTS

• **Human-centered computing** → *Ubiquitous computing*; • **Computing methodologies** → *Machine learning algorithms.*

KEYWORDS

Web Browser, Accurate Recognition of Human Motion, Tangible Interaction, Tangible Interaction

ACM Reference Format:
Dongyijie Pan, Zhengnan Li, Long Ling, and Guangzheng Fei. 2023. FLOW: Filtering and LSTM-based Optimization for Web Browser Interactions. In *Chinese CHI 2023 (CHCHI 2023), November 13–16, 2023, Denpasar, Bali, Indonesia.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3629606.3629611

1 INTRODUCTION

Web browser stands as one of the most pervasive applications worldwide, enabling a multitude of human-computer interactions. Beyond traditional keyboard and mouse inputs, modern browsers support a wide range of novel interaction methods, including natural interactions based on computer vision for human motion capture, tangible interactions based on a variety of innovative sensors and pattern recognition algorithms, and so on.

However, when we introduced various novel interaction methods into browsers, we found that browser-based applications' performance when handling these trendy human-computer interaction patterns were far behind many native applications. [13] Specifically, these browser-based applications produced more jitter and latency in interaction data.

There are three main sources of interaction data jitter and latency, network factors, devices factors and processing factors. [16] [23] Due to the cross-platform nature of browser-based applications, different devices have different hardware and software conditions. Native applications have direct access to a device's hardware and system APIs. This provides native applications with more efficient data processing and computational capabilities. Web applications, however, operate within the confines of a browser's sandbox environment, which limits their access to device resources and can consequently lead to lower performance. [20] Javascript has many characteristics that can drag down machine processing performance, such as its poor handling of precision. [25] As an interpreted language that is parsed and executed at runtime. JavaScript operates on a single thread, meaning that only one task can be executed at a time. Despite leveraging an event loop and asynchronous handling, its performance lags behind multi-threaded native applications when it comes to extensive computations or multitasking. [3] [10] Another example is the instability of browser applications, which can be caused in part by the dynamic loading of interface resources. [26] As a result, native applications typically outperform web applications in terms of execution speed.

Building upon these challenges, we propose "FLOW", a method to universally enhance the interactive experience on the browser side by combining the 1 euro filter, a low time complexity filtering algorithm, with a lightweight Long Short-Term Memory (LSTM) algorithm. With the help of TensorFlow.js[28](a front-end machine learning library) we eliminate the need for additional development languages and environments, conducting a comprehensive data collection, training, and experimentation process entirely within the JavaScript language environment. Our approach optimized data jitter and latency introduced during the interaction between browser-side MediaPipe BlazeHands and TUIO-based interactive table tokens. This offers enhanced convenience for front-end developers and provides an optimization method that is well aligned with the front-end development ecosystem.

Contributions of this research can be summarized thus:

- We proposed a light-weight algorithm with effective combination of the 1 euro filter and LSTM. It provides a very general solution paradigm for the data jitter and latency in human–computer interaction

2 RELATED WORK

2.1 Early methods of addressing jitter and latency

The detrimental effect of jitter and latency on HCI has been investigated from different viewpoints. In Meehan et al.'s study, it was found that instabilities such as jitter and latency in interactive systems can affect physiological indicators in users, such as heart rate.[21] If we need to use this kind of natural interaction on the web to build some more complex things, such as gesture-controlled first-person shooter games, we need to keep the latency under 100ms to satisfy the user's interactive experience.[27] [9]

Early methods of addressing jitter and latency in human-computer interaction involved using algorithms such as the moving average. [22][32]These types of algorithms work by calculating an average over a period of time to offset noise and reduce data volatility. However, they have significant limitations; while they can smooth data, they also introduce significant delay, particularly when dealing with rapidly changing data, the results can be less than ideal.

2.2 Various filters to smooth data

To address these issues, researchers started to utilize more sophisticated filtering algorithms, like the Kalman filter[31]. The operation of the Kalman filter is predicated on a 'predict-correct' model: initially predicting the system's next state, followed by a correction of the prediction with actual observations. This process not only smooths the data, reducing noise, but also imparts the filter with some predictive capabilities.

