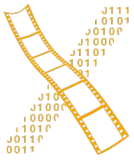


RWTH AACHEN UNIVERSITY

A Predictive Approach to Compensate Latency of Tangibles on Capacitive Multi-Touch-Displays

Bachelor's Thesis
submitted to the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
David Asselborn

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr. Leif Kobbelt

Registration date: 28.07.2016
Submission date: 22.09.2016

Eidesstattliche Versicherung

_____ _____

Name, Vorname Matrikelnummer

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel

_____

_____

_____

selbständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

_____ _____

Ort, Datum Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

_____ _____

Ort, Datum Unterschrift

Contents

List of Figures

List of Tables

Abstract

Tangibles are useful extensions of today's commonly used touch interfaces. But latency weakens the perception of tangibles being a seamless part of digital applications. When performing direct dragging tasks, latency is perceptible as the detected position is lagging behind the actual object. Especially high motion applications like an air hockey game using tangibles as mallets suffer from limited responsiveness.

Therefore we present a software approach to partly compensate system latency. The idea is to analyze the trace of a tangible in real time and try to anticipate its movement in short-scale. With prediction we can shift the detected position closer towards the real position of the tangible. Different prediction models are implemented: first simple short-term linear prediction as presented by Cattan et al. [2015b] and second polynomial regression allowing to predict curves.

These models are compared to each other in various configurations to get to know more about their strengths and weaknesses. Among others the euclidean distance between predicted and actual position is measured in a benchmark. Finally a user study is performed to validate theoretical results and see how participants perceive the effects of touch prediction. By taking up the motivation of the thesis they have to play air hockey with and without prediction. Overall 10 out of 14 participants preferred the game with touch prediction. This shows that the approach has the potential to be used in more applications and allows us to look ahead positively regarding future work.

Überblick

Tangibles stellen eine sinnvolle Erweiterung der heute weit verbreiteten Touchscreen Benutzeroberflächen dar. Latenz jedoch schwächt die Wahrnehmung eines einheitlichen Systems aus physischem Objekt und digitaler Umgebung. Werden Tangibles über den Tisch bewegt, sind selbst geringe Latenzen wahrnehmbar, da die erkannte Position der tatsächlichen hinterher hängt. Bewegungsintensive Anwendungen, zum Beispiel ein Air Hockey Spiel, welches Tangibles als Schläger nutzt, leiden unter eingeschränkter Reaktivität.

Daher präsentieren wir einen Softwareansatz zur teilweisen Latenzkompensation. Die Idee dahinter ist, den Pfad eines Tangibles permanent in Echtzeit zu analysieren und dadurch dessen Bewegung für einen kurzen Zeitraum vorherzusagen. Mittels unserer Vorhersage können wir die erkannte und tatsächliche Tangible Position näher zusammenbringen. Wir haben hierfür verschiedene Modelle implementiert: Erstens ein lineares Vorhersagemodell, wie bereits von Cattan et al. [2015b] präsentiert und zweitens ein Modell, welches mittels polynomialer Regression versucht, auch Kurven zu antizipieren.

Diese beiden Modelle vergleichen wir in verschiedenen Konfigurationen, um mehr über jeweilige Stärken und Schwächen herauszufinden. Unter anderem wird in einem Benchmark die euklidische Distanz zwischen vorhergesagter und dann tatsächlich erkannter Position berechnet. Abschließend führen wir eine Studie durch, um unsere theoretischen Ergebnisse zu validieren und zu analysieren, wie Nutzer die Effekte unserer Vorhersage wahrnehmen. Wir nehmen hierfür die initiale Motivation der Arbeit wieder auf und lassen die Teilnehmer mit und ohne kompensierte Latenz Air Hockey spielen. Insgesamt bevorzugten 10 von 14 Teilnehmern das Spiel mit kompensierter Latenz. Somit hat der Ansatz Potential, auch in weiteren Anwendungen genutzt zu werden, was uns positiv in Richtung zukünftiger Arbeit in diesem Bereich blicken lässt.

Acknowledgements

First of all I want to thank my supervisor Christian Cherek for giving helpful feedback when needed during the past few months while also providing enough space for my own ideas.

Moreover I want to thank Prof. Dr. Jan Borchers for letting me write the thesis at his chair. It was really interesting to use the available technology at the chair and I would be glad to attend additional courses in the field of HCI later on. Of course I also want to thank Prof. Dr. Leif Kobbelt for being my second examiner.

To all participants of my study, thank you for taking your time and giving helpful comments.

Last but not least, I want to thank my family and friends, especially Susanne, Ingrid and Andreas, for supporting and advising me.

Conventions

Throughout this thesis we use the following conventions.

Definitions of technical terms or short excursus are set off in coloured boxes.

EXCURSUS:
Excursus are detailed discussions of a particular point in a book, usually in an appendix, or digressions in a written text.

Definition:
Excursus

Source code and implementation symbols are written in typewriter-style text.

```
myClass
```

To increase readability when declaring functions their specific arguments are left out.

The whole thesis is written in American English. Although the thesis is written by one person plural form is used.

Chapter 1

Introduction

Touch screens and tangibles: an already perfect relationship?

Touch screens in combination with tangibles should bring the best of two worlds together. On the one hand we have tangibles: they are physical objects which can be detected by touch screens. Placed on them they can be moved actively or work as passive tokens. Their different shapes offer all kind of affordances and haptic feedback providing clues about how to use each object. By that, tangibles extend the two-dimensional plane surface of a touch screen, adding an additional type of feedback besides acoustic and visual. Furthermore, tangibles can be designed in a way that they are clearly visible from varying positions and greater distances, a potential advantage on larger tabletop displays. That are reasons why digital environments benefit from the extension with physical objects. On the other hand tangibles have to be set into context a touch screen can provide. Touch points can be detected and interpreted, allowing for example gesture recognition. The whole interface remains interactive by updating its displayed content depended on the situation and input. As a result, tangibles also strongly benefit from touch screens. The relationship between tangibles and touch screens could therefore be described as a symbiosis. But to perceive tangibles as objects belonging to a digital application the overall system latency has to be minimized.

Definition: *Latency*

LATENCY:
The delay between input action and output response. MacKenzie and Ware [1993]

Latency creates gap between physical and virtual object

A computer needs time to recognize and process touch input as well as react to it accordingly through displaying a response. Because of that, latency is unavoidable on common touch screens up to the present. Using tangibles this leads to two usually different positions: first the actual position of the physical tangible on top of the display and second its last detected position by the computer. During movement this detected position is lagging behind the real position. When performing solely slow movements they are close together, but with fast movements the gap between physical and virtual object can become quite large as illustrated in figure 1.1. Of course users expect the actual tangible position to be used, because it is their visual and haptic reference point. In case the tangible is not additionally displayed virtually on the screen they do not even know the lagging second position. In contrast, the computer only has information about this second position and that can cause problems.

Extreme case air hockey

Imagine playing a reactive game like air hockey with tangibles as mallets and a virtual puck and playing field. At best, the virtual representation of a mallet should always be exactly below the actual object as they are moving simultaneously. Due to the explained multiple sources of latency which cannot all be erased completely, reducing the latency to achieve this state is currently not possible without restrictions. Because of that, the detected mallet position is lagging behind the movement of the tangible. This can cause errors and misunderstandings if for example the user is not hitting the puck as intended. Resulting from it the user could blame himself for being too slow when in fact the system is too slow updating the tangible position for collision detection. Appearing problems like that are motivating the general aim of this thesis: to reduce the *perceived* latency. With a predictive approach we rather are compensating latency instead of actually reducing it, but if it works the effect on users can be the same.

