# Software-Reduced Touchscreen Latency

**Niels Henze[+], Markus Funk[+], Alireza Sahami Shirazi[*]**
[+]University of Stuttgart [*]Yahoo Inc.
[+]{firstname.lastname}@vis.uni-stuttgart.de, [*]alireza@yahoo-inc.com

## ABSTRACT
Devices with touchscreens have an inherent latency. When a user's finger drags an object across the screen the object follows with a latency of around 100ms for current devices. Previous work showed that latencies down to 25ms reduce users' performance and that even 10ms latency is noticeable. In this paper we demonstrate an approach that reduces latency using a predictive model. Extrapolating the finger's movement we predict where the finger will be in the next moment. Comparing different prediction approaches we show for three different tasks that prediction using neural networks is more precise than linear and polynomial extrapolation. Furthermore, we show through a Fitts' Law dragging experiment that reducing touch latency can significantly increases users' performance. As the approach is software-based it can easily be integrated into existing mobile applications and systems.

### Author Keywords
Touchscreen; Touch Input; Latency; Lag; Prediction

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

## INTRODUCTION & RELATED WORK
Latency denotes the time between a stimulus and a resulting response. Just as other input and output technologies, touch screens have certain latency. As described by MacKenzie and Ware, lag is inevitable and can be attributed to properties of input devices, software, and output devices [19, 22]. Kaaresoja and Brewster measured the latency of commercial smartphones [15]. They found that the latency strongly varies across software, hardware, and modalities. For visual feedback, the lowest latency they reported is around 75ms. But latency can go up to more than 200ms. Similarly, Ng et al. stated that visual latency of modern touch systems is between 50ms and 200ms [22]. This means that if an object is dragged on a touch screen, the object follows the finger with 50ms to 200ms. This latency results in a gap between finger and object that can be several centimeters.

[*]The work has been conducted while he was at the University of Stuttgart.


*MobileHCI '16*, September 06-09, 2016, Florence, Italy

DOI: http://dx.doi.org/10.1145/2935334.2935381

The effect of latency on perception and performance has been studied for different domains and modalities. Latency, for example, significantly affects user performance in VR [8], especially for free-hand pointing [31]. For AR, Allision et al. found that latency reduces the user's stability [2]. It is reported that latencies of 50ms and 100ms affect the ability of users to visually follow a virtual object in a head-mounted display setup [21]. Meehan et al. compared end-to-end latencies of 50ms and 90ms in VR. They showed that higher latency even results in a lower sense of presence and weaker physiological responses [20].

Previous work investigated the latency for different input devices including mouse and touchscreen [22, 24, 25, 27]. Anderson et al. showed that systems with higher latency can feel less responsive [3]. Ng et al. proposed the Accelerated Touch System, a prototype that combines a traditional direct-touch layer with a low-latency layer that displays nearly immediate visual feedback [22]. Jota et al. used this system to investigate the effect of different levels of latencies down to 1ms [14]. They found that users are able to discriminate differences far below the latency of current consumer devices [22]. Using a Fitts' Law dragging task it is further shown that latency down to 25ms decreases performance.

An approach to reduce latency is predicting the user's action. Ababsa et al., for example, compared different approaches to compensate head tracking latency for augmented reality [1]. Buker et al. investigated the effect of latency on simulator sickness while using a see-through helmet-mounted display [6]. They showed that predictive compensation results in a lower magnitude of simulator sickness. Xia et al. reduced touchdown latency for touchscreens by predicting when a finger touches the screen by observing the finger above the screen [32]. More related to our work is endpoint prediction for selection tasks. For example, Lank et al. propose a method for predicting gesture endpoints using the motion kinematics [17]. Similarly, Ziber et al. used statistical machine learning [33] and Pasqual and Wobbrock used template matching [23] for endpoint prediction for mouse pointing. Recent work by Cattan et al. also investigated the effect of linearly extrapolating the position of a user's finger on a tabletop to reduce latency [7]. Using a custom system optimized for low latency, the authors show that the simple linear extrapolation did not improve participants' performance for end-to-end latencies above 42ms.

We propose a software-based prediction of finger movement to reduce the latency of mobile touchscreen. Inspired by work in VR [6], we predict the future position of the user's finger

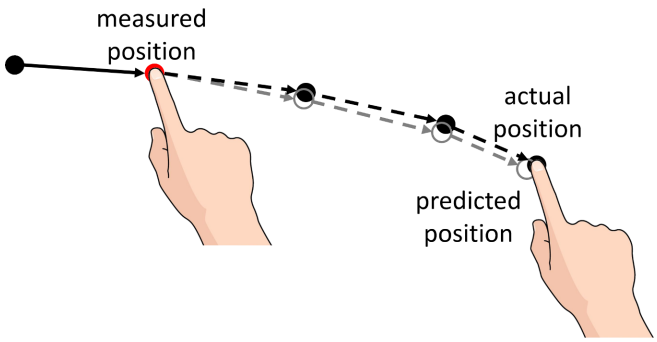



**Figure 1. By processing the collected data offline a predicted touch position can be compared with the actual position where the finger will be for every touch point. We use $error$ as a combined measure of $lag$ and $jitter$ for the comparison of different prediction techniques.**

on the touch screen. In contrast to previous work on endpoint prediction, e.g. [17, 23, 33], this requires not only to determine one out of a limited number of endpoints but to reconstruct arbitrary trajectories. Similar to Cattan et al. [7] we predict a finger's position using the previous trajectory. In contrast to Cattan et al. we use neural networks trained with data collected from actual touch strokes. We show that an ensemble of neural networks significantly reduces the prediction error compared to the linear extrapolation used by Cattan et al. We further show through a Fitts' Law experiment that predicting a finger's movement using neural networks on a standard mobile device significantly improves users dragging performance.