However, the use and fine-tuning of the Kalman filter necessitates the manipulation of numerous parameters. Applying it within the realm of human-computer interaction requires substantial experience with data signal processing and a solid understanding of linear algebra. Moreover, the Kalman filter mandates the storage of state vectors and covariance matrices for each observation node. As such, numerous matrix operations are involved in the filtering process, making its time complexity and space complexity both nonlinear. This may pose challenges when it comes to real-time and space-sensitive applications.

The 1 euro filter[7] has gained attention due to its superior performance in dealing with human-computer interaction issues. This filter is a low-pass filter with dynamically adjusted cutoff frequency. Its main advantage lies in its ability to find a good balance between smoothing data and maintaining system responsiveness. These characteristics make the 1 euro filter highly suitable for application in the field of human-computer interaction.

In recent research, Yiqiao Lin et al.introduced an improvement to Mediapipe BlazePose by incorporating the 1 euro filter.[18] It resulted in a significant increase in accuracy, elevating it from 32.43% to an impressive 95.87% in a C++ based environment. However, when the system latency is sufficiently high in the browser, the most that filtering algorithms can achieve is to moderately smooth the data and reduce jitter, thus only alleviating latency to a certain extent.

2.3 Combination of filter and machine learning

Combination of filter algorithm and machine learning has proven to be extremely powerful in human motion tracking. [6] Liu et al.'s work proposes a novel human motion prediction framework that combines a recurrent neural net-work (RNN) and inverse kinematics (IK) to predict human arm motion. A modified Kalman filter (MKF) is applied to adapt the model online and the demonstrate that the proposed method improves the prediction accuracy by approximately 14%[19][2]. In Kundu et al.'s work, they employed methods like Convolutional Long Short-Term Memory (ConvLSTM), Kalman Filter, and Dead Reckoning algorithms for optimizing latency in VR systems, allowing them to comprehensively compare prediction errors. This effectively enhanced the real-time interactivity of VR and AR systems.[17] Nevertheless, the web browser environment presents unique challenges due to its inherent limitations, particularly when implementing larger and more complex algorithms and models that are typically optimized for controlled laboratory settings. For instance, the online-adapted RNN network with IK and MKF, as proposed by Liu et al., can only predict data for the subsequent frame based on the previous N frames. The issue arises when there is a need to forecast data for multiple future frames; in this context, the 'N to 1' prediction methodology becomes considerably slow within the browser environment.

While nowadays, the Transformer algorithm, with its focus on the attention mechanism, is popular and broadly used in 'Seq2Seq' (Sequence to Sequence) trajectory prediction [29] [1] [11]. But the Transformer model is too heavy for web browser environment. Though it's powerful, RNN may be the best solution in this era.

3 METHODS

3.1 Derivation of the 1 euro filter

In our approach, when faced with various unstable interactive data in the browser, the first thing to do is to eliminate the visually obvious jitter, and to ensure that the delay introduced by the filtered data is as minimal as possible. For the coordinate data of each dimension, we consider it as a one-dimensional time-series input signal. We will apply an improved 1 euro filter to smooth the coordinates of the input data. It is essentially an adaptive low-pass filter, represented by the following equation:

$$\widehat{X_i} = \alpha * X_i + (1 - \alpha) * \widehat{X}_{i-1}, i \geq 2 \quad (1)$$

X represents any coordinate we are processing in the interaction data, with X_i denoting its observed value at time T_i. We define the coordinates processed by the filter as $\widehat{X_i}$. This α coefficient is an adaptively adjusted parameter through our subsequent algorithm, which balances jitter and latency. When jitter is the main influencing factor, it will increase, thereby reducing the weight of the previous frame; when latency is severe, it will decrease to increase the weight of the current frame's observed signal.