(a) Initially static

(b) Slow movement

(c) Fast movement

Figure 1.1: Illustration of a physical tangible and its virtual representation with increasing movement speed during acceleration

Predictive approach to compensate latency

The idea behind the predictive approach in general is to continuously analyze the trace of a tangible in real time and try to predict its movement for a certain short time frame, called prediction length. If, for example, this prediction length equals the system latency, it could be completely counteracted with a perfect prediction. It goes without saying that a prediction is always connected with uncertainty, nevertheless we try to shift the detected position closer towards the real position. For example, short-time linear prediction as presented by Cattan et al. [2015b] uses the last two detected positions to calculate the current direction of movement and its speed and assumes that the tangible continues this movement. The distance that the object presumably covers during the time we want to predict is added to the detected position. Detailed explanations are following in the main part of the thesis.

Restrictions of current work motivates extension

There exist two main restrictions of this simple linear model: it assumes first that *the movement continues with constant speed* and second that *the movement follows a straight line*. With larger prediction lengths this leads to a jittering movement when dealing with curves and under- or overshooting when dealing with speed changes. Because of that, there is definitely room for improvement. Therefore, it is worth testing if a polynomial regression on the trace of a tangible is helpful. It would take multiple last measured points and the speed between them into calculation to detect acceleration and deceleration phases. That could help to prevent under- and overshooting. Most important, it could also be used to tackle the second limitation and smoothen the movement during curves.

Structure of the thesis

With the upcoming related work we are going to motivate the use of tangibles and go into detail about different types of them. The impact of latency in various areas is pointed out along with possible approaches to reduce it by hardware or software. After that, the theoretical foundations of our work concerning linear prediction and its extension by polynomial regression are explained. Following, a short overview about relevant parts of the already existing framework is presented. How our own prediction is then embedded into it will be explained afterwards. To evaluate, first a benchmark is developed and used to get an idea about the influence of different parameter combinations, for example the polynomial degree and the amount of included previous positions. The prediction is tested and visualized on multiple example traces to get to know more about strengths and weaknesses of the models. Furthermore, we come back to the already mentioned extreme case of playing air hockey, performing a user study to find out how the prediction is perceived in real world by the participants. Finally, all the results are discussed and interpreted. Possible future work including new prediction models, different hardware and potential implementation improvements is mentioned as well.

Chapter 2

Related work

Research overview

My thesis is going to connect the research areas *tangibles*, *latency* and *prediction*. With the recent development of PERCs by Voelker et al. [2015] the next step of tangibles becoming a useful part of digital applications is achieved. But latency is still a problem. Various studies for example by Anderson et al. [2011] and Ng et al. [2012] have been made examining the impact of latency while using mouse, stylus or direct touch. Depending on the task users can perceive latencies even in the single digits. Achieving latency free hardware comes with restrictions as Leigh et al. [2014] demonstrate. Furthermore, a software approach is mostly independent of underlying hardware, so it can be used on different devices without complex adjustments. Among others Asano et al. [2005] study the movement trajectory of mouse input for endpoint prediction in WIMP environments with the purpose of using it on larger displays. Tangibles are often used on large tabletop display but not (yet) motorized, so this is not applicable to our current work environment. Focusing on shorter prediction intervals, LaValle et al. [2014] implement a software approach to reduce head movement latency of the virtual reality headset Oculus Rift. To compensate latency of touch input, short-term linear prediction is the most promising approach. With the model of Cattan et al. [2015b] the user can retain control, while the perceived latency is reduced.

2.1 Tangibles

Figure 2.1: iPhone as tangible can display information on third axis as shown by Li and Kobbelt [2015]

Why to use tangibles?

Tuddenham et al. [2010] compare the input via mouse, multi-touch and tangible performing manipulation and acquisition tasks. They conclude that users overall prefer tangibles as they are easy to use and enable precise adjustments. Another recent study by Cincuegrani et al. [2016] analyzes the influence of tangibles on the motivation of users. Participants had to perform a musical task either using tangibles or not. Resulting, their motivation was higher when interacting with a tangible user interface than a gesture-only interface. Shaer and Hornecker [2010] provide history, possible use cases and implementation methods of tangibles. Tangibles could be used in several application domains. Especially in education they can ease the initial contact with topics like programming, because they invite people to play around and help to visualize abstract concepts. The development of Li and Kobbelt [2015] suggests that tangibles themselves could also display additional information or provide an alternative view by being equipped with a display. As an example figure 2.1 shows an iPhone used as a tangible.

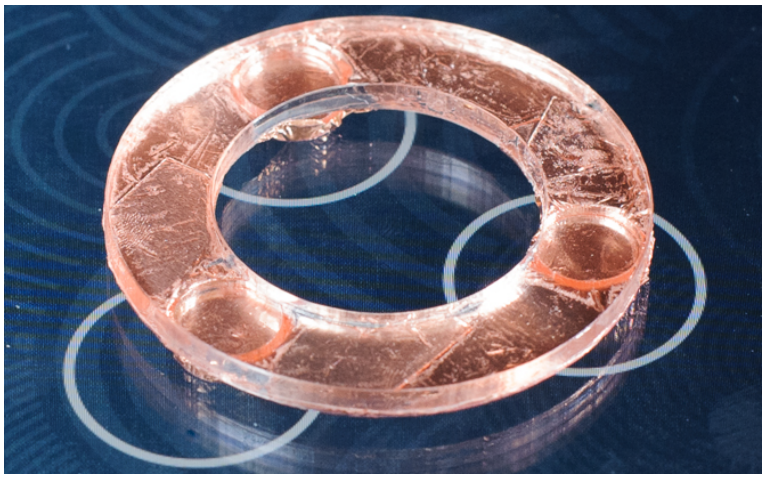


Figure 2.2: PUC creating three touch points developed by Voelker et al. [2013]

PUCs and PERCs

A passive untouched capacitive widget (PUC, Voelker et al. [2013]) is a passive tangible creating three touch points as shown in figure 2.2. Capacitive touch screens procedurally scan for touch input detected by changes in capacitance, but untouched PUCs are not grounded. Therefore, the three touch markers have to be arranged in a certain pattern to assure that only one of them is in a currently checked area at a time, then the other two can ground the tangible. Additionally, the exact orientation of the object can be detected having three positions. Unfortunately, when using PUCs the touch screen filters out their touch input after some time without movement. This can be avoided by establishing a connection via Bluetooth to transfer the status of the tangible to the device. This additional information channel helps to distinguish if the tangible position is lost due to missing movement or due to lifting it from the table. Voelker et al. [2015] describe and explain persistent capacitive tangibles (PERCs) which establish this connection and therefore provide a reliant detection of tangibles on touch screens even if they are not moved. Cherek et al. [2015] present different use cases for PERCs like a multiplayer air hockey game and a space invaders clone. As these are some exemplary input scenarios, they ideally should benefit from reduced latency through prediction. Because of my focus on latency, whose negative effects only occur with movement, in my work it is not important whether PUCs or PERCs are used.