## PREDICTING TOUCH POSITIONS

As shown in Figure 1, our aim is to reduce the end-to-end latency of mobile touch devices by predicting the finger's movement. For touchscreens, the end-to-end latency is the time between a user's action on the screen and the time a visual reaction is displayed. End-to-end touchscreen latency is between 50ms and 150ms for common mobile devices. Thus, when a user uses a finger to drag an object across the screen the dragged object follows the finger with a delay of up to 150ms. Depending on the speed of the finger the latency can result in a gap of several centimeters between the finger and the dragged object. Similarly, when a user draws a line there is a gap between the line that is drawn and the finger.

A potential approach for reducing touchscreen latency is to predict the position of the user's finger based on the current observation. Perfect prediction of the finger's position in 100ms could eliminate the end-to-end latency of a touchscreen with 100ms. To predict where the user's finger will be in the near future, we trained neural networks to predict the position of the user's finger. We first implemented and deployed a drawing and writing application for Android devices. We used the application to collect data from 6181 mobile devices. Using a subset of the data, we trained an ensemble of neural networks that predict the movement of the user's finger on the screen.

### Application for data collection

We implemented a drawing and writing application to collect a large number of touch strokes from a diverse sample and different devices [12, 13]. The application[1] provides a set of features that are typical for simple drawing and writing applications available for mobile devices (see Figure 2). The application displays a modal dialog at the first start that informs the user that the application is a research apparatus and asks if the user is willing to share stroke data with. We use the approach with two-buttons discussed by Pielot et al. [28]. The application provides two different views. In the applications gallery, users can create new pages or select pages to continue drawing on existing ones. Furthermore, users can delete pages, export them to the device's gallery and share them through the Android standard sharing mechanism.

In the page-view, users can use a number of tools to draw and write on a page. They can choose a color from a color picker (2 left). Nine different backgrounds are provided and users can also use an image from their gallery as background (2 center). Furthermore, six different pens and one eraser are provided (2 right). The six pens mimic different drawing and writing tools including a felt pen, a calligraphic pen, and a marker pen. Users can undo and redo strokes using the two arrows in the icon bar.

The Android system provides the application with the position of the user's finger or stylus while the user draws on the touchscreen. Touch events are delivered with a frequency of up to 60Hz. To draw smooth strokes, between two touch events, the intermediate positions are interpolated using polynomial interpolation. All touch events are stored within the application to enable undo and to enable the collection of touch strokes on our server. To prevent that Android's garbage collector starts while the user draws a stroke, strokes are stored in a pre-allocated memory. Whenever the finger or stylus lifts of from the touchscreen, the collected data is copied to newly allocated memory and the garbage collector is started.

In case the user is willing to share stroke data for research purposes we transmit the collected data whenever the user leaves the page view, either by going back to the gallery or by exiting the application. Stroke data is combined with information about the user's device and transmitted to our server via HTTPS to preserve users' privacy. On the server, the data from each individual device is stored separately.

### Collected data

We published the first version of the application on Google Play on January 25th, 2015. The first version, however, did not transmit data to our server as the languorous authors had to implement the logging facilities. On August 8th, 2015 we submitted an update of the application that records touch strokes and transmits them to our server. Between August 8th, 2015 and December 15th, 2015 we collected data from 6181 different devices. According to the selected locale,

[1]The drawing application Pen & Paper in Google Play: https://play.google.com/store/apps/details?id=net.nhenze.penandpaper

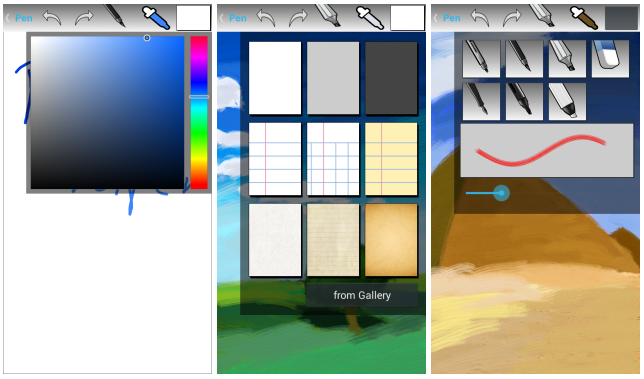



**Figure 2. Screenhots of the drawing application we used to collect strokes.**

the devices come from 83 different countries including Great Britain (846 devices) and Germany (351 devices). Most common device types in the dataset are the Galaxy Note 4 (132 devices), the Galaxy Note 3 (124 devices) and the Nexus 7 (79 devices). In total, we collected 1,457,231 strokes consisting of 36,105,759 touch events from the 6,181 devices.