To make α adaptive, we define it as follows:

$$\alpha = \frac{1}{1 + \frac{\tau}{\Delta T}} \tag{2}$$

The ΔT stands for the sampling period which can be computed from timestamps. Then the τ variable is the key, which is defined as:

$$\tau = \frac{1}{2\pi f_c} \tag{3}$$

$$f_c = f_{c_{min}} + \beta|\widehat{X_i}'| \tag{4}$$

Thus, by substituting the previous equations into Equation 1, we obtain:

$$\widehat{X_i} = (\frac{1}{f_c} * \frac{\widehat{X_{i-1}}}{2\pi\Delta T} + \widehat{X_i})\frac{1}{1 + \frac{1}{f_c} * \frac{1}{2\pi\Delta T}} \tag{5}$$

Through Equation 5, we discover that this is an algorithm with a time complexity of $O(1)$. The only parameter we can alter is f_c, which itself is determined by two other parameters: $f_{c_{min}}$ and β. They have clear conceptual relationships. Based on the principle of this algorithm, when the lag is the problem, β should be increased. On the other hand, when jitter becomes the primary influencing factor, just decrease $f_{c_{min}}$. The default setting of β is 0, and for $f_{c_{min}}$ is 1 Hz.

This algorithm is so succinct and elegant that we can encapsulate it thoroughly within just a few dozen lines of JavaScript code. However, even though this simple algorithm can strike a certain balance between jitter and latency to some extent, many problems we encounter in the real world are unavoidable due to the reasons mentioned earlier, and latency is still inevitable, no matter how we adjust the parameters.

3.2 Temporal trajectory prediction

In our approach, the temporal trajectory prediction of filtered data will play a key role. Temporal trajectory prediction based on various algorithms has been widely employed in diverse research domains, such as autonomous driving and pedestrian trajectory prediction.

Based on the condition of using TensorFlow.js, we have chosen the LSTM algorithm for temporal trajectory prediction. Long Short-Term Memory (LSTM) is a distinctive variant of Recurrent Neural Networks (RNNs) designed to address the vanishing gradient problem. By incorporating memory cells and gating mechanisms, LSTM excels in modeling long-term dependencies in sequential data, making it ideal for tasks involving temporal relationships. LSTM stands as a pivotal element in modern deep learning research, revolutionizing how sequential data is processed and understood.

In temporal trajectory prediction, LSTM can be employed to implement a 'Seq2Seq' approach, where we use past trajectory data to predict future trajectory data. We can certainly build an 'N to 1' prediction model, which predicts the next coordinate position based on the previous N steps, but such a sliding window prediction method is too time-consuming under low computational power. We recommend web developers to turn the 1 euro filter for reducing significant jitter, and then perform latency statistics on the filtered data. For example, under a fixed sampling rate, we observe how many frames the filtered data lags behind the ground truth. We found that using the maximum lag value multiplied by 1.5 or 2 as input to the LSTM yields significantly improved prediction accuracy.

The neural network structure we have built consists of two LSTM layers, each comprising 140 cells. The input and output data sizes of our network depend on the level of latency in the interactive system that we want to optimize. Assuming we aim to construct a model that predicts the next 15 frames based on the previous 30 frames, our neural network contains approximately 250,000 parameters. Without GPU acceleration, a single prediction can be completed in about 10ms, which means that even in a rendering scenario of 60 frames, this prediction would not significantly impact performance. This is the significance of adopting a "lightweight neural network."

We then adjust the dimensions of the input tensor based on the relevant information for each prediction target, such as whether it involves two-dimensional or three-dimensional coordinates and timestamps.Though on the browser side, the absence of GPU acceleration may slow down the training process, you can leverage TensorFlow based on Node.js to gain GPU access.

3.3 Combination of the 1 euro filter and LSTM

However, our LSTM network only outputs a single potential coordinate sequence. In many real-world human-computer interaction scenarios, the latency in frame numbers is often not a fixed value, and it can vary within a certain range, such as [5, 15] frames. This variability in latency makes it challenging to choose a single fixed "prediction frame" to determine the final position of the predicted data. In some experiments, we attempted to use an "averaged" fixed frame as our "prediction frame," which alleviated the latency issue to some extent. However, in many cases, it led to significant jitter, especially during the final stages of a moving trajectory.