2.2 Latency

Perception of latency

One of the first papers that focuses on analyzing the correlation between user performance with a mouse and input lag is the one by MacKenzie and Ware [1993]. The results are not simply transferable to tangibles, because of the indirect input method, but they show that latency has been a well known problem for a long time. Anderson et al. [2011] analyze the users' tolerance towards latency while performing everyday tasks on a touch screen. As a result, the participants accepted high latencies up to 580ms. This could be explained by the novel interaction with the used tablet computer screen at that time (transferable to initial contact with the input method tangibles) and the examined slow tasks (comparable to some tasks with tangibles like playing a strategy table top game). But other studies show that latency reductions way below 580ms can be perceived during more reactive tasks and result in a better user performance. One of these is a basic study that examines the possible lower bound of latency perception with direct touch input performed by Ng et al. [2012]. They highlight that differences can be noticeable down to 2ms. But it is too simple to be fixed to one number, as Jota et al. [2013] get different slightly higher results. Nevertheless, these publications support the importance of reducing perceived latency.

Influence of latency depending on task

Deber et al. [2015] examine the users' perceptions of latency for indirect and direct interaction, performing dragging and tapping tasks. Resultantly, they find out that dragging tasks with direct interaction are the most latency sensitive tasks. That leads to the assumption that tangible user interfaces would highly benefit from a reduced perceived latency. Jota et al. [2013] present the influence of latency on pointing tasks. The smaller and further away the target is (larger Fitts' ID), the greater becomes the negative impact of latency. As tangibles are often used on large surfaces with potentially small targets the influence of latency is supported.

Latency while using a stylus

Most studies concentrate on mouse or direct touch input, the question remains how latency is perceived using tangibles. As such studies are rare investigations about using a stylus are of interest, because the stylus is a transfer object to make an input right on the display just like tangibles do. Participants had to sketch and write using a stylus under different latency conditions in a study realized by Annett et al. [2014]. The lower bound of perceived latency is found out to be at around 50ms when drawing a line or writing and at around 60ms when drawing more complex shapes. Overall, this is considerably higher than the 2ms while performing direct touch tasks as observed by Ng et al. [2012]. Additionally the latency perception highly depends on the visual environment. With eye-tracking they emphasize that participants used larger objects like their own hands or the used stylus as a reference point for latency estimation. They did not focus on the specific gap between the actual nib of the stylus and its displayed virtual ink. In another study by Ng et al. [2014] participants should perform dragging and scribbling tasks with a digital pen. They were able to distinguish between very low latencies in the single digits when dragging a box and a median latency of 40ms while scribbling.

Gaming and latency

Latency has a high impact on performance when playing first person shooters including fast movements as Ivkovic et al. [2015] show. This could be transferred to high motion games on tabletops like air hockey. Additionally, Meehan et al. [2003] have an interesting approach to connect latency with the feeling of presence by measuring the heart frequency in virtual reality. As a result, with lower latencies the participants heart frequency was higher, signaling a more distinct feeling of presence even with the relatively poor graphics at that time. This indicates a connection between these two factors and could also occur in attenuated form using tangibles.

Measure latency with and without additional hardware

To analyze the impact of latency it first has to be measured to find out the status quo. This can be done with or without additional hardware as shown by Bérard and Blanch [2013]. In every case the actual input position has to be monitored continuously to compare it to the recognized position at each time frame. Using a high speed camera this position can be captured most precisely. Another approach to obtain the actual position without additional hardware is to strictly predefine the input movement. Through moving a finger with a given constant speed in a circle it is assumed to know the real finger position at every time frame. This position then can be compared to the acquired input to derive the overall latency by the gap between the two. The procedure leads to an estimation approximately 4ms away from the more precise one gained by using a camera. Cattan et al. [2015a] address the problem that following a circle with a given speed is not that easy to perform. They introduce an easy-to-perform approach for latency estimation that could potentially be used on our tabletop screen and additionally shows us a use case for the prediction model. Users have to move a finger at any desired but constant speed on a straight line. The detected position is visualized on the screen, with latency it is constantly lagging behind the finger. The overall latency is now measured by tweaking the prediction length until prediction fully compensates the displayed lag. This is indicated by a match of visualized and actual finger position. Validated by a user experiment, this method is more precise than the one introduced by Bérard and Blanch [2013] while being easy to perform.

2.3 Prediction

Why to use prediction?

By documenting the construction of a 1ms latency system, Ng et al. [2012] show that building an almost latency-free system is possible but complex. The work of Leigh et al. [2014] supports that assumption. A lot of parts like the touch recognition, its processing and the display have to be optimized. Currently their system comes with restrictions. A projector has to be used to display content on the touch surface and multi-finger gestures are not supported. Overall a software approach is preferred, because it can be widely used on today's common hardware. Vaidyanathan [2008] gives fundamental information about prediction, the formula of Cattan et al. [2015b] can be derived by the approaches here. In general, with prediction comes uncertainty and false predictions could cause jittering movement. Pavlovych and Stuerzlinger [2009] examine the trade off between spatial jitter and overall latency. A high amount of jitter can strongly increase the error rate whereas latency has a stronger effect on human performance than a small amount of jitter.

Different predictive approaches

This thesis will focus on continuous short-term prediction as already pointed out in the introduction. Nevertheless, there exist other types of prediction. Xia et al. [2014] aim to reduce the overall interaction latency with touch screens by predicting the location and time of a tap. But Deber et al. [2015] conclude that tapping tasks are not as latency sensitive as dragging tasks. Another type of prediction is endpoint prediction. While also analyzing the trajectory of a touch it aims for a much longer prediction length. This could of course help to cover large distances on tabletops, but without motorized tangibles an implementation would not be useful. Asano et al. [2005] show the Delphian desktop which predicts mouse movements in WIMP environments by analyzing cursor trajectories. Described knowledge about the hand movement phases could be of interest, because acceleration and adjustment phases are difficult to deal with using prediction. Lank et al. [2007] share the aim to predict user input for performance improvements with a longterm approach. Biswas et al. [2013] further develop different endpoint prediction methods by using a neural net-

work or a Kalman filter taking into account velocity, bearing, distance and acceleration. Kalman filters have already been tested on our system, they are not capable to make appropriate position corrections in our tangible system as they deal best with noisy data. But they could be used to filter out glitches of the display's touch recognition. To predict the endpoint of a mouse movement by comparing the initial movements to already known personal patterns is another approach by Pasqual and Wobbrock [2014]. This algorithm needs some data before it can be accomplished. For a personal device this is acceptable, but a larger tabletop display especially in public could be used by different people with different movement patterns every day.

Effects of continuous prediction

One type of prediction we also want to implement has already been evaluated with direct touch. Cattan et al. [2015b] use continuous linear prediction to entirely counteract the latency in a range of 25ms to 75ms. They examine test users' performance and subjective preferences. First participants had to perform dragging tasks with varying difficulty and a system latency of 25 to 75ms. This latency was then either completely counteracted using prediction or not. Above a compensated latency of 42ms user performance was not increased through prediction. This could be explained by the rising prediction error that especially interferes with the needed precise adjustments during target acquisition. Nevertheless, counteracting 25ms of latency led to an improved performance for every task difficulty. In another task users should perform moves as they liked with and without prediction and choose their preference. Only when compensating 25ms, participants preferred an enabled prediction as some otherwise felt a loss of control. As highlighted by Pavlovych and Stuerzlinger [2009] jitter can increase the error rate and seems to occur when compensating higher latencies. It is important to keep that in mind and carefully analyze the impact of greater prediction lengths on the movement trajectory.

