PredicTouch: A System to Reduce Touchscreen Latency using Neural Networks and Inertial Measurement Units

Huy Viet Le[1], **Valentin Schwind**[1], **Philipp Göttlich**[2], **Niels Henze**[1]
VIS, University of Stuttgart, Stuttgart, Germany
[1]{huy.le, valentin.schwind, niels.henze}@vis.uni-stuttgart.de, [2]inf87644@stud.uni-stuttgart.de

ABSTRACT
Touchscreens are the dominant input mechanism for a variety of devices. One of the main limitations of touchscreens is the latency to receive input, refresh, and respond. This latency is easily perceivable and reduces users' performance. Previous work proposed to reduce latency by extrapolating finger movements to identify future movements - albeit with limited success. In this paper, we propose *PredicTouch*, a system that improves this extrapolation using inertial measurement units (IMUs). We combine IMU data with users' touch trajectories to train a multi-layer feedforward neural network that predicts future trajectories. We found that this hybrid approach (software: prediction, and hardware: IMU) can significantly reduce the prediction error, reducing latency effects. We show that using a wrist-worn IMU increases the throughput by 15% for finger input and 17% for a stylus.

ACM Classification Keywords
H.5.2. Interfaces and Presentation: User Interfaces

Author Keywords
Latency; prediction; lag; touch input; IMU; neural network.

INTRODUCTION
Touchscreens are the primary input mechanism for a wide range of devices, including smartphones, tablets, car entertainment systems and ATMs. As all other input mechanisms, touchscreens have certain latency. As they combine input and output, touchscreens' latency is more visually apparent than the latency of other mechanisms, such as mouse or keyboard. When drawing a line or when dragging an object across the screen, even a small amount of touchscreen latency (below 10*ms*) is easy to perceive [30]. As shown Figure 1, latency results in a clearly visible gap between the finger or stylus held by the user and the corresponding response, such as a line or a following object. The higher the latency the larger the gap.

Touchscreens are built using several components and layers, each potentially responsible for introducing latency. When a user touches the screen, the registered input is passed through a pipeline of different processing components before visual feedback can be visualized on the screen. This results in unavoidable lag in the range of 50*ms* to 200*ms* [5, 18, 32] which is clearly visible and decrease users' performance [16].

ISS '17, October 17–20, 20167, Brighton, United Kingdom

ACM ISBN 978-1-4503-4691-7/17/10...$15.00
DOI: https://doi.org/10.1145/3132272.3134138

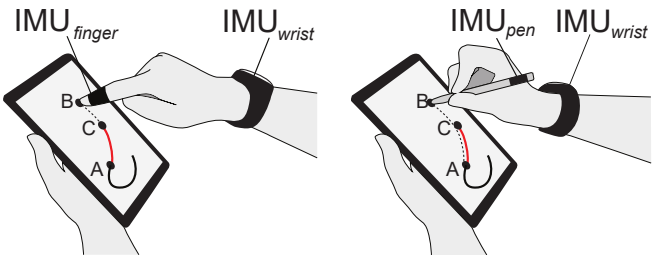


Figure 1. Sketch of *PredicTouch*. Point (A) shows the touch position registered by touchscreens due to latency. Point (B) shows the actual position of finger/stylus. Point (C) shows the touch position extrapolated by *PredicTouch* to reduce the visible latency of touchscreens.

Previous work proposed different approaches to reduce touchscreen latency. On the one side, hardware-based approaches use additional hardware such as high-speed cameras [32, 22] and motion capture systems [3] combined with high-speed projectors to visualize users' fingers with a latency down to 1*ms*. While these approaches work reliably, they require immobile external sensors that are only suitable for stationary setups but not mobile devices. On the other side, software-based approaches extrapolate the finger's movement [3, 11] to reduce the visual gap between finger and dragged object. The recent state-of-the-art approach from Henze *et al.* [11] uses artificial neural networks to predict the finger's future position based on the latest trajectory of the finger. Since software-based approaches do not rely on external sensors, they can be readily integrated into off-the-shelf mobile devices, such as commercial smartphones and tablets. However, these approaches induce a high amount of jitter due to prediction errors which can even decrease users' throughput (i.e. speed and accuracy). These limitations make both approaches unsuitable for latency reduction on off-the-shelf mobile devices.

In this paper, we present a hybrid approach that improves the software-based approach by Henze *et al.* [11] through additional sensors to capture the hand's micro-movements. As previous work already demonstrated the capability of IMU sensors to reconstruct hand and stylus movement in a wide range of use cases [14, 38, 10], we show that incorporating IMU data into a machine learning model that predicts the finger's future position significantly reduces the prediction error. Through a user study, we further show that the IMU data also helps to significantly increase users' throughput. As a wide range of commercially available technologies includes IMUs, such as smartwatches, smart pens or even smart clothes, we conclude that this low-cost approach can reduce latencies of ever-increasing touchscreen devices around us.

RELATED WORK

Prior work investigated the effect of latency on the user and proposed different latency reduction approaches. IMUs were used to reconstruct motion trajectories for several use cases.

Effect and Perception of Touch Latency

Prior work showed that latency negatively affects the user experience [17]. A body of work developed approaches to measure the latency of touchscreen devices [2, 5, 18] and showed that the latency of commercial touchscreen devices ranges from 50 *ms* to 200 *ms* [5, 13, 18, 32]. This is easily perceivable for humans. While Ritter *et al.* [34] showed that users considered latencies above 170 *ms* as unacceptable for dragging tasks, Deber *et al.* [6] reported that the threshold for noticing latency when dragging an object across the touchscreen is 11 *ms*. Ng *et al.* [30] further showed that users are able to distinguish between latency differences of 1 *ms*. Investigating the effect of latency on the user while using a stylus to draw, Annett *et al.* [1] found that even very low levels of latency down to 2 *ms* can be discriminated by the user.

Besides the perception of latency, researchers also investigate the effect of latency on the user's performance. Jota *et al.* [16] showed that latency of 25 *ms* already significantly decreases the user's performance measured by the throughput [26] in a Fitts' Law dragging task. Kaaresoja *et al.* [19] investigated different delays when entering sequences of numbers on a touchscreen, and found that keypads with a constant delay and the smallest feedback delay variation were faster to use and produced fewer errors in comparison to a wider delay variability. Further, users perceive buttons with longer delays as heavier, with a need for greater force when pressing [19].

Previous work repeatedly showed that even very small amounts of touchscreen latency are noticeable and reduces users' performance. In summary, touchscreen latency clearly has a negative effect on the user experience.

Reducing Touch Latency

As even a very low amount of latency is easily noticeable and affects the user's performance, researchers proposed different approaches to reduce latency. Previous approaches can be categorized into *hardware-based* and *software-based* approaches. Hardware-based approaches consist of additional hardware that captures the user's finger faster than capacitive touchscreens. For example, Ng *et al.* [31, 32] combined a high-speed camera to capture the user's hand with a high-speed projector to build a touchscreen with a latency of 1 *ms*. Similarly, Cattan *et al.* [3] used an optical marker tracking system to build a touch screen with 25 *ms* latency. Based on a projected capacitive architecture, Leigh *et al.* [25] developed a multi-touch sensor with a frame rate of 4000 *Hz* and latency of 40 microseconds. The sensor, however, is not transparent, bulky and consumes nine watts even without the high-speed projector. While these approaches resulted in low latencies, they are only suitable for stationary setups.