As we assume that the strokes users draw are influenced by a large number factors including display size, device performance, and weight, we decided to focus on a small subset of devices that are common in our dataset. We only consider devices that provided at least two strokes and we selected a subset containing only data from eight different device types: We considered data from the Nexus 7 developed by Asus (79 devices, 15,610 strokes, 276,281 events), the Samsung Galaxy Note 3 (124 devices, 36,754 strokes, 892,360 events), the Samsung Galaxy Note 4 (131, devices, 30,321 strokes, 571,587 events), the Samsung Galaxy S5 (63 devices, 24,130 strokes, 518,433 event), the Samsung Galaxy S6 (41 devices, 10,117 strokes, 265,319 events), the LG Nexus 5 (32 devices, 4,683 strokes, 98,105 events), the Samsung Galaxy Tab S 8.4 (31 devices, 14,116 strokes, 263,341 events), and the Samsung Galaxy S4 (28 devices, 3,259 strokes, 94,743 events). In total, the subset contains 2,980,169 touch events, from 138,990 strokes that were produced on 529 different devices.

### Constructing the input vector

We aim to predict where the user's finger will be in the near future based on the previous movement of the finger. At runtime, we receive a stream of x/y touch position together with a timestamp from the touchscreen. At each point in time, we can use the previous positions to predict the subsequent positions. Using the recorded data we know for each touch event not only the previous but also the subsequent positions. Using a sliding window we therefore partition the collected data into samples consisting of a defined number (e.g. n=17) of touch positions. From the samples, we derive features that can be feed into supervised learning algorithms. For n=17, the first 10 touch events would be used as the previous touch events that go into the supervised learning algorithm. The aim is to predict the remaining 7 events.

We perform three steps to make the sample rotation, position, and speed invariant. 1) First, to make the samples rotation invariant we determine the angle $\alpha$ of the vector between the last two touch events that go into the learning algorithm and rotate the whole sample to align the vector with the y-axis. 2) Second, to make the samples position invariant we take the first derivation of the x/y position, thereby deriving the speed of the finger along both axis from the touch positions. 3) Third, we normalize the length of the sample by determining the total length $\beta$ of the input part of the sample and dividing each component of the whole sample by $\beta$.

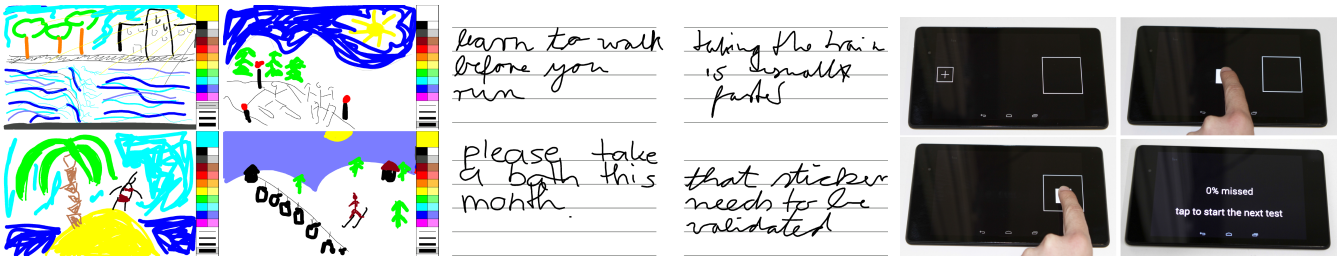
The y and the x component of the preprocessed sample serve as the input into the supervised learning algorithm. In addition, we feed the algorithm with the angle $\alpha$, the normalized total length of the sample $\beta$, and the time the algorithm should look into the future. Finally, we grouped the subsamples eight devices described in the previous section by their screen size in five groups and add a binary vector that indicates the screen size of the device the sample originates from. Using 10 consecutive touch events (the 10 most recent ones) results in a total of 26 input values that go into the supervised learning algorithm. As the output values, we use one of the components that followed the last event that served as input.

### Neural network training

Using the dataset and the approach for constructing the input vector we derived a training set. Using the machine learning framework Encog [11] and custom code, we experimented with different supervised learning algorithms and different sample sizes. Using cross validation we tested, for example, support vector machines and nearest neighbor search. We also tested different numbers of touch events that go into the sample. Artificial neural networks and using ten touch events as input provided the lowest error. Therefore, we focused on neural networks and ten touch event as input in the following.

Using again cross validation we tested different network configurations and activation functions. Besides the input layer with 26 neurons and the output layer with 2 neurons, we decided for a single hidden layer with 96 neurons. Using additional hidden layers increased the error and using additional neurons in the hidden layer increased the training time without decreasing the error. As values in the input sample can be both positive as well as negative we use a hyperbolic tangent (TanH) activation function which typically performs better compared to other differentiable and monotonic activation functions [16]. We trained the network using resilient propagation (RPROP) [5, 29] as previous comparisons suggest that RPROP provides a high training speed and a good performance [9, 30].