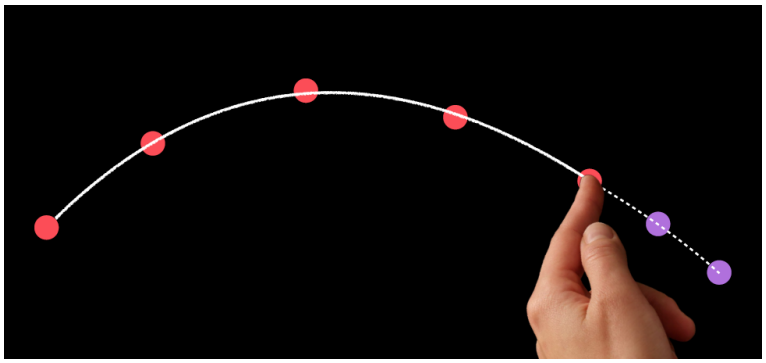


Figure 2.3: iOS9 includes touch prediction as presented by Tsoi and Xiao [2015]

Applied approaches to deal with latency

In practice a form of touch prediction has already been implemented by Apple since the release of iOS9. The approach is presented by Tsoi and Xiao [2015], but implementation details are not given. When looking at figure 2.3 you get the impression that Apple has extended the prediction beyond linear, because the predicted dashed line slightly follows the curve. Lags are also tackled in the area of remote shape displays: Leithinger et al. [2014] provide a scenario where shape displays are used to work together in a shared space from different locations. They have problems with latency during their projected environment changes. To handle this they have to control the object movement, synchronizing original and projected movement. LaValle et al. [2014] show that virtual reality is another latency sensitive area where latency has to be minimized to enable a realistic experience and prevent motion sickness. They also use a predictive approach to reduce the perceived latency between head movement and screen change on the Oculus Rift.

Chapter 3

Touch Prediction

3.1 Theory

With prediction in general we try to make an educated guess about what will happen in future. Hints about any progression can be obtained from relevant data if available. In our case we want to predict movement on a small-scale to partly compensate latency. Let us define

$$p_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \tag{3.1}$$

as the two-dimensional position a touch is detected at frame $i \in \mathbb{N}$. To predict any future value p_{i+n} with $n \in \mathbb{N}$ we can try to gather information out of already recorded touch points p_i, p_{i-1}, p_{i-2} etc.

Linear short-term prediction

With short-term linear prediction as used by Cattan et al. [2015b] only the last two detected positions are taken into consideration for prediction. We proceed as following: at the time a touch point is detected we calculate the current speed of movement

$$s_i = \frac{p_i - p_{i-1}}{\Delta t} \tag{3.2}$$

with Δt being the time between two frames. With 60 frames per second occurring in our environment we can calculate for example $\Delta t = 1000/60 \approx 16.67ms$. Now we have the

speed vector which as a side effect is already pointing at the current direction of movement. With the assumption that maintaining the speed and the direction of movement leads to a good estimation we can define our *predicted* position

$$\hat{p}_i = p_i + l \cdot s_i \tag{3.3}$$

with l being the amount of latency we want to compensate.

Polynomial regression

Another option is to take multiple last detected touch points into calculation. With that approach polynomial regression can be used. The idea behind regression is to find a function of a certain polynomial degree that approximates the correlation between an independent variable x and a dependent variable y best. Therefore we perform polynomial regression twice: with x_i always being the time frame and y_i being first the x and second the y component of a *original* position p_i. For degree $m \in \mathbb{N}$ we structure the data of $n \in \mathbb{N}$ given p_i

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^m \\ 1 & x_2 & x_2^2 & \dots & x_2^m \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^m \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_m \end{bmatrix} \tag{3.4}$$

with one data pair per row. Now we search for coefficients a that minimize the residuals ϵ using the method of ordinary least squares. Illustrating the steps to resolve for the coefficients vector we use the short form:

$$\overrightarrow{y} = X\overrightarrow{a} \tag{3.5}$$

To solve for $\overrightarrow{a}$ we could invert X, but that is not preferable, because it takes much computation time. Therefore better multiply on both sides with the transposed matrix X^T of X to assure a square matrix in order to use LU decomposition:

$$X^T\overrightarrow{y} = X^T X\overrightarrow{a} \tag{3.6}$$

Next split up $X^T X$ to easier to solve lower and upper matrices along with the permutation matrix P:

$$PX^T X = LU \tag{3.7}$$

With that done, first substitute $X^T X \overrightarrow{a}$ in 3.6 to calculate the unknown vector $\overrightarrow{z}$ with forward substitution

$$PX^T \overrightarrow{y} = L\overrightarrow{z} \tag{3.8}$$

and finally solve with backward substitution

$$\overrightarrow{z} = U\overrightarrow{a} \tag{3.9}$$

to get values for $\overrightarrow{a}$. As a result, with these values we get two approximated functions f_x, f_y separately describing the movement of a touch on the x-axis and the y-axis over time. Compensating a latency l at time t we can take the values from $f_x(t+l)$, $f_y(t+l)$ for the *predicted* position.

Measurement of prediction quality

To measure the quality of prediction an obvious method is to compare the predicted position with the then actual detected position. This is, for example, also done by LaValle et al. [2014] evaluating their predictive tracking. We use the mean absolute error (MAE) given in pixels to process this difference between predicted and actual value. If we want to predict $m \in \mathbb{N}$ frames we compare $n \in \mathbb{N}$ actually detected positions p_i with the prediction we made m frames ago $\hat{p}_{i-m}$. The error is then defined as:

$$MAE = \frac{1}{n} \sum_{i=m}^{n} \|\hat{p}_{i-m} - p_i\|_2 \tag{3.10}$$

With that we get an overall estimation about the quality of certain prediction models and can compare them to each other. For further analysis every single absolute error can be checked to get information about outliers.

Separate distance and direction error

Prediction in general is based on two components: the distance to cover and the direction the movement should continue on. For a detailed analysis it can make sense to split up the error calculation looking at each of these mean component errors (MCE) for predicting $m \in \mathbb{N}$ frames separately. In case of the mean distance error given in pixels we want to indicate how well a prediction model handles speed. We are taking the distance between the predicted position m frames ago and the original position m frames ago as well as the distance between the current actual position and the original position m frames ago and then take the difference between them:

$$MCE_{dis} = \frac{1}{n}\sum_{i=m}^{n}(|\|\hat{p}_{i-m} - p_{i-m}\|_2 - \|p_i - p_{i-m}\|_2|) \quad (3.11)$$

For the mean direction error given in degrees we first have to specify how the direction angle is measured at all. With the given coordinate system of the display as reference a straight move to the right should have a direction angle of $0°$, to the top $90°$, to the left $180°$ and to the bottom $-90°$. Taking two positions p_i and p_j we can get the direction angle of a movement from p_i to p_j:

$$\alpha(p_i, p_j) = atan2(y_j - y_i, x_j - x_i)\frac{180}{\pi} \quad (3.12)$$

Now that we can calculate movement direction angles, we further need the difference between two of them. To correctly get it we define:

$$\beta(\alpha_1, \alpha_2) = |(\alpha_1 - \alpha_2 + 180) \bmod 360 - 180| \quad (3.13)$$

The function is built up in a way to always ensure getting the shortest circular difference between the two given angles. With these definitions in mind we can finally calculate the overall mean direction error, which indicates how well a prediction model anticipates direction:

$$MCE_{dir} = \frac{1}{n}\sum_{i=m}^{n}\beta(\alpha(p_{i-m}, \hat{p}_{i-m}), \alpha(p_{i-m}, p_i)) \quad (3.14)$$

The definition of these specific error measurements enables a more precise analysis of the performance of different prediction models. In combination with the later introduced visualization of original and predicted traces we have powerful tools for evaluation.