Jota et al. decomposed motion capture, particularly on-screen motion capture, into three phases:initial reaction, large ballistic motion, and “feedback-adjusted final adjustments” where the finger is stopped inside the a target area.[14] [8] They found that the last phase was the most affected by latency. In our experiments, we also observed a similar behavior. After a high-speed movement comes to an end, the previous prediction methods we used often exhibit a noticeable "overshooting" effect, as if having inertia, departing from the actual position, and then bouncing back and forth like a ball towards the ground truth. If we rely on a fixed prediction frame, assuming a fixed system error in our interaction system, this limitation seems challenging to avoid.

We then focused on the changes of smoothing coefficient, α, in Equation 1. We found that it was highly sensitive to changes in acceleration and deceleration movements. This led us to the idea of using this parameter to determine the number of frames to lag behind. In each experiment's data collection phase, we needed to evaluate the range of delays for the filtered data. Despite assuming a fixed delay value for the system during the interaction, the perceived latency by the user is also positively correlated with speed (as higher speeds result in larger relative distances between the current point and the ground truth). However, after linearly mapping the α distribution with the delay range, we observed that this simple mapping could largely alleviate the issue of 'overfitting'.

For HCI researchers who want to realize their crazy ideas in the browser or those who wish to leverage mature open-source models like Mediapipe but face insurmountable gaps between browser-based model performance and native systems, we often need to adjust, retrain, and perform transfer learning on existing tracking algorithms that were not originally optimized for browser development in our specific application scenarios. However, because our own datasets and lightweight algorithms cannot achieve perfection, we need to apply another round of smoothing to the coordinate points provided by these models to make occasional offset prediction points more accurate. So, how can we filter the motion curve to better match the real human movement trajectory? We go back to the beginning and use the 1 euro filter once again. This time, since the previous filtering and neural network have already learned features like coordinates, velocity, and acceleration, we can simply use the default parameters to complete our prediction work. So we add the final filter that with $\beta = 0$, and for $f_{c_{min}} = 1Hz$ and finish our job.

4 EXPERIMENTS AND RESULTS

There are a lot of browser-side applications that use non-traditional interaction methods. Our experiments are mainly aimed at two different application scenarios, which are hand motion capture based on Mediapipe and token interactive table application based on TUIO protocol.

4.1 Mediapipe Hands

4.1.1 *Introduction.*

Mediapipe Hands[34] is a real-time on-device hand tracking solution that predicts a hand skeleton of a human from a single RGB cameraIt can detect 21 3D landmarks of the hands in an image or a continuous stream. Since Mediapipe is a framework for building cross-platform multimodal applied ML pipelines, the Mediapipe Hands model can be used on a variety of platforms, including Android, iOS, Web and desktop PCs.

4.1.2 *Experiment Environment.*

Due to the considerable accuracy of Mediapipe Hands based on GPU acceleration, we used it as the ground truth to assess the extent of our optimization on the browser side. For experiments with GPU acceleration, our hardware environment consisted of an Intel Core i7-8750H CPU, 16GB of memory, a GeForce GTX 3060 Ti GPU with 8GB of memory. The operating system used was Ubuntu 22.04, with Mediapipe version 0.10.3 and Python version 3.10.

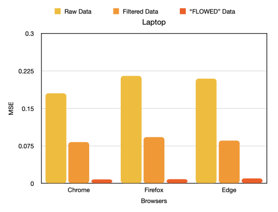
For the control group of hardware devices, we used the Lenovo Legion 7 Slim laptop with AMD Ryzen 7 5800H, GeForce RTX 3060 Laptop GPU, the MacBook Pro with the M1 chip, iPhone 13 with 256GB memory, and an Android smartphone with a meager 12GB LPDDR5 RAM. The browser version used on all devices was the latest, and GPU acceleration mode was not used. All control group hardware devices used their built-in standard RGB monocular cameras (front-facing cameras).