Finally, we randomly divided the training set into six parts and trained ensembles of neural networks [10]. We trained a set of networks that each predicts one of the seven touch events that follow the sample. For each distance from the last touch event that went into the input vector we trained six neural networks. To predict a touch position that followed the sample after a certain number of events (e.g. t=4) we combine the output of the networks that predict the previous (t=3), the current (t=4) and the subsequent (t=5) event. Thereby, we combine the output of 18 neural networks. We trained six networks for each of the seven touch events that follow the




(a) (b) (c)

**Figure 3. (a) Four exemplary pictures painted by the participants. (b) Exemplary phrases written by the participants. (c) The Fitts' Law dragging task used in the study.**

sample resulting in 42 neural networks. We trained each network for about six hours on an Intel processor with 16 hardware cores resulting in a total training time of about 10 days.

## COMPARING PREDICTION APPROACHES

We conducted an experiment to compare the developed neural network-based prediction with other approaches. In the study, we asked participants to perform three representative tasks that involve moving the finger across a touchscreen. We recorded the finger movement and used the collected data as a ground truth to determine the error for each approach and compensated amount of latency.

### Method

The experiment consisted of three tasks to compare the prediction approaches: writing, Fitts' Law, and drawing (see Figure 3a, 3b). We also considered three prediction levels: 33.3ms, 66.7ms, and 100ms which is equivalent to looking 2, 4, and 6 touch events in the future assuming a constant refresh rate of 60Hz. As we processed the recorded data offline we know for each touch event not only the previous finger positions but also the subsequent finger positions.

We conducted the experiment using an Android application that presents the tasks and records all occurring touch events. We used a Nexus 7 tablet with a 7.02 inch (178mm) display and a resolution of $1920 \times 1200$ pixels. The device provides touch events and updates the screen with a constant rate of 60Hz. We deactivated all background services and used a low-level function provided by the Android system to record touch events. We measured a latency of 100ms using a 240Hz camera (see [4]). We recruited 6 male and 2 female participants with an average age of 27.25 years (SD=2.96).

In the first task, participants had to write 10 phrases randomly selected from the phrase set by MacKenzie et al. [18] with their index finger. In the second task, participants performed a Fitts' Law tasks that replicated the design by Jota et al. [14]. Participants dragged a square inside a target square. We used three target sizes (24mm, 32mm, and 40mm) and three distances (28mm, 68mm, 120mm), resulting in nine combinations. Each combination was repeated eight times resulting in 72 tasks. We randomized the order of all task. Between the Fitts' Law tasks we displayed the participant's average error rate. If the error rate was below 5% we encouraged participants to speed up and if the error rate was above 5%, we encouraged them to be more precise. Compared to Jota et al. [14], we scaled down distances and target sizes by 20% to fit the smaller display of the used 7 inch tablet. Finally, participants had to draw a scene from their holiday. We asked them to complete the drawing in less than five minutes. In case participants did not come up with an idea for a picture we encouraged them to draw a scene from an imaginary holiday.

### Results

In total, we collected 70,917 touch events for the writing task, 7,518 for the Fitts' Law task, and 36,737 for the drawing task. Figure 3a shows four exemplary drawings and Figure 3b shows four of the written sentences. For each recorded touch event we derived the 10 preceding and the 6 succeeding touch positions. In case a sequence contained a lift-off event we used the lift-off position for all subsequent events and if a sequence contained a touch-down event we used the touch-down position for all preceding events. Processing the data offline enables us to compare the predicted position with the actual position. We use the Euclidean distance between the predicted position and the actual position in 33.33ms, 66.67, and 100ms as the only error metric to compare the different approaches. This error metric combines lag and jitter in one metric.

Table 1 shows the average error for the approaches, tasks, and prediction distances. Considering each touch event as a sample, we used a repeated measures ANOVA with paired t-tests as post hoc tests to compare the conditions. Due to the large sample size all differences are statistically significant ($p<.001$). Thus, we refrain from a description of the inferential statistics. Compared to the baseline (no extrapolation), linear extrapolation (1st order polynomial) reduces the error for up to 66ms for all three tasks. Further looking in the future by 100ms, however, the baseline outperforms linear extrapolation for drawing and the Fitts' Law tasks. The same effect is apparent for all extrapolation approaches. Their error grows faster than the baseline. Increasing the degree of the polynomial increases the error. The results also show that the ensemble of neural networks clearly outperforms the other approaches. In comparison with the baseline for 66ms, the error is 51.2% smaller for writing, 53.1% smaller for drawing, and 49.9% smaller for the Fitts' Law task. Comparing the neural network extrapolation with linear extrapolation for 66ms, the average error is 31.3% smaller for writing, 46.2% smaller for drawing, and 33.9% smaller for the Fitts' Law tasks.

| | Writing | | | Drawing | | | Fitts' Law | | |
|---|---|---|---|---|---|---|---|---|---|
| Approach | 33.33ms | 66.67ms | 100ms | 33.33ms | 66.67ms | 100ms | 33.33ms | 66.67ms | 100ms |
| no extrapol. | 15.4px | 29.7px | 42.5px | 40.2px | 77.9px | 110.1px | 31.5px | 52.1px | 65.5px |
| 1st or. polyn. | 7.8px | 21.1px | 38.5px | 25.1px | 67.9px | 123.4px | 16.9px | 39.5px | 68.8px |
| 2nd or. polyn. | 10.5px | 32.6px | 67.3px | 35.3px | 107.3px | 215.6px | 32.6px | 97.2px | 196.8px |
| 3rd or. polyn. | 22.8px | 98.9px | 259.5px | 76.7px | 332.8px | 870.9px | 87.0px | 383.1px | 1010.7px |
| Neural networks | 6.0px | 14.5px | 24.8px | 16.1px | 36.5px | 60.2px | 11.4px | 26.1px | 43.4px |