3.2 Software Framework

Multitouchkit

One motivation using a software approach for latency compensation is its flexibility and independence concerning underlying hardware. To enable an easy inclusion of touch prediction into the already existing environment, it is fully compatible with the Multitouchkit (MTK). The MTK provides a framework for working with tangibles on multi-touch-displays. Written in Objective-C it supports Mac OS and iOS out of the box. It is developed by Simon Voelker and fully reworked by Linden [2015], in future it is planned to rewrite it using Swift. For our work it is important to first get to know how the MTK is structured with a special focus on its touch processing.

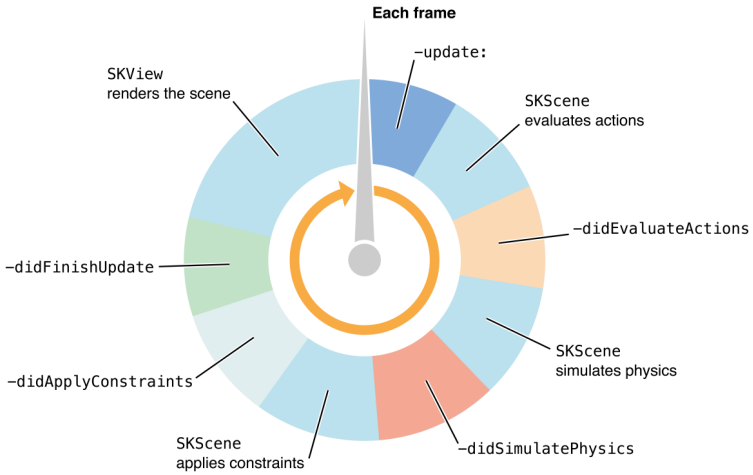


Figure 3.1: Rendering loop of SpriteKit taken from the SpriteKit programming guide by Apple [1]

Rendering loop

In general the MTK is based on SpriteKit by Apple, it also uses its rendering loop shown in figure 3.1. Before anything on the scene is rendered, at first with `update:` touch processing is performed. In the MTK several steps are executed every time this method is called. Most important for touch prediction is, that after all touches are recognized, `preProcess:` is called. This global delegate allows a

[1] https://developer.apple.com/library/content/documentation/GraphicsAnimation/Conceptual/SpriteKit_PG/Introduction/Introduction.html

`MTKScene` to manipulate touches before anything else is done with them. So this is the appropriate point in the rendering loop to execute the prediction. After this call, among others, the position of tangibles is updated and can therefore just adapt the predicted position.

Representation of touches

The continuous movement of a touch over time is called a trace and saved in an instance of `MTKTrace`. In order to a clear assignment, every trace gets an unique identifier provided by the MTK. Touch input can be obtained from different sources: supported are mouse and UITouch as well as TUIO and JSON via network. Therefore, the input source can provide optional information about its name and type, which can be stored in the trace, too. To help handling traces, each one has a current state expressed through Begin, Move or End. `MTKEntry` is responsible for saving a single touch of a trace. Single touch entries have to be specified by a timestamp and position and can contain information about size and orientation useful for a possible cursor. Concerning memory management whole traces can be compressed, deleting their touch entries while keeping the other information.

Tangibles in the MTK

With their markers tangibles create three touch points for an exact detection of their position and rotation. Their marker distances and angles in relation to each other can be specified creating a tangible model to associate certain patterns with certain tangibles. For a correct detection all three created traces have to be found. During movement, it can be possible that traces are lost and reappear after a while. Once the pattern has been detected properly, two detected markers are sufficient to calculate its position and rotation. Additionally, the MTK continuously scans for new traces to associate them with a matching tangible again. Furthermore and very useful for visualizing the problem of latency, a tangible can have a customizable virtual representation of its last detected position on the screen.

3.3 Implementation

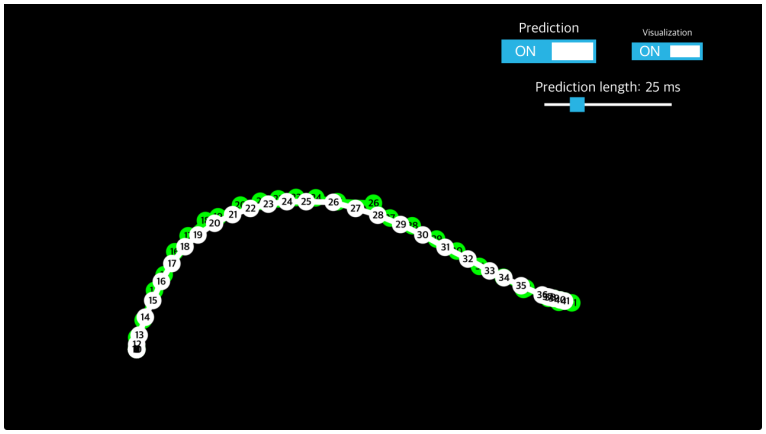


Figure 3.2: Basic scene to show effects of prediction

Touch prediction is implemented providing multiple features which will be explained in detail in this section:

- Full compatibility with the MTK for comfortable inclusion into the already existing environment, working with Mac and iOS
- Example scene including UI elements to enable prediction, set prediction length and optionally activate real time visualization, screenshot in figure 3.2
- Manipulation of touches based on a chosen prediction model
  - Framework administrates prediction relevant information of traces like original position, speed and direction angle
  - Already implemented models are short-term linear prediction as presented by Cattan et al. [2015b] and polynomial regression with variable buffer length
- Persistently save trace information to file
- Run benchmark on saved trace comparing different models and parameters as well as visualize original and predicted trace

3.3.1 Class Structure

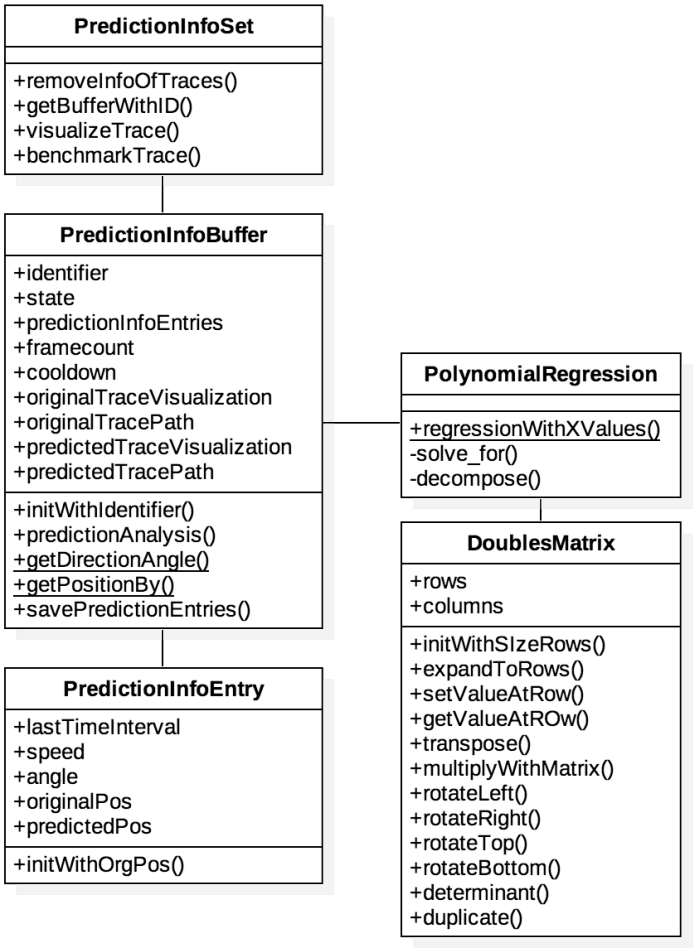


Figure 3.3: Overview on touch prediction classes including their instance variables and provided functions