In contrast, software-based approaches use the measurements provided by the touchscreen and can be readily deployed into off-the-shelf touchscreen devices. Cattan *et al.* [3] reduced touchscreen latency by linearly extrapolating the finger movement. They showed that a large part of the latency's negative effect can be compensated when running their approach on a 25 *ms* latency system. However, using the prediction with latencies above 42 *ms* did not yield better performance. Similarly, Henze *et al.* [11] predict the finger movement on commercially available mobile devices with a high latency of 100 *ms*. They show that shallow neural networks can better predict a finger movement than linear interpolation. While the authors improved throughput by predicting the finger's position in 33 *ms*, the approach results in a high prediction error. This error induces a high amount of jitter which causes negative reactions and even reduces throughput when predicting a finger's position in 66 *ms*. We propose to reduce the predicting error with information about the user's hand movement captured by IMUs.

Using IMUs to Capture Micro-Movements

IMUs are widely available in a large number of commercial devices, including smartphones, smart watches, tablets, and even styli. Previous research used IMUs to reconstruct a hand model with eight IMUs placed on the hand [14]. Amongst others, IMUs capture the *micro-movements* of the objects in which they are integrated through acceleration and orientation. Researchers used this information for a wide range of use cases. For example, Wang *et al.* [38] and Hsu *et al.* [15] used an IMU integrated into a stylus to reconstruct its motion trajectory and for hand-written digit recognition. Similarly, Deselaers *et al.* [7] used the IMU integrated into smartphones to enable users to use their smartphones as a pen. When attached to the user's wrist, IMUs can be used for gesture recognition [23, 4], activity recognition [28] or to extend the input space on commodity touchscreens [39]. Using accelerometers on the wrists and hips, He *et al.* [10] showed that it is also possible to predict human movements.

DESIGN OF PREDICTOUCH

To reduce the error when reducing touchscreen latency by predicting the finger movement [11], we use inertial measurement units (IMUs) to feed the model with additional information about the user's hand or stylus movements. We refer to these movements as *micro-movements* in the following. *PredicTouch* includes three IMUs to record micro-movements in three locations separately. This enables us to train models and to test different IMU locations as well as their combination for both finger input and stylus input. We analyze the most effective IMU location(s) and their practicality with an integration into common mobile devices (e.g. smartwatches or styli) in mind.

System

PredicTouch consists of three IMUs (MPU-9250[1]) connected via cable to an Arduino Micro. The Arduino retrieves the sensor values and forwards these to a Google Nexus 7 tablet. To capture the hand's micro-movements, we placed two IMUs on the user's hand – IMU_{wrist} on the wrist of the hand used for interaction and IMU_{finger} on the second segment of the index finger between the distal interphalangeal joint (DIP)

[1] http://www.datasheetspdf.com/datasheet/MPU-9250.html

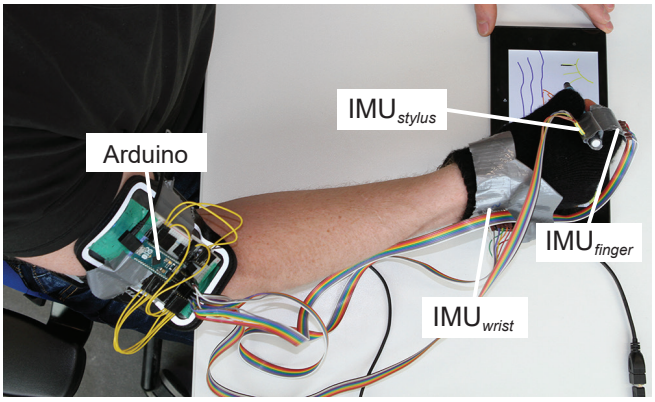


Figure 2. System overview: IMUs at a stylus, the index finger, and the hand wrist are connected to an Arduino board located at the upper arm.

and the proximal interphalangeal joint (PIP). We deliberately chose these locations as this makes it easier to integrate *PredicTouch* into off-the-shelf mobile devices such as smartwatches or smartrings[2]. Both IMUs are sewn onto a unisize glove and further fastened with adhesive tape which ensures a firm attachment onto the user's hand to avoid noise in the recordings. To also capture a stylus' micro-movements, we attached a IMU_{stylus} on the upper side of a passive capacitive stylus (Wacom Bamboo Stylus Solo, 9 *mm* tip). *PredicTouch* used by a participant is depicted in Figure 2.

Each IMU has 9 degrees of freedom, with three axes each for an accelerometer, gyroscope, and a magnetometer. The accelerometer samples with a frequency of $4\,kHz$, gyroscope with a frequency of $1\,kHz$ and the magnetometer with a frequency of $100\,Hz$. To minimize the sensor reading time and the time to transfer the sensor values to the tablet, we used the serial peripheral interface (SPI) on an Arduino Micro to access all three IMU sensors at once. Further, we send all 3×9 sensor values to the tablet via a serial connection using a USB OTG cable. We used the open-source Android USB serial driver library[3] to receive the sensor values in our Android application. Reading the sensor values takes $0.84\,ms$ on average while the average round trip time (reading and sending the sensor values to the tablet and awaiting the ACK) is $1.66\,ms$ ($SD = 0.79\,ms$). The touch prediction uses a neural network running on the Google Nexus 7 which has a 7.02 inch screen size with a resolution of $1920 \times 1200\,px$ and a sampling rate of $60\,Hz$.

Data Collection

To train artificial neural networks for predicting future touch positions, we designed three task types to collect representative touch input and the respective micro-movements performed by finger and stylus, based on previous work [11]. Participants performed these tasks wearing *PredicTouch* so that we recorded both the micro-movements and the input on the touchscreen.

Study Design & Tasks

The study implements a 3×2 within-subjects design with the independent variables task type (drawing, writing, and dragging) and the input method (finger and stylus). Using a balanced Latin square, we instructed participants to perform these tasks in a counterbalanced order. The tasks are depicted in Figure 3 and include: A *drawing task*, in which we instructed participants to draw a picture of their last vacation – we encouraged them to finish in less than ten minutes and proposed to draw a tropical island in case they did not come up with an own idea; a *writing task*, in which participants wrote phrases randomly selected from the phrase set by MacKenzie *et al.* [27] 10 minutes; and a *Fitts' Law based dragging task* that replicates the design by Jota *et al.* [16], which requires participants to drag a smaller square into a larger one. For this, we used three sizes for the smaller target ($24\,mm$, $32\,mm$, $40\,mm$) and three distances ($28\,mm$, $68\,mm$, $120\,mm$), resulting in nine combinations. These combinations were repeated eight times resulting in 72 instances. The study took about 60 minutes including breaks between each task. Participants wore the glove sitting in front of a table with the Nexus 7 tablet.

Participants

This study was conducted in a lab, at a technical university located in central Europe. We recruited 18 participants (9 female) from the university's campus. Participants were on average 26.8 years old ($SD = 11.6$) and were all right-handed with an average hand size of $17.5\,cm$ ($SD = 1.2\,cm$). No participant had any motor weakness or disease. All participants had prior experience using touchscreens for browsing or chatting. Further, six participants had prior experience with touch stylus. Participants were reimbursed for their participation with 10 €.