For the training data, we adjusted the filter parameters specifically and estimated the system latency on different platforms and browsers. Despite the differences in devices, we found that setting the filter parameters $f_{c_{min}}$ to 0.003 and β to 0.01 can visually reduce the jitter to the greatest extent.As for system latency, mobile devices generally have greater latency than computers, but it is not extremely severe.

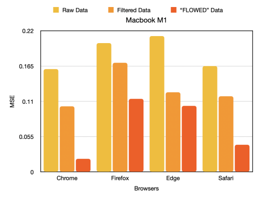
For the test data, we collected 30-second videos at 30 frames per second from volunteers who provided real-time hand videos. The age of the volunteers ranges from 18 to 45 years old, including both men and women.

4.1.3 *Results.*

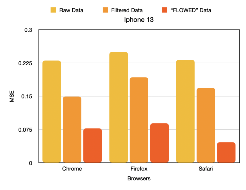
For the training dataset, after data set partitioning, our LSTM model achieved a prediction accuracy of 97.6% (calculated based on the loss function in tensorflow) for the subsequent 10 frames. Our experiment will save the data of each frame in the test video on the browser side, while also saving the video. Then, the video is calculated offline under conditions where the experimental environment has backend acceleration. We determine the superiority of the algorithm by first calculating the Euclidean distance of the spatial coordinates of each palm node in each frame. Then, the mean square error (MSE) of the 21 points in each frame is calculated. MSE = $\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$. According to this calculation method, the smaller the MSE, the smaller the gap between our optimization results and the true value. Then, for different browsers on different hardware, we calculate the average MSE of each frame running on them as our performance estimate.



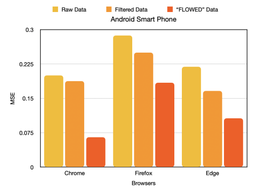
(a) laptop



(b) Macbook



(c) iPhone



(d) Andorid

Figure 1

In the Figure, we list the original Mediapipe data on the Javascript side, the data processed by our parameter-tuned 1 euro filter, and the data after the method combining LSTM and 1 euro filter we discussed above, namely "FLOW", for different hardware and browser platforms respectively. It can be seen that although there are still differences between different platforms, our method has greatly reduced the instability of Mediapipe Hands data on the browser side in terms of numerical values for each environment.

4.2 TUIO based token interactive table application

4.2.1 *Introduction.*

TUIO (Tangible User Interface Objects) [15]is a protocol specifically designed for transmitting the state of tangible objects and multi-touch events that are generated by interactive table surfaces and walls. TUIO is an open framework and has become a de-facto standard in the field of interactive surfaces. It is used in a wide array of projects around the globe, ranging from interactive tables to full-room installations.

In our experiment, we utilized TUIO signals to configure metal contacts in a specific geometric arrangement within a tangible token. Our recognition program is based on capacitive pattern methods.[30] [12] The token recognition program can identify the tangible token's number and positional information. This setup enables a multitude of cool interactive designs and developments to be performed on a touchscreen using this tangible token.

However, even the most advanced token recognition programs currently available suffer from a considerable degree of instability, such as losing individual touch point data. In browser-based applications, these recognition programs need to set up a local server to send UDP signals to the browser to receive token coordinate information, which may introduce additional latency in the application. In practical applications, the coordinate information of tangible tokens frequently experiences jitter and severe latency, leading to a suboptimal user experience. Our method not only mitigates the instability of browser-based computer vision applications, but it also significantly enhances the user experience of these tangible interactions.

4.2.2 *Experiment Environment.*

For this experiment, we chose a 55-inch LED capacitive touchscreen with a response time of 5ms, a maximum resolution of 3840*2160, and a screen capable of detecting up to 40 touch points simultaneously. The computer host is equipped with an Intel Core i7 9700 processor, 16GB of memory, and an RTX 3070 graphics card, the operation system is Windows10 and the browser version is Chrome 113 beta.