The new classes for touch prediction are ordered similar to the already established classes handling touches in general. `PredictionInfoSet` is a set containing all necessary information about current traces regarding prediction. This information is stored in one `PredictionInfoBuffer` instance for each trace. Among others, the buffer includes an array filled with elements of the type `PredictionInfoEntry`, which handle data describing single touches.

PredictionInfoEntry

Starting with the lowest class in hierarchy, a `PredictionInfoEntry` instance is created for every touch registered by the MTK. Most important, it has to store the detected touch position. This is done, because the prediction always relies on information about past original positions before any manipulation. As these positions in the correlating `MTKEntry` have to be overwritten with the predicted positions to be globally recognized they are saved here separately. Along with this, other useful information like predicted position, speed and direction angle of movement are stored in this class. The two latter values are used for short-term linear prediction while the predicted position is saved for an easy comparison to the original position during benchmark. Of course the class can hold additional properties if needed for different prediction models.

PredictionInfoBuffer

The `PredictionInfoBuffer` holds an array containing the recent touch prediction entries. The amount of stored entries can be influenced by changing the buffer length. The buffer length sets at which amount of entries the oldest entry of the array is overwritten to not affect prediction anymore. With the purpose of saving the whole trace, overwriting is not activated during trace recording. Furthermore, the buffer has an unique identifier and state, which equal the ones used for the associated `MTKTrace`. For visualizing a trace, our buffer saves information about the current frame count since the beginning of the trace. Nodes displaying the original and predicted trace on a scene are saved here, too. Concerning methods, the actual `predictionAnalysis:` is part of this class as it is performed on the buffer data. In order to predict, speed and direction of movement are calculated using two supporting functions. First the direction angle of movement in degrees given two points can be obtained. Second a movement angle, a starting point and the expected distance to cover are sufficient to derive a new position. The possibility to save an array of entries is included in this class as well.

PredictionInfoSet

All active `PredictionInfoBuffer` are part of the `PredictionInfoSet`. The class enables to search for a buffer with a given identifier and provides functions to benchmark and visualize it. `visualizeTrace:` has to be

provided with the trace to visualize, the requested size and the scene it should be displayed on. With this information it can display every single detected touch as a small circle filled with an ascending number of the frame the touch was recognized in. All the circles are connected to visualize the trace. Because this can be done similarly for predicted and original position just using different colors, you can quickly compare their positions. `benchmarkTrace:` is executed with a given file name providing the raw data. Additionally, all relevant parameters can be set: How many frames should be predicted? Which model should be used with which buffer length? If the model uses polynomial regression its degree can be specified. If the trace should be visualized a scene to visualize on is needed. With all these pieces of information the trace can be reconstructed again part by part to perform prediction like on real time data. The benchmark can return the mean absolute error (definition 3.10), the mean distance error (definition 3.11) or the mean direction error (definition 3.14).

Polynomial regression

Objective-C itself provides no standard library for performing polynomial regression. Therefore, an implementation developed by Gilles Lesire [2] and published under GNU General Public License is first tested and then embedded into the project. Basically it follows the steps described in the theory section. Given two arrays with x and y values as well as the wanted polynomial degree it outputs an array with the coefficients of the approximated function. To be capable of that a representation of a matrix containing Double elements is developed by him as well. Included are necessary supporting functions to expand, multiply and transpose these matrices.

3.3.2 General Process

Let us have a process-oriented look at what is done with incoming original positions to derive any prediction. All new touch points are gathered in the `preProcess:`. Depending on the state of a trace the following procedure differs.

[2] https://github.com/KingIsulgard/iOS-Polynomial-Regression

Beginning Trace

In case of a beginning trace, no prediction is executed because at this point obviously not enough data is available. Instead the prediction framework for this particular trace is initialized. The first position and the identifier of the trace are copied to a new `PredictionInfoBuffer`. If activated, the visualization is also initialized and adds a small circle at the first position to indicate the touch.

Moving Trace

When handling a moving trace things differ: the prediction buffer of the trace already exists and can be updated with a new original position detected at this frame. Furthermore, prediction analysis is executed. Here depending on the chosen model different calculations are performed. If linear short-term prediction is used, the prediction is based only on the last and current position. To predict, the direction angle of movement and its current speed are calculated, so that a new corrected position can be derived. Otherwise, polynomial regression including the whole buffer history is executed. In any case the `predictionAnalysis:` returns a predicted position which is then set in the original `MTKEntry` to be globally used now. Visualization additionally draws a path from the last to the current position and indicates the new touch with a circle and frame count. This is done with the original and predicted positions in parallel using different colors enabling an easy comparison of the two on screen.

Ending Trace

Dealing with an ending trace, it optionally can be saved first. Afterwards, if no visualization is active all prediction information concerning the particular trace is erased. Otherwise, the data stays in storage as the trace visualization remains on the screen to be visible even after the touch has ended. Only at the time a new trace begins it will be erased with all its information to have a clear start with enough space on the display for a new trace visualization.

Benchmark

A benchmark cannot use `preProcess:` itself as it is performed on prerecorded data read from a file instead of actually detected positions on a display in real time. This file contains an array of `PredictionInfoEntry` instances including among others their captured original positions and timestamps in chronological order. To execute prediction as on real time data, this array works as a template

for procedurally rebuilding the trace just as it would be detected in `preProcess:`. It is first initialized and then step by step for each recorded frame the associated already recorded entry is added, so `predictionAnalysis:` and `visualizeTrace:` can work with the data without modifications. Simultaneously, the desired error measurements are executed and at the end returned to estimate the quality of prediction.