Data Set

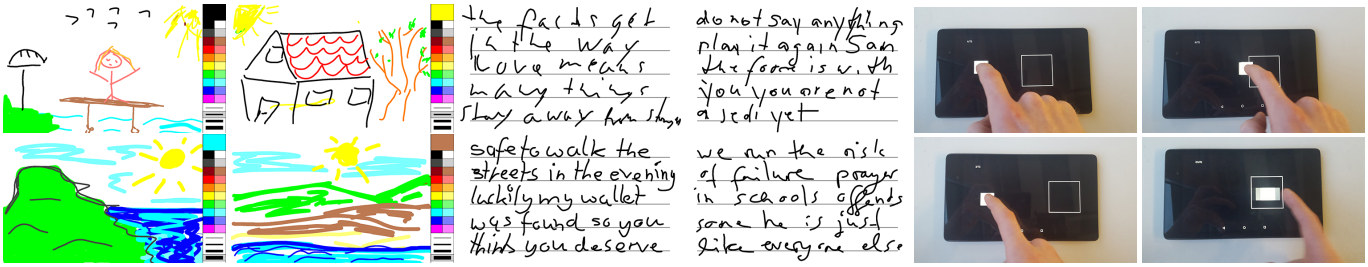
We collected $503,647$ touch samples performed with the finger, and $500,020$ touch samples with the stylus from the touchscreen. A touch sample consists of the timestamp, and the (x,y) position of the touch. Due to variances in the touchscreen's sampling rate (expected frequency is $60\,Hz$), we filtered out all the inadvertent touch samples with a temporal gap of more than $30\,ms$ ($33\,Hz$). Thus, 2.2% of the touch samples made with the finger and 1.3% made with the stylus were removed. After removing, the average frame rates are $17.0\,ms$ (SD=$2.1\,ms$) and $17.0\,ms$ (SD=$2.0\,ms$) for finger and stylus. For the IMUs, we collected 14,216,904 IMU samples while participants used their finger to perform input on the touchscreen and 14,119,369 while using the stylus. These samples consist of a timestamp and three (x,y,z) triples for the accelerometer, gyroscope, and magnetometer each. We used a nearest-neighbor approach to match the IMU sensor values with their respective touch samples based on their timestamps.

Training Artificial Neural Networks

Predicting the user's finger or stylus position in the near future is done by an artificial neural network which we trained using *TensorFlow*. To test and compare different IMU positions and combinations, we trained 4×2 neural networks whereas each uses the touch trajectory and a specific IMU configuration. For input using the finger, we tested all feasible combination of the IMU: touch trajectory only, IMU_{finger}, IMU_{wrist} and both $\text{IMU}_{finger} + \text{IMU}_{wrist}$. For stylus input, besides training on touch trajectory only, the IMU configurations are IMU_{stylus}, and its combination with IMU_{finger} and IMU_{wrist}. IMU_{stylus}

[2] http://nfcring.com/

[3] https://github.com/mik3y/usb-serial-for-android



(a) (b) (c)

Figure 3. (a) Examples of the pictures painted by the participants, (b) screenshots of the written text, and (c) photos of the Fitts' dragging task.

could not be omitted in any configuration as it is required to capture the micro-movements of the stylus.

For each point in time t, the neural networks use the touch trajectory consisting of a position and rotation invariant version of the previous 10 touch samples, and the 10 latest micro-movements of the hand measured by the IMUs at a time t based on the highest frequency of the three IMU sensors (accelerometer, $4\,kHz$). To compare different neural networks with each other, we used the Euclidean distance between the actual and the predicted point as the comparison metric. Results are shown in Figure 4.

Constructing the Input Vector

Each time an input vector is constructed, we take the 11 latest touch samples in time $t-10$ to t and their respective (x,y,z) triples for the accelerometer, gyroscope, and magnetometer each. Afterwards, the touch samples that represent a touch trajectory are made position and rotation invariant similar to Henze *et al.* [11]. Position invariance was yielded by calculating the difference for x and y between the 11 touch samples, resulting in 10 tuples representing directional vectors in x and y direction. The same process was performed on the values of the accelerometer, gyroscope, and magnetometer of the IMU samples. In a second step, we took the 10 vectors from the previous step and made them rotation invariant. This was done by rotating the 10 directional vectors counterclockwise so that the one farthest back in time is aligned with the y-axis. Thus, directional changes in the trajectory are observed from a fixed angle which makes them rotation invariant.

The input vector for the neural network consists of the 10 directional vectors that are position and rotation invariant. Additionally, for each IMU we added the 10 latest (x,y,z) triples of the accelerometer and gyroscope starting from time t. This gives the neural network information about the recent micro-movements of the hand or stylus. This results in 20 input values from the touchscreen, plus 60 further input values for each IMU that is used.

Neural Network Architecture

We trained a multi-layer feedforward neural network [36] for regression to predict where the user's finger or stylus will be in the near future. We tested different network configurations including variations of the amount of neurons and layers, activation functions, and optimizers provided by *TensorFlow*. The final network consists of an input layer with 20 neurons plus further 60 for each IMU used and is connected to three hidden layers comprising 500, 400 and 300 neurons. The output layer consists of 12 neurons whereas each pair represent the (x,y) position of a future touch in the subsequent time $t \in T$ with $T = \{16\,ms, 33\,ms, 49\,ms, 66\,ms, 72\,ms, 99\,ms\}$.

We used the adaptive gradient algorithm (AdaGrad [8]) as the optimizer and two rectified linear units (ReLU [29]) and one sigmoid as activation functions. To initialize the network weights and biases, we used the Xavier initialization scheme as proposed by Glorot *et al.* [9]. The cost function for optimization is the Euclidean distance between the points of the predicted trajectory p and the points of the actual trajectory a over all six subsequent time points t as shown in Equation (1).

$$cost = \sqrt{\frac{\sum_t^T (p_t - a_t)^2}{|T|}} \tag{1}$$

The neural network was trained until the cost function converges with a threshold of $\leq .001$. This was the case after 660.5 epochs on average (SD = 86.7).

Model Validation and Comparison

For each trained neural network, we implemented a leave-one-out cross-validation [20] over all 18 participants. With this, we evaluated the model's performance in the form of the Euclidean distance in px between the predicted point and the actual point. For all tasks and prediction times T, the neural networks that involve at least one IMU sensor yield a lower average Euclidean distance than their counterparts that use only the touch trajectory. For both finger input and stylus input, two three-way ANOVAs and Bonferroni-corrected pairwise comparisons revealed significant differences between all aforementioned conditions and interactions (with $p < .001$), except between the usage of IMU_{wrist} and $\text{IMU}_{wrist} + \text{IMU}_{stylus}$ when using the stylus ($p < .966$). To compare our results with the trajectory-only approach by Henze *et al.* [11], we report the results of three prediction levels ($33\,ms$, $66\,ms$ and $99\,ms$) which equals to predicting 2, 4, and 6 touch samples into the future when assuming a constant sample rate of $60\,Hz$. Figure 4 shows the Euclidean distances in px for all tasks, different IMU configurations and the three prediction times.