Measuring jitters or latency entails using a camera, to capture the input motion and the associated feedback. [4] In this experiment, we utilized a camera to capture video footage of the token during operations, aligning the camera's coordinate system with that of the screen. The square rendered on the screen was centered at the token's midpoint coordinates. We employed object tracking from the OpenCV library to ascertain the actual position of the token. Analysis was performed by contrasting the disparities between the coordinates rendered by the browser at the same moment and the ground truth coordinates. In this experiment, we primarily collected interaction data from four different adult users for data processing, training, and evaluation.

4.2.3 *Results.*

In our application, we initially tuned the filter parameters $\alpha = 0.001$ $\beta = 0.1$ to minimize the discrepancy between the filtered data values and the ground truth. Through data analysis of various token movement actions over a 30-minute period, we found that the overall delay ranged between 7 to 16 frames.

According to the theory of Jota et al. , we categorized the collected data into three stages according to the acceleration of the token in each frame. **1:initial reaction** , **2:large ballistic motion**, **3:feedback adjusted final adjustments**. Based on these three criteria, we have organized the data into the following table: original data, data after 1 Euro filter processing, data after prediction using LSTM + mean latency frame, and data after α-mapped frame and subsequent filtering

Table 1: MSE of three stages

Methods	Stage1	Stage2	Stage3
Original	0.000	0.3215	0.1817
1 euro filter	0.000	0.2915	0.1734
LSTM+Mean Frame	0.002	0.0926	**0.1926**
FLOW(**Our Method**)	0.000	0.0515	0.0749

From the experimental data, we observe that the data predicted after using LSTM + Mean Frame can effectively reduce latency during 'large ballistic motion'. Visually, the rendering position appears to closely follow the token's movement at high speed. However, the LSTM predictions without further filtering and processing resulted in pronounced data jitter during Step 1 and Step 3, indicating the onset and termination stages of motion. This overfitting phenomenon, as mentioned earlier, becomes particularly prominent in the later stages of motion. This effect is also evident in the data, where the accuracy of predictions in Step 3 is even lower after undergoing LSTM + Mean Frame processing.

After applying the FLOW method, we effectively mitigated the overfitting issue caused in Step 3. This phenomenon is more pronounced in scenarios with greater system latency, and we did not encounter such severe overfitting in the experiments conducted with Mediapipe. The experimental data validates the reliability of our approach. This significantly enhances the interaction experience when using physical interaction tokens. We are currently planning to develop a table ice hockey game for browser-based interaction, utilizing tokens and an interactive table. This optimization will enable us to create more engaging applications on the browser platform.

5 CONCLUSION AND FUTURE WORK

FLOW provides a feasible and experimentally implementable paradigm to address jitter and latency in complex human-computer interaction applications on the browser platform. The 1 euro filter serves as a preprocessor before using neural networks, and it dynamically selects predicted values and further smooths the output results after the neural network's predictions. Moreover, our optimized and integrated algorithm maintains a lightweight profile in terms of both time and space complexity. In application scenarios where prompt human-computer interaction feedback is crucial, our algorithm effectively addresses the two issues encountered in the experiment. We believe it holds the potential to be applied in various interactive applications on the browser platform.

But for HCI researchers and individuals frequently working in small teams, it's often challenging to amass a substantial quantity of high-quality interaction data tailored to specific scenarios. Many times, building upon this paradigm requires more specific algorithms, such as those addressing the topological structure of human joints and the kinematics of motion, to further optimize our paradigm.

In the future, browser-based applications will undoubtedly possess even greater performance and computational capabilities. The continuous advancement of hardware and communication technologies will enhance the real-time nature of human-computer interaction applications. For frontend developers themselves, from the concept of Progressive Web Apps (PWA)[5] to WebAssembly, from leveraging WebGL[24] to access GPUs to the emerging prominence of WebGPU, we are continually exploring the performance limits of browser-based applications, striving to bridge the performance gap between browser-based applications and native system applications.