**Table 1. Average Euclidean distance between the actual touch position in 33.33ms, 66.67ms, and 100ms and the position predicted by the different approaches. 10px are equivalent to 0.79mm and 10mm are equivalent to 127.20px. No extrapolation assumes that the touch position remains static and does not project into the future.**

### Discussion

We compared approaches that predict where the user's finger will be observed in 33.3ms, 66.7ms, and 100ms using three tasks. By recording the touch data without extrapolation we could compare the approaches using one dataset. The results show that extrapolation using an ensemble of neural networks outperforms polynomial extrapolation for all tasks. Further, the results show that the neural networks extrapolation reduces the objective error induced by latency. The results, however, cannot show if this can increase the users' performance. The error is a combination of the two factors: lag and jitter. Hardware-based reduction of latency, as used in [22], only reduce the lag without causing jitter. Extrapolation reduces the lag but at the same time increases jitter due to prediction errors. Latency reduces the users' performance [14] and Pavlovych and Stuerzlinger reported that latency has a stronger effect on human performance compared to low amounts of spatial jitter [26]. As Pavlovych and Stuerzlinger also discuss, jitter can also dramatically increases the error rate. Therefore, the effect of extrapolation on users' performance needs to be investigated.

## ANALYSING THE EFFECT OF PREDICTION

We conducted a second experiment to investigate the effect of software-reduced touchscreen latency. We integrated the ensemble of neural networks into the apparatus from the previous study. We collected objective measures using a Fitts' Law dragging task and collected subjective feedback for the Fitts' Law and a drawing task.

### Method

The study uses a within-subjects design with two tasks. In the first task participants perform the Fitts' Law dragging task used for the previous study. We used three target sizes $\times$ three distances $\times$ eight repetitions, resulting in 72 tasks per condition. With the two prediction models and a baseline with no prediction, the study comprised 216 dragging tasks. The target sizes and distances used in the Fitts' law task were the same as in the first study. The order of the dragging tasks was randomized to avoid sequence effects. In the second task participants could freely draw images. In both tasks, we compared three conditions: (1) no prediction (baseline), (2) predicting the finger's position 33.3ms (two touch events at 60Hz) in the future, and (3) predicting the finger's position 66.7ms (four touch events at 60 Hz) in the future. For the Fitts' Law task we collected the time it took to select the target and the error rate. For both tasks we asked participants to rate the jitter, lag and the unpleasantness of the condition.

We recruited 14 participants (6 female, 8 male) with an average age of 26.3 years. None of the participant took part in the first study. We used the same Nexus 7 tablet as in the first study. We extended the Android application from the first study to implement the 33.3ms and the 66.7ms prediction. We integrated the neural network prediction into the apparatus used in the previous study. Using the Android version of the Encog machine learning framework we simply load the pre-trained networks into the application. As the Nexus 7 is equipped with a quad-core Snapdragon S4 Pro processor, we feed the input data from the touchscreen into four neural networks in parallel. One prediction using the whole ensemble of neural networks takes about five milliseconds on average.

### Results

We analyzed the results using a repeated measures ANOVA and Bonferroni-corrected t-tests if appropriate. Analyzing the results of the first task we determined if the prediction has an effect on the error rate. Without prediction, participants failed for 3.67% (SD=2.59%) of the tasks, with 33.33ms prediction they failed to correctly complete 5.56% (SD=3.18%), and with 66.67ms prediction the failed to correctly complete 5.85% (SD=3.70%) of the tasks. A repeated measures ANOVA did not reveal that the condition had an effect on the number of errors participants made ($F(2,26)=3.24$, $p=.06$).

Without prediction, the average task completion time is 452ms (SD=122ms), 353ms (SD=68ms) with 33.33ms prediction, and 368ms (SD=65ms) with 66.67ms prediction. An ANOVA shows that prediction reduces the task completion time ($F(2,26)=8.23$, $p=.002$). Post hoc tests showed that without prediction, participants needed more time compared to 33.33ms prediction ($p=.025$) and 66.67ms prediction ($p=.048$). The analysis did not reveal a difference between 33.33ms prediction and 66.67ms prediction ($p=.676$). In line with [14] we determined if prediction has an effect on throughput (bits/s). The average throughput is 3.81 bits/s (SD=0.73 bits/s) without prediction, 4.71 bits/s (SD=0.62 bits/s) with 33.33ms prediction, and 4.46 bits/s (SD=0.51 bits/s) with 66.67ms prediction (see Figure 5). A repeated measures ANOVA showed that prediction has an effect on the average throughput ($F(2,26)=10.11$, $p=.001$).Post hoc tests revealed a lower throughput without prediction compared to 33.33ms prediction ($p=.007$) and 66.67ms prediction ($p=.035$). We found no difference between 33.33ms prediction and 66.67ms prediction ($p=.335$)