To report the Euclidean distance for all three prediction levels, we use a square bracket notation, starting with the $33\,ms$ level (i.e. $[33\ 66\ 99]\,ms$). For input with the stylus, the neural network that uses the touch input and IMU_{stylus} samples leads to the lowest Euclidean distance of all IMU configurations. This neural network achieved an average Euclidean distances of $[9.9\ 20.9\ 37.1]\,px$ (SD = $[15.7\ 30.4\ 41.1]\,px$). For finger input, the model using both IMU_{wrist} and IMU_{finger} with the

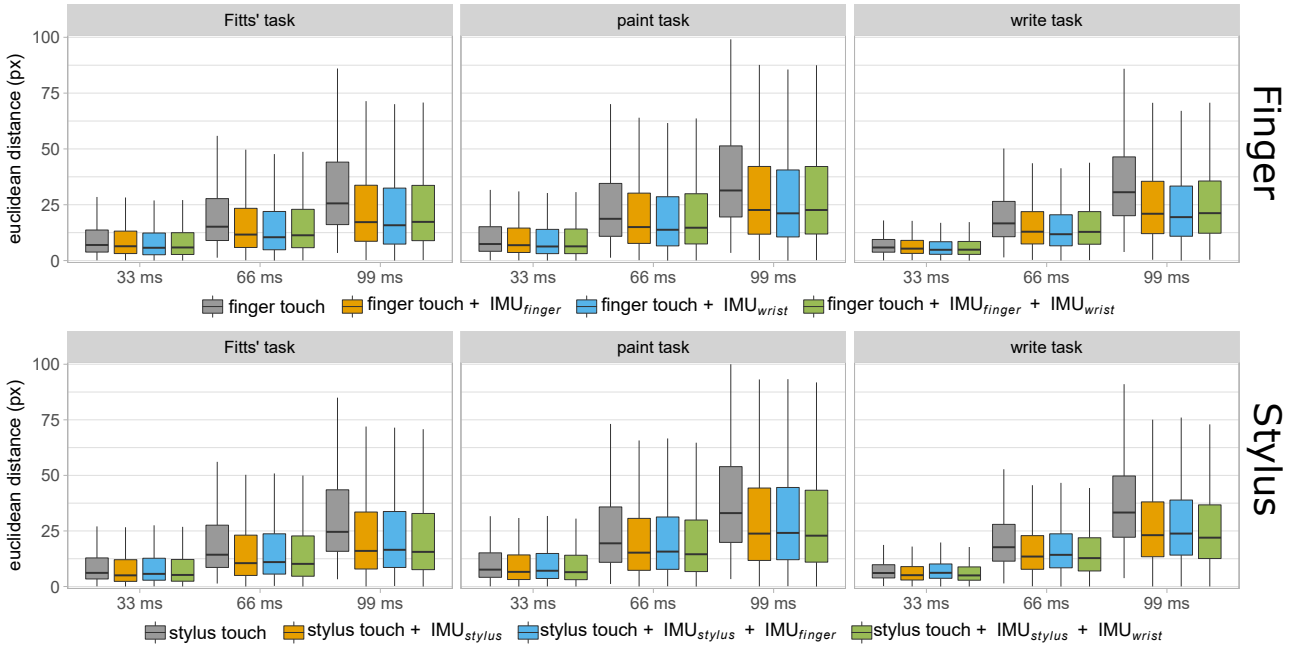


Figure 4. Euclidean distances (*px*) between the predicted and actual touch position for a 33 *ms*, 66 *ms*, and 99 *ms* prediction for finger and stylus input. 10 *px* equals to 0.79 *mm* and 10 *mm* equals to 127.20 *px*. Error bars show the 75% quantiles, boxes the 50% quantiles, the black line depicts the median.

touch input leads to the lowest distance of [9.4 20.6 34.0] *px* (*SD* = [14.9 28.5 48.9] *px*). When using only IMU$_{wrist}$ and touch for finger input, we achieved a distance of [10.0 22.0 35.9] *px* (*SD* = [15.1 28.9 44.4] *px*).

Additionally, we tested a combination of all three IMUs for stylus usage. This leads to a distance of [10.0 21.8 34.8] *px* (*SD* = [16.0 31.0 46.8] *px*) which is higher than using only IMU$_{stylus}$. We also tested using the magnetometer values of the IMU sensors and found they lead to higher distances than without. As an example, using magnetometer values with IMU$_{wrist}$ and IMU$_{finger}$ for finger input leads to distances of [10.0 23.0 38.2] *px* (*SD* = [15.1 29.4 45.0] *px*).

Discussion

We trained artificial neural networks for different combinations of IMU$_{wrist}$, IMU$_{finger}$, IMU$_{stylus}$ as well as touch trajectory only and compared them with each other using leave-one-out cross-validation. We showed that the error rate is significantly lower when involving IMUs in the prediction compared to using touch only. For finger input, using IMU$_{wrist}$ and IMU$_{finger}$ together with the touch trajectory on the display leads to the lowest error – closely followed by using only the IMU$_{wrist}$ and the touch trajectory. When using a stylus, touch trajectory and IMU$_{stylus}$ leads to an error that is significantly lower than other combinations for stylus input. We found that using magnetometer data lead to a higher prediction error. Since magnetometers are prone to interference through nearby ferromagnetic objects and electro-magnetic fields, they overfit the model and result in a lower prediction performance.

An IMU$_{wrist}$ and IMU$_{stylus}$ could be readily deployed into off-the-shelf products such as smartwatches and an IMU built into a stylus (e.g. NoteOn Smartpen[4]). To fully utilize the IMUs' sampling rate, we decided to use cables to connect the sensors and a serial connection to transfer the data to the tablet. The API of recent smartwatches limit the sampling rate of their IMUs to around 100 *Hz* which is sufficient for their main use cases. However, Laput *et al.* [23] have shown that it is possible to capture an LG G W100 smartwatch's accelerometer data at 4 *kHz* through OS kernel modifications. This would be a possible solution to integrate *PredicTouch* into the mobile devices that users already carry with them. IMU$_{finger}$ recently provides a challenge as this requires the user to wear a special ring to entail an IMU.

We conducted a lab study to collect a data set in a controlled setting leveraging the full technical capabilities of our system. In contrast, Henze *et al.* [11] developed an application which was freely available in the Google Play Store to collect a large data set. When comparing their prediction results for writing with our results when using only the touch trajectory, we found that our prediction error is higher. This could be due to participants performing touch input differently which makes it challenging to compare results across studies. When using our collected data, a comparison of the approach by Henze *et al.* [11] and our hybrid approach revealed that using IMUs always lead to a lower prediction error than using only the trajectory across all tasks.

To summarize, our work compares well with the closest software-only prediction models for painting tasks, is worse for a writing task, but excels at dragging tasks. Since *PredicTouch* works best for dragging and painting tasks, we focus on reducing latency for dragging and painting tasks in the remaining work. As the model for finger input, we use only IMU$_{wrist}$ (which had the second lowest error rate after IMU$_{wrist}$ + IMU$_{finger}$) with the touch trajectory as there is recently no smartring with an open API to use IMU$_{finger}$ to implement

[4] https://hackaday.io/project/2678-noteon-smartpen

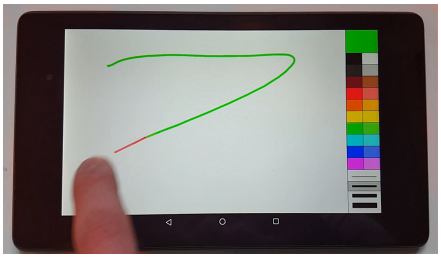


Figure 5. Image of 66 ms touch-only prediction in the painting task. The prediction is exemplarily highlighted by the red line.