In the coming years, we would have the opportunity to incorporate more complex yet powerful machine learning models into browsers. Within the active Tensorflow.js community, numerous HCI researchers and enthusiasts collaborate to push the boundaries of browsers. At the same time, the HCI researchers and their achievements, like $1 Unistroke Recognizer[33], before the surge of machine learning algorithms also remind us that apart from feeding interaction data into black-box-like hidden layers, we humans still possess countless ingenious ideas to solve seemingly challenging problems using concise code and remarkably low time complexity. Together, we will expand the possibilities of the browser, the most fundamental medium for communication between humans and computers, and humans and the internet. The potential for the future is immense.

REFERENCES

[1] Emre Aksan, Manuel Kaufmann, Peng Cao, and Otmar Hilliges. 2021. A spatio-temporal transformer for 3d human motion prediction. In *2021 International Conference on 3D Vision (3DV)*. IEEE, 565–574.

[2] Andreas Aristidou, Joan Lasenby, Yiorgos Chrysanthou, and Ariel Shamir. 2018. Inverse kinematics techniques in computer graphics: A survey. In *Computer graphics forum*, Vol. 37. Wiley Online Library, 35–58.

[3] Michael Bebenita, Florian Brandner, Manuel Fahndrich, Francesco Logozzo, Wolfram Schulte, Nikolai Tillmann, and Herman Venter. 2010. SPUR: a trace-based JIT compiler for CIL. In *Proceedings of the ACM international conference on Object oriented programming systems languages and applications*. 708–725.

[4] François Bérard and Renaud Blanch. 2013. Two touch system latency estimators: high accuracy and low overhead. In *Proceedings of the 2013 ACM international conference on Interactive tabletops and surfaces*. 241–250.

[5] Andreas Biørn-Hansen, Tim A Majchrzak, and Tor-Morten Grønli. 2017. Progressive web apps: The possible web-native unifier for mobile development. In *International Conference on Web Information Systems and Technologies*, Vol. 2. SciTePress, 344–351.

[6] Pradipta Biswas, Gokcen Aslan Aydemir, Pat Langdon, and Simon Godsill. 2013. Intent recognition using neural networks and Kalman filters. In *International Workshop on Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data*. Springer, 112–123.

[7] 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*. 2527–2530.

[8] Elie Cattan, Amélie Rochet-Capellan, Pascal Perrier, and François Bérard. 2015. Reducing latency with a continuous prediction: Effects on users' performance in direct-touch target acquisitions. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces*. 205–214.

[9] Seong-Ping Chuah and Ngai-Man Cheung. 2014. Layered coding for mobile cloud gaming. In *Proceedings of International Workshop on Massively Multiuser Virtual Environments*. 1–6.

[10] Andreas Gal, Brendan Eich, Mike Shaver, David Anderson, David Mandelin, Mohammad R Haghighat, Blake Kaplan, Graydon Hoare, Boris Zbarsky, Jason Orendorff, et al. 2009. Trace-based just-in-time type specialization for dynamic languages. *ACM Sigplan Notices* 44, 6 (2009), 465–478.

[11] Francesco Giuliari, Irtiza Hasan, Marco Cristani, and Fabio Galasso. 2021. Transformer networks for trajectory forecasting. In *2020 25th international conference on pattern recognition (ICPR)*. IEEE, 10335–10342.

[12] Zanzhen Huang, Minyong Shi, Guangzheng Fei, and Yaxin Zhu. 2020. PTPG: A Poisson Triangular Pattern Generator for Tokens on Tangible Surfaces. *IEEE Access* 8 (2020), 76019–76027.

[13] Abhinav Jangda, Bobby Powers, Arjun Guha, and Emery Berger. 2019. Mind the gap: Analyzing the performance of webassembly vs. native code. *arXiv preprint arXiv:1901.09056* (2019).