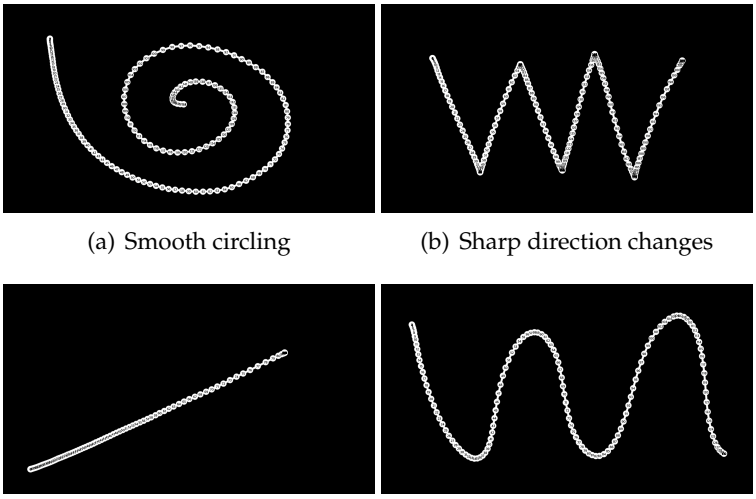
Chapter 4

Evaluation

Overview

Evaluation of touch prediction is performed in two steps. Initially four captured exemplary traces are used to test the developed models and their parameter configurations. This is done to get a first idea about how they behave in certain scenarios and point out their strengths and weaknesses. Additionally, captured input of air hockey game play is used to test how prediction models perform on it. This bridges the gap to the second evaluation step consisting of a user study. By taking up the motivation given in the introduction, participants should play two games of air hockey with tangibles as mallets: one game without prediction, the other one with the model that performed best during benchmark on the air hockey trace. Users are asked which game they overall prefer along with more detailed questions on how they perceive the connection between physical mallet and virtual puck. This should give an idea whether prediction leads to an improvement in this challenging application or not.

4.1 Benchmarks



(a) Smooth circling (b) Sharp direction changes

(c) Straight line (d) Curves

Figure 4.1: Characteristic traces to benchmark on

Benchmark on captured traces

As presented in the last chapter, the implementation of touch prediction enables customization by offering different models with variable parameters. A first look at the prediction quality is taken by analyzing how these parameters influence the mean absolute error (MAE) calculated as defined in equation 3.10. In our case the MAE is given in pixels and can also be described as the mean euclidean distance between predicted and actual position, overall a lower score means better performance. Independent variables we want to analyze are prediction length and prediction model with further configurations on polynomial degree and buffer length. All tests are based on prerecorded data sets and therefore can be easily reproduced and extended. The traces are captured on a 55 inch Full HD capacitive multi-touch-display. With its 40 pixel per inch therefore 10px equal 6.35mm on this particular screen.

4.1.1 Prediction Length

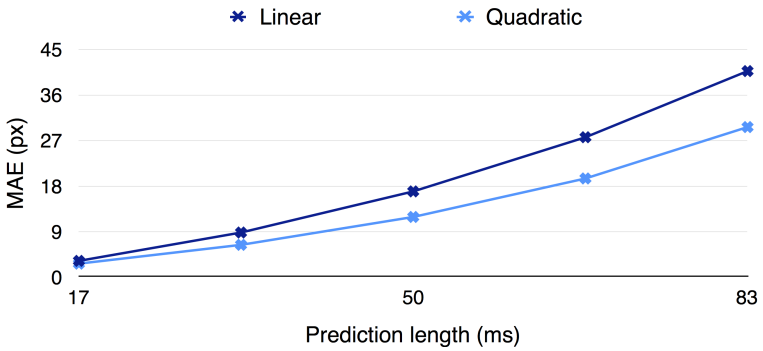


Figure 4.2: Development of MAE over all example traces in dependence of prediction length

MAE grows beyond linear with increasing prediction length

Before we start any detailed benchmarks one important question is which prediction length to choose. It goes without saying that the longer you look into future the more uncertain your prediction becomes, so a linear scaling is not expected. In order to verify this assumption we use the linear model as well as a quadratic model with a buffer length of five to predict one to five frames (with 60 Hz detection rate resulting in a prediction time of roughly 17ms to 83ms). The calculated errors of both models are averaged over all example traces seen in figure 4.1. Results are displayed in figure 4.2 and show that the error in both cases is not only increasing linearly with a greater prediction length. For example going from one to two frames of linear prediction, the MAE increases by a factor of 2.62, doubling the prediction length from two to four frames increases the error by a factor of 3.12. This confirms that doubling the prediction length not only doubles the overall error as uncertainty increases. When frequent changes of movement speed and direction are expected a shorter prediction length should be preferred. In that case the prediction error in general is expected to be higher and the beyond linear scaling makes longer predictions even worse. In contrast to this, an application with mostly smooth movements and a therefore expected smaller prediction error could allow for an increased prediction length. Overall the prediction length should only carefully be increased while always taking the actual use case into account.

4.1.2 Degree and Buffer Length of Polynomial Regression

Procedure

Now we compare the MAE for different buffer lengths and polynomial degrees predicting 50ms (three frames with a 60 Hz detection rate) using again the captured exemplary traces shown in figure 4.1. The buffer length specifies how many past values are taken into calculation for the approximated function describing the trace. The general form of this function is specified by the polynomial degree. Because a function of degree n needs at least $n + 1$ points to be defined, the lowest buffer length is fixed for each of the three polynomial degrees. At maximum we want to include data of 30 frames, so for every configuration the benchmark starts after 30 detected positions as from this point on every configuration has enough data available for prediction. Notice that we cannot compare to the actual tangible position and therefore only compare the models to each other, because we do not know the exact overall system latency. But it can be said that concerning the MAE a good prediction leads to a much lower error than no prediction at all. If we just compare every position with the one three frames ago we get a MAE of 74.16px between both, significantly higher than the results we achieve with a good prediction.

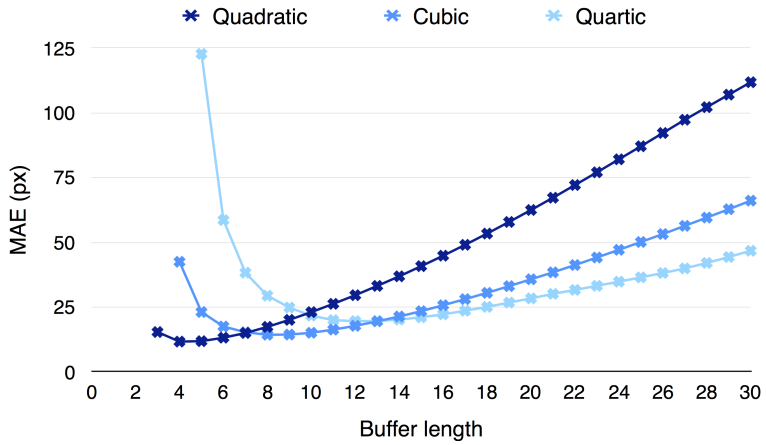


Figure 4.3: MAE of prediction executed on all example traces using quadratic, cubic and quartic regression with variable buffer length

Results and discussion

The results displayed in figure 4.3 show the averaged MAE on all four traces with the different configurations. Overall quadratic regression with a buffer length of four performed best resulting in a MAE of 11.69px. For cubic regression a buffer length of eight is needed for a result of 14.33px, quartic regression already needs twelve detected positions for a MAE of 19.60px. It can be seen that with increasing polynomial degree more values are needed to find a good approximated function. However, in general cubic and quartic perform worse than quadratic regression, not reaching its lowest MAE. An additional look into the detailed test results for each particular trace show that using cubic and quartic regression do not lead to a lower prediction error on any single trace. Concluding the movement can be described best by a quadratic model, cubic models and anything beyond are not appropriate to approximate these short movement intervals. Therefore the upcoming benchmarks will solely focus on quadratic regression.