PredicTouch in commercial hardware. Further, using IMU_{wrist} + IMU_{finger} requires users to carry both devices and would not work if one is missing. For stylus input, we simply used the IMU_{stylus} configuration as this lead to the lowest error rate in the model validation.

COMPARING PREDICTION EFFECTS

While we showed that IMUs can significantly reduce the objective error, the results do not show how this affects the users' performance. Thus, we conducted a study to evaluate the effects on the user.

Study Design

To be in line with Henze *et al.* [11], we used the same levels of *prediction* to reduce latency: 0 *ms* (baseline), 33 *ms*, and 66 *ms*. As previously mentioned, the average error (for stylus and finger) was up to 37.1 *px* ($SD = 41.1$) using the 99 *ms* prediction. Thus, the resulting jitter of this prediction level would be impractical and unreasonable high to be tested in a user study. The three *prediction* levels were tested with an *IMU* (or not) per *device* (finger touch input with IMU_{wrist}, and stylus with IMU_{stylus}), which lead to 12 tasks per participant. Since an *IMU* could only be tested at the 33 *ms* and 66 *ms* *prediction* levels, the factor *prediction* is nested into the factor *IMU*. Nested designs are proposed by Krzywinski *et al.* [21] to understand the variability in the hierarchy of subsamples. In our case, they prevent cross comparisons between the *IMU* and the three *prediction* levels as well as cross comparisons between the *prediction* levels and the *devices*. However, they allow to include the usage of IMUs with the 0 *ms* prediction condition using the finger as well as the stylus. Thus, our experimental design finally results in 10 different conditions.

In line with Henze *et al.* [11], we used a Fitts' Law dragging task and a painting task. As in the data collection study, the Fitts' Law dragging task includes three target sizes, three distances, and eight repetitions, which results in 72 tasks per condition. The order of the tasks was randomized. Target sizes and distances were the same as in the data collection study. We measured errors and task completion time (TCT) to determine the throughput of each participant. The higher the throughput, the more accurate and faster users can perform input. In the painting task, participants were asked to draw a picture from their last holidays. Participants that did not come up with an idea were asked to paint a picture of tropical islands. We used a 7-point Likert items to ask participants for the perceived jitter, lag and unpleasantness after each condition (7 indicates a high perception). Conditions and tasks were pseudo-randomized using a pre-defined block plan to avoid sequence effects.

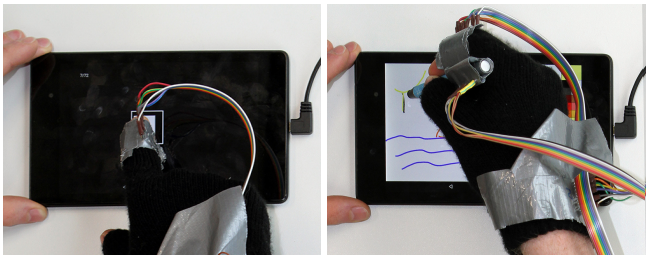


Figure 6. Participant using PredicTouch and conducting a Fitts' task with the finger (left) and a painting task with the stylus (right).

Apparatus

Using *TensorFlow Mobile*[5], we integrated the neural networks into the system used in the data collection study (see Figure 6). The model that only used the touch trajectory as input require on average 2.565 *ms* ($SD = 0.859$ with $N = 1,000$ samples) to execute. Models that used touch trajectory and IMUs require on average 3.935 *ms* ($SD = 0.944$ with $N = 1,000$ samples) to execute. In the Fitts' Task, predicted positions were directly applied to the square while in the painting task, the prediction is visualized by appending an additional line segment at the tip of the regular line (Figure 5 shows the exemplary additional line segment in red. While the appended line segment reduces the gap between the user's moving finger and the line, only the touch positions provided by Android were kept on the canvas.

Participants and Procedure

We recruited 16 right-handed participants (8 male, 8 female) from our university with a mean age of 28.8 years ($SD = 11.9$). No the participants suffered from motor weakness or disease, eight participants already took part in the first study. Participants wore the glove sitting in front of a table with the Nexus 7 tablet. Every participant received a compensation of 10€. All participants reportedly use touchscreens daily. Styli were used daily by 1 participant, 11 used a stylus weekly, 3 daily, 1 monthy, and 1 participant mentioned never to use a stylus.

After signing a consent form, the participants were briefed about the experimental procedure. After completing a task, participants were asked to rate the perceived jitter, lag, and unpleasantness of their input. We collected qualitative feedback to understand whether and how people perceive lag on their device and for which purposes they would like to reduce lag. The following three questions were asked in a semi-structured interview: (1) Have you ever felt latencies on one of your touch devices during touch movements? (1.a) If so, has that disturbed you? (2) Can you imagine touch tasks with a smaller latency? (2.a) If so, what would be the benefits for you? (2.b) Which tasks could benefit from using such a system?

RESULTS

We analyzed the effects of the independent variables DEVICE (*finger*, *stylus*), *IMU* (*yes*, *no*), and PREDICTION (0 *ms*, 33 *ms*, 66 *ms*) on the average throughput per participant, the task completion time (TCT), the error rate, as well as the subjective ratings. We used Bonferroni-corrected t-tests for pairwise comparisons when post-hoc tests were conducted.

[5] https://www.tensorflow.org/mobile/

device	IMU	prediction	throughput (bit/s) M	throughput (bit/s) SD	TCT (ms) M	TCT (ms) SD	error (%) M	error (%) SD
finger	touch only	0 ms	2.643	1.046	678	207	2.5	2.1
		33 ms	2.788	1.052	635	196	2.3	3.0
		66 ms	2.821	1.338	662	260	5.1	5.0
	touch + IMU	33 ms	2.889	1.240	636	219	1.6	1.6
		66 ms	3.051	1.397	611	225	2.4	2.7
stylus	touch only	0 ms	2.452	0.939	721	197	3.0	2.2
		33 ms	2.719	1.033	654	208	2.0	3.0
		66 ms	2.800	1.184	644	218	4.2	4.0
	touch + IMU	33 ms	2.781	1.088	647	215	2.3	2.5
		66 ms	2.870	1.114	615	200	2.8	1.7

Table 1. Average throughput (bit/s), task completion time (TCT in ms), and error rate (in %) of the participants during the Fitts' task.

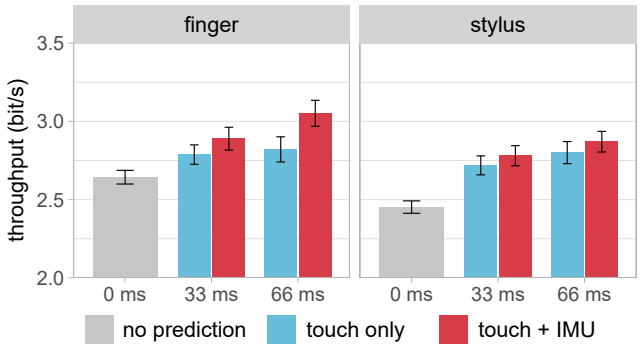


Figure 7. Average throughput in the Fitts' task for finger and stylus. Error bars show 95% confidence interval.