[14] Ricardo Jota, Albert Ng, Paul Dietz, and Daniel Wigdor. 2013. How fast is fast enough? a study of the effects of latency in direct-touch pointing tasks. In *Proceedings of the sigchi conference on human factors in computing systems*. 2291–2300.

[15] Martin Kaltenbrunner, Till Bovermann, Ross Bencina, Enrico Costanza, et al. 2005. TUIO: A protocol for table-top tangible user interfaces. (2005), 1–5.

[16] Teemu Kämäräinen, Matti Siekkinen, Antti Ylä-Jääski, Wenxiao Zhang, and Pan Hui. 2017. A measurement study on achieving imperceptible latency in mobile cloud gaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. 88–99.

[17] Ripan Kumar Kundu, Akhlaqur Rahman, and Shuva Paul. 2021. A study on sensor system latency in vr motion sickness. *Journal of Sensor and Actuator Networks* 10, 3 (2021), 53.

[18] Yiqiao Lin, Xueyan Jiao, and Lei Zhao. 2023. Detection of 3D Human Posture Based on Improved Mediapipe. *Journal of Computer and Communications* 11, 2 (2023), 102–121.

[19] Ruixuan Liu and Changliu Liu. 2020. Human motion prediction using adaptable recurrent neural networks and inverse kinematics. *IEEE Control Systems Letters* 5, 5 (2020), 1651–1656.

[20] Yun Ma, Dongwei Xiang, Shuyu Zheng, Deyu Tian, and Xuanzhe Liu. 2019. Moving deep learning into web browser: How far can we go?. In *The World Wide Web Conference*. 1234–1244.

[21] Michael Meehan, Sharif Razzaque, Mary C Whitton, and Frederick P Brooks. 2003. Effect of latency on presence in stressful virtual environments. In *IEEE Virtual Reality, 2003. Proceedings*. IEEE, 141–148.

[22] Brad A Myers, Rishi Bhatnagar, Jeffrey Nichols, Choon Hong Peck, Dave Kong, Robert Miller, and A Chris Long. 2002. Interacting at a distance: measuring the performance of laser pointers and other devices. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 33–40.

[23] Albert Ng, Julian Lepinski, Daniel Wigdor, Steven Sanders, and Paul Dietz. 2012. Designing for low-latency direct-touch input. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. 453–464.

[24] Tony Parisi. 2012. *WebGL: up and running*. " O'Reilly Media, Inc.".

[25] Marija Selakovic and Michael Pradel. 2016. Performance Issues and Optimizations in JavaScript: An Empirical Study. In *Proceedings of the 38th International Conference on Software Engineering* (Austin, Texas) *(ICSE '16)*. Association for Computing Machinery, New York, NY, USA, 61–72. https://doi.org/10.1145/2884781.2884829

[26] Marija Selakovic and Michael Pradel. 2016. Performance issues and optimizations in javascript: an empirical study. In *Proceedings of the 38th International Conference on Software Engineering*. 61–72.

[27] Shu Shi, Cheng-Hsin Hsu, Klara Nahrstedt, and Roy Campbell. 2011. Using graphics rendering contexts to enhance the real-time video coding for mobile cloud gaming. In *Proceedings of the 19th ACM international conference on Multimedia*. 103–112.

[28] Daniel Smilkov, Nikhil Thorat, Yannick Assogba, Charles Nicholson, Nick Kreeger, Ping Yu, Shanqing Cai, Eric Nielsen, David Soegel, Stan Bileschi, et al. 2019. Tensorflow. js: Machine learning for the web and beyond. *Proceedings of Machine Learning and Systems* 1 (2019), 309–321.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[30] Nicolas Villar, Daniel Cletheroe, Greg Saul, Christian Holz, Tim Regan, Oscar Salandin, Misha Sra, Hui-Shyong Yeo, William Field, and Haiyan Zhang. 2018. Project zanzibar: A portable and flexible tangible interaction platform. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13.

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

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

[33] Jacob O Wobbrock, Andrew D Wilson, and Yang Li. 2007. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and*

technology. 159–168.
[34] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. 2020. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214* (2020).