Analyzing participants' subjective feedback (see Figure 4 left) revealed an effect on the perceived jitter ($F(2,26)=48.26$,

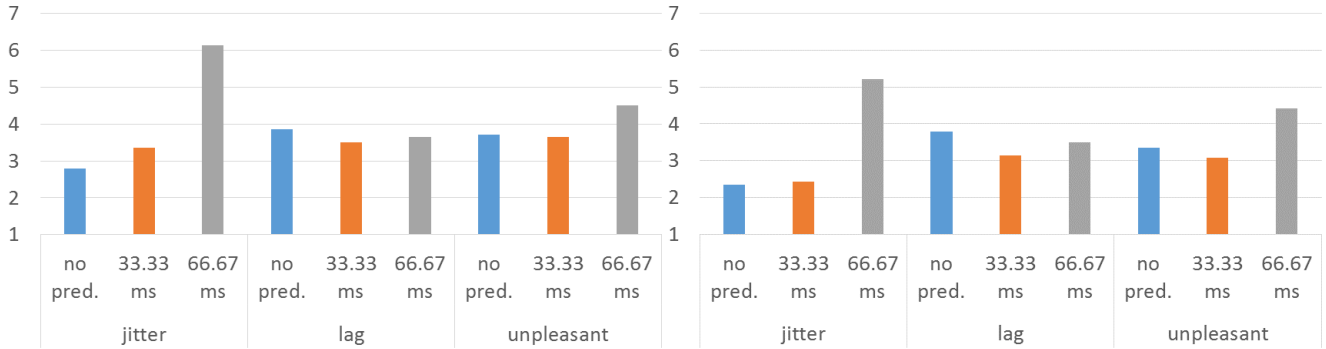



**Figure 4. Participants average subjective rating for the Fitts' law task (left) and for the drawing task (right).**

p<.001), the average rating of the jitter was 2.79 (SD=1.25) without prediction, 3.36 (SD=1.45) with 33.33ms prediction, and 6.14 (SD=0.77) with 66.67ms prediction. A post hoc test revealed that there was more perceived jitter with 66.67ms prediction compared to the other conditions (both p<.001). There was no significant difference between no prediction and 33.33ms prediction (p=.537). An ANOVA revealed no effect on the perceived lag (F(2,26)=0.27, p=.763). The average rating of the lag was 3.86 (SD=2.03) without prediction, 3.50 (SD=1.83) with 33.33ms prediction, and 3.64 (SD=0.93) with 66.67ms prediction. An ANOVA did also not reveal an effect on the unpleasantness (F(2,26)=1.92, p=.168). The average rating of the unpleasantness was 3.71 (SD=2.02) without prediction, 3.64 (SD=1.55) with 33.33ms prediction, and 4.50 (SD=1.74) with 66.67ms prediction.

For the second task we collected only subjective feedback (see Figure 4 right). An ANOVA revealed an effect on the perceived jitter (F(2,26)=15.84, p<.001). The jitter was rated 2.36 (SD=1.45) without prediction, 2.43 (SD=1.16) with 33.33ms prediction, and 5.21 (SD=1.93) with 66.67ms prediction. Post hoc tests revealed a difference between 66.67ms prediction and no prediction (p=.005) as well as with 33.33ms prediction (p=.001). We found no difference between no prediction and 33.33ms prediction (p=1.0). We found no effect on the perceived lag (F(2,26)=0.47, p=.631). The lag was rated 3.79 (SD=1.76) without prediction, 3.14 (SD=1.56) with 33.33ms prediction, and 3.50 (SD=1.70) with 66.67ms prediction. An ANOVA revealed an effect on the unpleasantness (F(2,26)=4.861, p=.016). The average rating of the unpleasantness was 3.36 (SD=2.02) without prediction, 3.07 (SD=1.49) with 33.33ms prediction, and 4.43 (SD=1.50) with 66.67ms prediction. Post hoc tests revealed that 66.67ms prediction was rated less pleasant than 33.33ms prediction. We found no difference between no prediction and 33.33ms prediction (p=1.0) or 66.67ms prediction (p=.192).

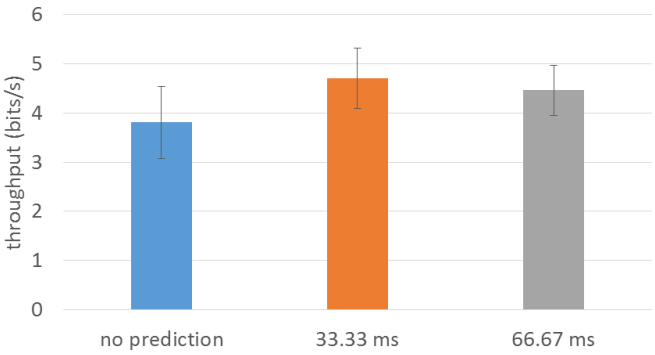



**Figure 5. Average throughput for the conditions. Error bars show standard error.**

## Discussion

The results show that prediction can significantly increase users' speed and throughput for a dragging task. At the same time we found no significant effect on the error rate. Thus, the results suggest that extrapolating the movement of the users' finger to reduce the latency can increase the users' speed without having a relevant effect on the errors made. Subjective feedback shows for both tasks that the 66.67ms prediction results in significantly more perceived jitter. For the second task, subjective feedback shows that 66.67ms prediction is less pleasant than 33.33ms prediction. Overall, objective results show that prediction can improve users' performance. While 66.67ms prediction results in more perceived jitter and can be perceived as less pleasant, we found no significant differences between no prediction and 33.33ms prediction. Descriptive results suggest, however, that the 33.33ms prediction might result in a more pleasant experience.