Throughput

The results of the Fitts' throughput measures are depicted in Figure 7. Outliers in the raw data were filtered through keeping the data within the 1% and 99% quantiles. Only completed tasks were included in the analysis of the throughput. The throughput was calculated with respect to target size and target distance as suggested by MacKenzie [26]. The results of the Fitts' task are summarized in Table 1.

We conducted a three-way repeated measures (RM) ANOVA and found a significant effect of IMU [$F(1,15) = 6.329, p = .023$] and a significant effect between the levels of PREDICTION [$F(2,90) = 2.233, p = .047$] on the average throughput. Considering the *stylus* condition *with IMU*, pairwise comparisons revealed significant differences between the $0\,ms$ and $66\,ms$ ($p = .012$) as well as between the $33\,ms$ and $66\,ms$ ($p = .050$). Considering *no IMU* and the *stylus* conditions pairwise comparisons between the prediction levels showed a significance difference between $0\,ms$ and $66\,ms$ ($p = .014$).

Using *an IMU* in the *finger* condition, pairwise comparisons between the prediction levels showed a significance difference between $0\,ms$ and $33\,ms$ ($p = .022$) and between $0\,ms$ and $66\,ms$ ($p = .017$). Using *no IMU* in the *finger* condition, pairwise comparisons between the prediction levels showed a significance difference between $0\,ms$ and $33\,ms$ ($p = .015$) and between $33\,ms$ and $66\,ms$ ($p = .022$). No significant effects were found between both DEVICES [$F(1,15) = 1.485, p = .242$]. Since prediction time is a nested factor of the IMU condition, interactions could only be compared between the levels of DEVICE × IMU [$F(1,15) = 1.518, p = .237$], which was not significant. The results of the average throughput per participant are depicted in Figure 7.

TCT and Error Rate

We found significant effects of IMU [$F(1,15) = 12.24, p = .003$] and PREDICTION [$F(2,90) = 9.411, p < .001$] on the average TCT of the participants. A significant effect was found for DEVICE [$F(1,15) = 1.018, p = .329$], however, there was no interaction effect of DEVICE × IMU [$F(1,15) = .573, p = .461$]. Pairwise comparisons showed the same of levels of significant results as the previously reported throughput. There were neither a significant effects of IMU [$F(1,15) = 1.232, p = .284$], DEVICE [$F(1,15) = 0.024, p = .087$], PREDICTION [$F(2,90) = 1.301, p = .133$], nor an interaction of DEVICE × IMU [$F(1,15) = .144, p = .389$] on the average error rate per participant.

We summarize that the objective measures showed an increased throughput and a reduced TCT while using an IMU. There was no interaction effect, which means that *finger* as well as *stylus* benefit from the factor IMU. The independent variables showed no effects on the error rate.

Subjective Ratings

We analyzed the 7-point ($1 = low$, $7 = high$) ratings for the perceived jitter, lag, and unpleasantness of the Fitts' and paint task. All results are depicted in Figure 8. The three ratings were given on a 7-point Likert scale and, thus, non-parametrical data. To evaluate non-parametrical data in a multi-factorial analysis we used aligned rank transformations (ART) introduced by Wobbrock *et al.* [40][6].

For the perceived jitter ratings of the Fitts' task we found a significant effect of DEVICE [$F(1,15) = 20.390, p < .001$], IMU [$F(1,15) = 51.759, p < .001$], and PREDICTION [$F(2,90) = 245.966, p < .001$] as well as an interaction effect of DEVICE × IMU [$F(1,15) = 4.956, p = .041$]. For the perceived lag, we found no significant effect of DEVICE [$F(1,15) = 0.067, p = .798$], however, of IMU [$F(1,15) = 11.118, p = .004$] and PREDICTION [$F(2,90) = 31.435, p < .001$]. There was no interaction effect of DEVICE × IMU [$F(1,15) = 0.156, p = .699$]. For the perceived unpleasantness, we found no significant effect of DEVICE [$F(1,15) = 3.444, p = .0831$] or IMU [$F(1,15) = .605, p = .448$] , however, of PREDICTION [$F(2,90) = 33.193, p < .001$]. No interaction interaction effect of DEVICE × IMU [$F(1,15) = 2.832, p = .113$] was found for the perceived unpleasantness in the Fitts' task.

Analyzing the subjective ratings for the paint task we found a significant effect of PREDICTION [$F(2,90) = 96.721, p < .001$], DEVICE [$F(1,15) = 44.658, p < .001$], IMU [$F(1,15) = 9.382, p = .007$], however, no interaction effect of DEVICE × IMU [$F(2,90) = .976, p = .339$] on perceived jitter. For the perceived lag during the paint task we found no significant effect of DEVICE [$F(1,15) = 0.267, p = .612$], however, of IMU [$F(1,15) = 11.118, p = .004$], and PREDICTION [$F(2,90) = 31.435, p < .001$], without an interaction of DEVICE × IMU [$F(1,15) = 0.074, p = .789$].

[6] https://depts.washington.edu/aimgroup/proj/art/

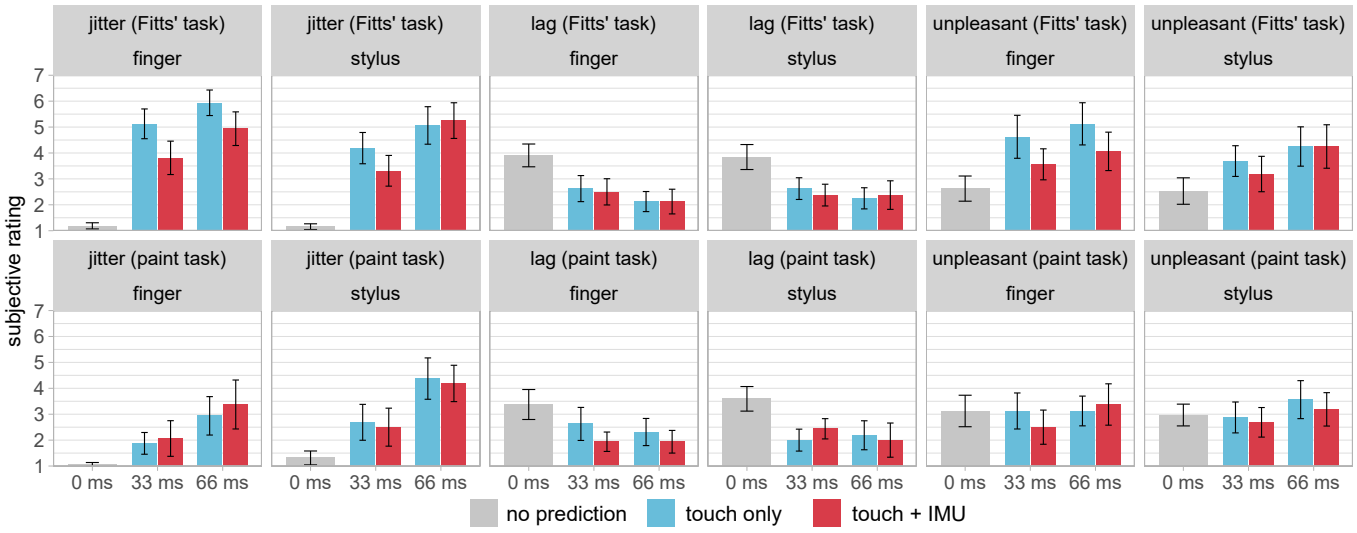


Figure 8. Mean subjective ratings for the Fitts' and paint task using the finger or the stylus (1 = low, 7 = high). Error bars show 95% confidence interval.

Qualitative Feedback

Finally, we analyzed the transcribed comments provided by the participants after completing both tasks. Two researchers went through all comments and annotated the key concepts to identify how participants perceived latencies. The majority of our participants ($N = 12$) already noticed lag of mobile devices' touchscreens. Only one participant mentioned having never paid attention to that. They also pointed out, that lower latencies lead to less frustration (4), higher speed (8), a better accuracy/precision (5), more concentration (2) and more continuity while working (1).

Participants mentioned that they previously noticed latency on their mobile device while scrolling (7), gaming (4), writing/swiping (4), and painting (1). When participants noticed latencies, they all felt disturbed (12). Participants mentioned that the following applications on touchscreens may profit by using lower latencies: drawing (8), gaming (7), writing (5), scrolling (4), typing/swiping (4), panning and zooming (2), as well as while dragging elements or icons (1). We summarize that latency negatively affects performance and speed, the users' workflow, the mental workload, and satisfaction. Thus, reactive applications and application where precise inputs are desired would profit by reducing latency.

Discussion

We analyzed the effects of prediction to reduce software latency on touchscreens with an additional IMU at the wrist or integrated into a stylus. We collected objective and subjective measures in a Fitts' dragging task as well as in a painting task to investigate the users' performance and experience.

We used the data received from external IMUs to reduce touchscreens' latency. We showed that the prediction significantly increases the user's throughput in a Fitts' task for the 33 *ms* and 66 *ms* prediction with up to 15% for finger input and 17% for stylus input. Using an IMU significantly increases the user's throughput compared to using the touch trajectory only without having negative effects on the error rate. This was the case for both devices (finger and stylus) which means that using IMUs sensors as additional input for the neural networks are practical and useful. Although we encouraged our participants to speed up if the error rate was below 5% similar to Henze *et al.* [11], they reported higher throughputs for the Fitts' tasks. The average throughput reported *e.g.* by Jota *et al.* [16] are in line with our results.

The subjective results show that the IMU and the prediction, as well as the usage of an IMU, can significantly reduce the perceived lag. Differences of the perceived lag are larger while using the stylus than compared to using the finger, which indicates that especially the stylus benefits from the additional IMU. This is supported by a larger difference of throughput in the Fitts' tasks, where the IMU lead to a higher increase of the first prediction level (33*ms*) in the stylus than in the finger condition. Jitter, however, was perceived less pleasant. The subjective ratings show, that the perceived jitter depends on the task. The ratings for the perceived jitter in the painting task were significantly lower than in the Fitts' task, which leads to lower unpleasantness ratings for the painting task.

Qualitative feedback at the end of our user study gives useful insights into the perception of latencies and to understand which applications are affected and potentially profit by reducing them. Our analysis shows that participants felt frustrated with a lacking coherence between the actual finger or pen position and the touchscreen display. Applications would benefit from reducing latencies when they are either highly reactive (*e.g.* gaming, painting), when the input should be very precise (*e.g.* selecting, typing) or both (*e.g.* writing, drawing).

APPLICATION AREAS AND IMPLICATIONS

We proposed *PredicTouch*, a combination of a software-based approach and IMUs for predicting touch trajectories to reduce latency. We developed models to predict the touch trajectory based on the previous touch trajectory and micro-movements measured by IMUs. Our evaluation shows that a prediction of 66*ms* leads to a higher throughput without significant effects on the error rate. However, the model validation results (see Figure 4) and the participants' subjective ratings show

that the prediction of 33 *ms* leads to a lower unpleasantness rating without significant effects on the error rate. This can be explained by the fact that latency reduces the users' performance [16] and has a stronger effect on performance than the low amounts of jitter [33]. We showed that users perceive less lag while yielding a significantly higher average throughput on the cost of an increase in jitter. Users perceived more jitter during the Fitts' Task as the size of the dragged object is larger than the size of a finger. This suggests that applications with small, concealable or no cursors are especially suited for our approach as they are less affected by jitter.

Mobile games offer many possibilities to compensate jitter since steadily visible cursors are rarely used. Fruit Ninja[7] is a fast-paced mobile game that visualizes the users' input (i.e. slash to cut the fruit) through animated trajectories. As this makes it difficult for users to see the jitter, such games benefit from the latency reduction with minimal side effects. Our approach can also be used to reduce latency while performing gestures. With gesture sets in reasonable sizes, gesture recognizers such as the $P, $1 or $N should still be able to distinguish between given gestures based on a nearest-neighbor classification [37]. Similarly, gesture keyboards such as *ShapeWriter* [41] or Google's GBoard can benefit from the prediction to reduce latency. Moreover, applications can predict which UI components will be touched next to preload functions similar to web browsers to increase responsiveness. Moreover, UI components could e.g. be enlarged when the user is about to touch them. This could avoid touching at a wrong position due to errors or the fat-finger problem [35].

LIMITATIONS

Both studies were conducted with students from a technical university in central Europe who belong to a similar group in terms of experiences. Thus, results may be different when conducted in countries with either less or more touchscreen and stylus usage. Further, we used a set of specific tasks (i.e. Fitts' dragging task, painting, and writing) that may differ from real world use cases. Participants were all seated during the study while performing the tasks which may lead to less IMU noise than while walking or being in a moving vehicle.

CONCLUSION AND FUTURE WORK

In this work, we presented *PredicTouch*, a system to reduce latency on touchscreens by extrapolating the finger's or stylus' position from the previous movement. We built on previous work by Henze *et al.* [11] who showed that software-reduced touchscreen latency can significantly increase users' performance. They trained an artificial neural network using touch trajectories collected from a painting app to predict the finger position in the near future. We extended this approach and involved inertial measurement units (IMUs) to provide the neural network with the hand's and stylus' micro-movements in addition to the touch trajectories. This significantly decreases the prediction error, and increases users' throughput by 15% and 17% for finger and stylus input respectively when predicting 66 *ms* into the future. With smartwatches, digital pens, and wearable technologies becoming more ubiquitous, our system could be readily deployed into commodity devices to reduce latency at the cost of a low amount of spatial jitter.

In line with previous work, we used standard multi-layer feedforward neural networks to show the positive effect of IMUs. More specialized models, recently proposed to improve software-reduced touchscreen latency [12], such as RNNs, LSTMs, or hidden Markov models as well as techniques used in machine learning competitions such as ensembles, bagging or boosting might improve the prediction performance. We expect that the prediction can also be improved by feeding the machine learning models with rich capacitive data from around the device [24]. These approaches can be employed in future work to improve touch prediction in general. As data can be collected while the mobile device is used, the model could continuously be improved similarly to mobile keyboards improving their word predictions. With such a self-improving system, we envision a framework that enables people to use IMUs integrated into different devices or smart clothes to reduce touchscreen latency without carrying additional hardware.

Acknowledgements: This work is supported by the DFG within the SimTech Cluster of Excellence (EXC 310/2), through project C04 of SFB/Transregio 161, and the MWK Baden-Württemberg within the Juniorprofessuren-Programm.