MacKenzie and Colin showed a linear correlation between speed and latency [19]. We, however, observed almost the same performance if predicting the finger's position in 33.33ms and 66.67ms. As the speed is significantly higher compared to the baseline, it can be concluded that there is no linear correlation for the prediction approach used. We assume that this is a result from the interaction of latency and jitter (see [26] for a discussion of the trade-off between latency and jitter). While the level of prediction increases, the perceived latency decreases. This accordingly increases the users' speed. The prediction error (which equals to jitter) also increases with the level of prediction which decreases the speed.Thus, there is a trade-off between the amount of compensated latency and the amount of resulting prediction error.

## CONCLUSIONS AND FUTURE WORK

In this paper, we investigated how to improve users' performance by reducing the latency of touchscreens using a software-bassed approach. We proposed different approaches to extrapolate the fingers' position on a touchscreen from the previous movement. Based on a dataset consisting of touch

events from three representative tasks, it is shown that extrapolating using an ensemble of neural networks trained on a large dataset results in the lowest prediction error. We further show that extrapolation significantly increases users' performance for a Fitts' Law task. As the approach is purely software-based it could be easily integrated into existing applications and systems.

The prediction approaches considered in this paper can only be a selection of the possible techniques. Different machine learning algorithms are certainly worth exploring. We believe that considering the physiological restrictions of the human hand when interacting with mobile devices is a promising direction. Using a model of the human hand that predicts the probability of particular hand postures could help to extrapolate the hand's movement. Such an approach would naturally further benefit from multi-touch tasks where more than one finger touches the screen at a time. Furthermore, 33.33ms prediction only reduces one-third of the latency if the device has an end-to-end latency of 100ms. Using devices that already have less latency, however, 33.33ms prediction could already remove most latency.

## ACKNOWLEDGMENTS

This work is funded by the German Research Foundation within the SimTech Cluster of Excellence (EXC 310/1) and by the MWK Baden-Württemberg within the Juniorprofessuren-Programm.

## REFERENCES

1. Fakhreddine Ababsa, Malik Mallem, and David Roussel. 2004. Comparison between particle filter approach and Kalman filter-based technique for head tracking in augmented reality systems. In *Proceedings or the IEEE International Conference on Robotics and Automation*, Vol. 1. IEEE, 1021–1026. DOI: http://dx.doi.org/10.1109/ROBOT.2004.1307284
2. Robert S Allison, Laurence R Harris, Michael Jenkin, Urszula Jasiobedzka, and James E Zacher. 2001. Tolerance of temporal delay in virtual environments. In *Virtual Reality, 2001. Proceedings. IEEE*. IEEE, 247–254. DOI: http://dx.doi.org/10.1109/VR.2001.913793
3. Glen Anderson, Rina Doherty, and Subhashini Ganapathy. 2011. User Perception of Touch Screen Latency. In *Design, User Experience, and Usability. Theory, Methods, Tools and Practice*. Springer, 195–202. DOI: http://dx.doi.org/10.1007/978-3-642-21675-6_23
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*. ACM, 241–250. DOI: http://dx.doi.org/10.1145/2512349.2512796
5. H Braun and M Riedmiller. 1992. RPROP: a fast adaptive learning algorithm. In *Proceedings of the International Symposium on Computer and Information Science VII*.
6. Timothy J Buker, Dennis A Vincenzi, and John E Deaton. 2012. The Effect of Apparent Latency on Simulator Sickness While Using a See-Through Helmet-Mounted Display Reducing Apparent Latency With Predictive Compensation. *Human Factors: The Journal of the Human Factors and Ergonomics Society* 54, 2 (2012), 235–249. DOI: http://dx.doi.org/10.1177/0018720811428734
7. 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*. ACM, New York, NY, USA, 205–214. DOI: http://dx.doi.org/10.1145/2817721.2817736
8. Stephen R Ellis, François Bréant, Brian M Menges, Richard H Jacoby, and Bernard D Adelstein. 1997. Operator interaction with virtual objects: effect of system latency. *Advances in human factors/ergonomics* (1997), 973–976.
9. JM Hannan and JM Bishop. 1997. A comparison of fast training algorithms over two real problems. In *Artificial Neural Networks, Fifth International Conference on (Conf. Publ. No. 440)*. IET, 1–6.
10. Lars Kai Hansen and Peter Salamon. 1990. Neural network ensembles. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 10 (1990), 993–1001.
11. Jeff Heaton. 2015. Encog: Library of Interchangeable Machine Learning Models for Java and C#. *Journal of Machine Learning Research* 16 (2015), 1243–1247. http://jmlr.org/papers/v16/heaton15a.html
12. Niels Henze and Martin Pielot. 2013. App Stores: External Validity for Mobile HCI. *interactions* 20, 2 (March 2013), 33–38. DOI: http://dx.doi.org/10.1145/2427076.2427084
13. Niels Henze, Martin Pielot, Benjamin Poppinga, Torben Schinke, and Susanne Boll. 2011. My app is an experiment: Experience from user studies in mobile app stores. *International Journal of Mobile Human Computer Interaction (IJMHCI)* 3, 4 (2011), 71–91.
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*. ACM, 2291–2300. DOI: http://dx.doi.org/10.1145/2470654.2481317
15. Topi Kaaresoja and Stephen Brewster. 2010. Feedback is... late: measuring multimodal delays in mobile device touchscreen interaction. In *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction*. ACM, 2. DOI: http://dx.doi.org/10.1145/1891903.1891907