REFERENCES

1. Annett, M., Ng, A., Dietz, P., Bischof, W. F., and Gupta, A. How low should we go?: Understanding the perception of latency while inking. In *Proc. GI* (2014).
2. Bérard, F., and Blanch, R. Two touch system latency estimators: High accuracy and low overhead. In *Proc. ITS* (2013).
3. Cattan, E., Rochet-Capellan, A., Perrier, P., and Bérard, F. Reducing latency with a continuous prediction: Effects on users' performance in direct-touch target acquisitions. In *Proc. ITS* (2015).
4. Chowdhury, D., Banerjee, S. J., Sanyal, K., and Chattopadhyay, M. A real time gesture recognition with wrist mounted accelerometer. In *Information Systems Design and Intelligent Applications*. 2015.
5. Deber, J., Araujo, B., Jota, R., Forlines, C., Leigh, D., Sanders, S., and Wigdor, D. Hammer time!: A low-cost, high precision, high accuracy tool to measure the latency of touchscreen devices. In *Proc. CHI* (2016).
6. Deber, J., Jota, R., Forlines, C., and Wigdor, D. How much faster is fast enough?: User perception of latency & latency improvements in direct and indirect touch. In *Proc. CHI* (2015).
7. Deselaers, T., Keysers, D., Hosang, J., and Rowley, H. A. Gyropen: Gyroscopes for pen-input with mobile phones. *IEEE Transactions on Human-Machine Systems 45*, 2 (2015).
8. Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res. 12* (2011).

[7] https://fruitninja.com/

9. Glorot, X., and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proc. AISTATS'10*, vol. 9 (2010).
10. He, B., Bai, J., Zipunnikov, V. V., Koster, A., Caserotti, P., Lange-Maia, B., Glynn, N. W., Harris, T. B., and Crainiceanu, C. M. Predicting human movement with multiple accelerometers using movelets. *Medicine and science in sports and exercise 46*, 9 (2014).
11. Henze, N., Funk, M., and Shirazi, A. S. Software-reduced touchscreen latency. In *Proc. MobileHCI* (2016).
12. Henze, N., Le, H. V., Mayer, S., and Schwind, V. Improving software-reduced touchscreen latency. In *Proc. MobileHCI-EA* (2017).
13. Henze, N., and Poppinga, B. Measuring latency of touch and tactile feedback in touchscreen interaction using a mobile game. In *Proc. IWRL* (2012).
14. Hrabia, C.-E., Wolf, K., and Wilhelm, M. Whole hand modeling using 8 wearable sensors: Biomechanics for hand pose prediction. In *Proc. AH* (2013).
15. Hsu, Y.-L., Chu, C.-L., Tsai, Y.-J., and Wang, J.-S. An inertial pen with dynamic time warping recognizer for handwriting and gesture recognition. *IEEE Sensors Journal 15*, 1 (2015).
16. Jota, R., Ng, A., Dietz, P., and Wigdor, D. How fast is fast enough?: A study of the effects of latency in direct-touch pointing tasks. In *Proc. CHI* (2013).
17. Kaaresoja, T., Anttila, E., and Hoggan, E. The effect of tactile feedback latency in touchscreen interaction. In *Proc. WHC* (2011).
18. Kaaresoja, T., and Brewster, S. Feedback is... late: Measuring multimodal delays in mobile device touchscreen interaction. In *Proc. ICMI-MLMI* (2010).
19. Kaaresoja, T., Hoggan, E., and Anttila, E. Playing with tactile feedback latency in touchscreen interaction: Two approaches. In *Proc. INTERACT*, P. Campos, J. Graham, Nicholasand Jorge, N. Nunes, P. Palanque, and M. Winckler, Eds. (2011).
20. Kearns, M., and Ron, D. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. *Neural computation 11*, 6 (1999).
21. Krzywinski, M., Altman, N., and Blainey, P. Points of significance: Nested designs. *Nature Methods 11*, 10 (2014).
22. Kusano, T., and Komuro, T. 3d tabletop user interface with high synchronization accuracy using a high-speed stereo camera. In *Proc. ITS* (2015).
23. Laput, G., Xiao, R., and Harrison, C. Viband: High-fidelity bio-acoustic sensing using commodity smartwatch accelerometers. In *Proc. UIST* (2016).
24. Le, H. V., Mayer, S., Bader, P., and Henze, N. A smartphone prototype for touch interaction on the whole device surface. In *Proc. MobileHCI-EA* (2017).
25. Leigh, D., Forlines, C., Jota, R., Sanders, S., and Wigdor, D. High rate, low-latency multi-touch sensing with simultaneous orthogonal multiplexing. In *Proc. UIST* (2014).
26. MacKenzie, I. S. Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction 7*, 1 (1992).
27. MacKenzie, I. S., and Soukoreff, R. W. Phrase sets for evaluating text entry techniques. In *Proc. CHI-EA* (2003).
28. Mannini, A., Intille, S. S., Rosenberger, M., Sabatini, A. M., and Haskell, W. Activity recognition using a single accelerometer placed at the wrist or ankle. *Medicine and science in sports and exercise 45*, 11 (2013).
29. Nair, V., and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proc. ICML* (2010).
30. Ng, A., Annett, M., Dietz, P., Gupta, A., and Bischof, W. F. In the blink of an eye: Investigating latency perception during stylus interaction. In *Proc. CHI* (2014).
31. Ng, A., and Dietz, P. H. The effects of latency and motion blur on touch screen user experience. *Journal of the Society for Information Display 22*, 9 (2014).
32. Ng, A., Lepinski, J., Wigdor, D., Sanders, S., and Dietz, P. Designing for low-latency direct-touch input. In *Proc. UIST* (2012).
33. Pavlovych, A., and Stuerzlinger, W. The tradeoff between spatial jitter and latency in pointing tasks. In *Proc. EICS* (2009).
34. Ritter, W., Kempter, G., and Werner, T. User-acceptance of latency in touch interactions. In *Proc. UAHCI*, M. Antona and C. Stephanidis, Eds. (2015).
35. Siek, K. A., Rogers, Y., and Connelly, K. H. Fat finger worries: How older and younger users physically interact with pdas. In *Proc. INTERACT* (2005).
36. Tahmasebi, P., and Hezarkhani, A. Application of a modular feedforward neural network for grade estimation. *Natural resources research 20*, 1 (2011).
37. Vatavu, R.-D., Anthony, L., and Wobbrock, J. O. Gestures as point clouds: A $p recognizer for user interface prototypes. In *Proc. ICMI*, ICMI '12 (2012).
38. Wang, J.-S., Hsu, Y.-L., and Liu, J.-N. An inertial-measurement-unit-based pen with a trajectory reconstruction algorithm and its applications. *IEEE Transactions on Industrial Electronics 57*, 10 (2010).
39. Wilkinson, G., Kharrufa, A., Hook, J., Pursglove, B., Wood, G., Haeuser, H., Hammerla, N. Y., Hodges, S., and Olivier, P. Expressy: Using a wrist-worn inertial measurement unit to add expressiveness to touch-based interactions. In *Proc. CHI* (2016).
40. Wobbrock, J. O., Findlater, L., Gergle, D., and Higgins, J. J. The aligned rank transform for nonparametric factorial analyses using only anova procedures. In *Proc. CHI* (2011).
41. Zhai, S., Kristensson, P. O., Gong, P., Greiner, M., Peng, S. A., Liu, L. M., and Dunnigan, A. Shapewriter on the iphone: From the laboratory to the real world. In *Proc. CHI-EA* (2009).