16. Bekir Karlik and A Vehbi Olgac. 2011. Performance analysis of various activation functions in generalized MLP architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems* 1, 4 (2011), 111–122.

17. Edward Lank, Yi-Chun Nikko Cheng, and Jaime Ruiz. 2007. Endpoint prediction using motion kinematics. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 637–646. DOI: http://dx.doi.org/10.1145/1240624.1240724

18. I Scott MacKenzie and R William Soukoreff. 2003. Phrase sets for evaluating text entry techniques. In *CHI'03 extended abstracts on Human factors in computing systems*. ACM, 754–755. DOI: http://dx.doi.org/10.1145/765891.765971

19. I Scott MacKenzie and Colin Ware. 1993. Lag as a determinant of human performance in interactive systems. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*. ACM, 488–493. DOI: http://dx.doi.org/10.1145/169059.169431

20. Michael Meehan, Sharif Razzaque, Mary C Whitton, and Frederick P Brooks Jr. 2003. Effect of latency on presence in stressful virtual environments. In *Virtual Reality, 2003. Proceedings. IEEE*. IEEE, 141–148. DOI:http://dx.doi.org/10.1109/VR.2003.1191132

21. W Todd Nelson, Merry M Roe, Robert S Bolia, and Rebecca M Morley. 2000. *Assessing simulator sickness in a see-through HMD: Effects of time delay, time on task, and task complexity*. Technical Report. DTIC Document.

22. 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*. ACM, 453–464. DOI: http://dx.doi.org/10.1145/2380116.2380174

23. Phillip T Pasqual and Jacob O Wobbrock. 2014. Mouse pointing endpoint prediction using kinematic template matching. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 743–752. DOI:http://dx.doi.org/10.1145/2556288.2557406

24. Andriy Pavlovych and Carl Gutwin. 2012. Assessing Target Acquisition and Tracking Performance for Complex Moving Targets in the Presence of Latency and Jitter. In *Proceedings of Graphics Interface 2012*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 109–116. http://dl.acm.org/citation.cfm?id=2305276.2305295

25. Andriy Pavlovych and Wolfgang Stuerzlinger. 2009a. The Tradeoff Between Spatial Jitter and Latency in Pointing Tasks. In *Proceedings of the 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. ACM, New York, NY, USA, 187–196. DOI: http://dx.doi.org/10.1145/1570433.1570469

26. Andriy Pavlovych and Wolfgang Stuerzlinger. 2009b. The tradeoff between spatial jitter and latency in pointing tasks. In *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*. ACM, 187–196. DOI: http://dx.doi.org/10.1145/1570433.1570469

27. Andriy Pavlovych and Wolfgang Stuerzlinger. 2011. Target Following Performance in the Presence of Latency, Jitter, and Signal Dropouts. In *Proceedings of Graphics Interface 2011*. Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 33–40. http://dl.acm.org/citation.cfm?id=1992917.1992924

28. Martin Pielot, Niels Henze, and Susanne Boll. 2011. Experiments in app stores-how to ask users for their consent. In *Proceedings of the workshop on Ethics, logs & videotape*.

29. Martin Riedmiller and Heinrich Braun. 1993. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Neural Networks, 1993., IEEE International Conference on*. IEEE, 586–591.

30. Wolfram Schiffmann, Merten Joost, and Randolf Werner. 1993. Comparison of optimized backpropagation algorithms. In *In Proceedings of the European Symposium on Artificial Neural Networks*, Vol. 93. Citeseer, 97–104.

31. Richard HY So and German KM Chung. 2005. Sensory motor responses in virtual environments: Studying the effects of image latencies for target-directed hand movement. In *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*. IEEE, 5006–5008. DOI: http://dx.doi.org/10.1109/IEMBS.2005.1615599

32. Haijun Xia, Ricardo Jota, Benjamin McCanny, Zhe Yu, Clifton Forlines, Karan Singh, and Daniel Wigdor. 2014. Zero-latency tapping: using hover information to predict touch locations and eliminate touchdown latency. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, 205–214. DOI:http://dx.doi.org/10.1145/2642918.2647348

33. Brian Ziebart, Anind Dey, and J Andrew Bagnell. 2012. Probabilistic pointing target prediction via inverse optimal control. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*. ACM, 1–10. DOI: http://dx.doi.org/10.1145/2166966.2166968