4.1.3 Quadratic Prediction vs Linear Prediction

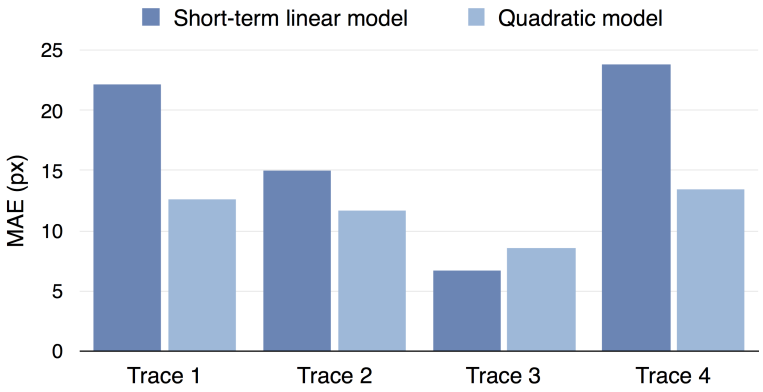


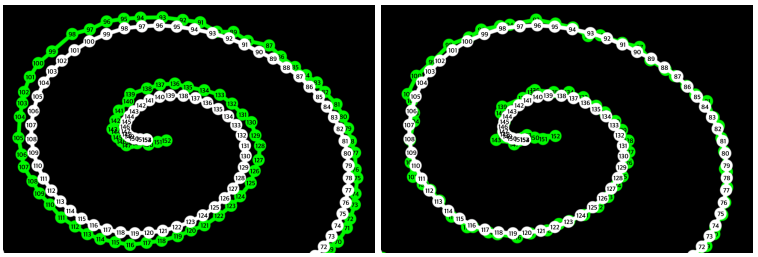
Figure 4.4: MAE of prediction comparing short-term linear model against quadratic model on each trace

Procedure

Next we directly compare short-term linear prediction against the best quadratic prediction again predicting three frames on each particular trace. The linear model always only takes the current and the previous position into calculation whereas with the quadratic model the optimal buffer length differs.

Results and Discussion

The results can be seen in figure 4.4. For the first example containing steady circular movement quadratic regression with a buffer length of five performs best. The sharp direction changes of the second example trace are predicted best with quadratic regression and a shorter buffer length of four. On the third trace following a straight line short-term linear prediction works best, over-performing the best quadratic model. Finally the curve movement of the last example trace can be anticipated best by quadratic regression again with a buffer length of four. All in all, the best model seems to be dependent on the characteristics of the trace, a universal solution does not exist. Therefore a more detailed analysis is worth the effort.



(a) Linear prediction (b) Quadratic prediction

Figure 4.5: Visualization of prediction models handling curves showing the white original trace and the green predicted trace

Curve handling characteristics

Figure 4.5 shows how linear and quadratic prediction anticipate curves. The linear model produces a steady offset as it always assumes a straight movement. The quadratic model on the other hand can predict the smooth curve almost perfectly allowing the predicted green trace to closely follow the original white trace. If we take a closer look at the mean distance error (see equation 3.11) and the mean direction error (see equation 3.14), they support what can be seen on the visualization. On the first example trace quadratic prediction overall works best although it produces a mean distance error of 10.53px which is higher than the 6.20px produced with linear prediction. But in contrast the mean direction error with the quadratic model is significantly lower than with the linear model, 2.94° instead of 12.50°. This proves that the better overall result of the quadratic model

on the first example trace is solely caused by its anticipation of the curve not the speed. By that a strong recommendation for the quadratic model can be made in case the input is expected to contain of mostly smooth curves.

Speed handling characteristics

Nevertheless, the linear approach has its advantages on straight lines and is a bit more reactive when dealing with sudden changes. This could be explained by the amount of last positions the models take into account. The short-term linear model always only takes the last two positions, therefore every new position with its speed and direction has a high impact on the prediction. If multiple last positions are used, every new position is only partly influencing the new prediction. This has a regularizing effect but it also could make the system less reactive. If we look at the anticipation of speed changes the simple but more reactive linear model can handle the abrupt stop in the third trace better having a slightly smaller mean distance error of 6.41px versus 7.70px.

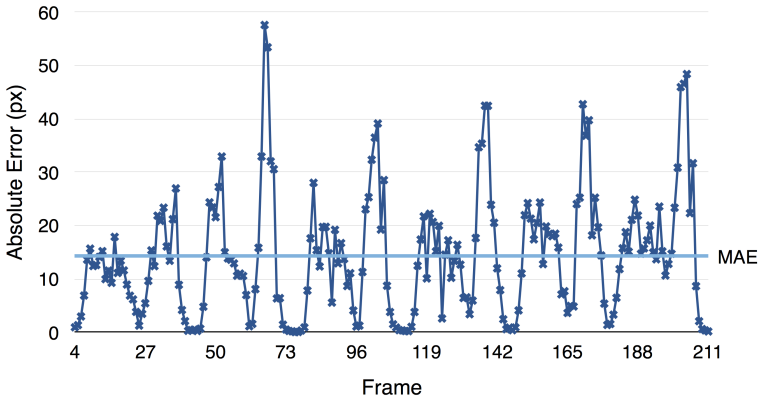


Figure 4.6: Absolute error over time with the linear model predicting three frames on the second example trace

Where prediction has problems

Besides characteristic strengths and weaknesses of the models it generally can be said that the more extreme changes occur in short time the harder it becomes to predict the movement. This can also be seen in figure 4.6 showing the peaks of the absolute error at every sharp direction change of the second example trace using linear prediction. In contrast the error is almost zero during the short movement parts with constant speed on a straight line.

4.1.4 Application Air Hockey

Analyze air hockey game play

In this section we compare the best models on traces captured while playing air hockey with tangibles as mallets under the same condition the later user study will be executed on. Figure 4.7 visualizes the captured movement of a player. Typical straight moves from the center to the puck can be detected as well as curvy backwards movements. The movement pattern here already shows that the use case is challenging including sharp direction and speed changes. Therefore and supported by our own experience during test matches, we decide to set the prediction length to 33ms (two frames with 60 Hz detection rate) as this seems to be the sweet spot between not causing any recognizable change with a very short prediction length and causing false predictions and jitter with a too long prediction length. For the benchmark we use captured traces with a total of more than 12,000 positions to analyze. For comparison the linear as well as the quadratic model with a buffer length of four to seven are chosen as they performed best on the exemplary traces before.

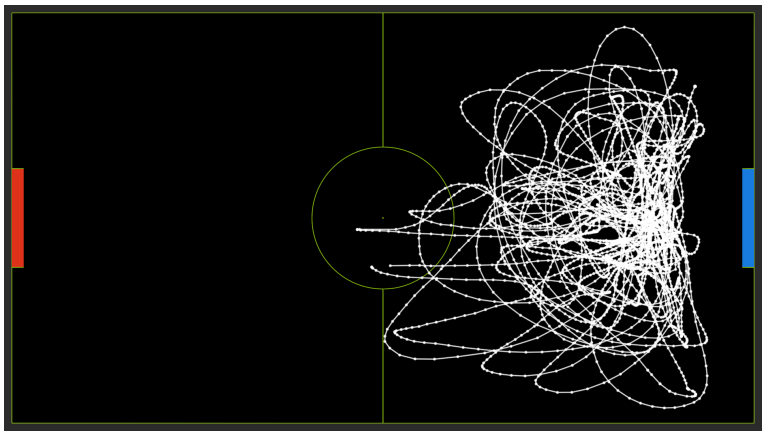
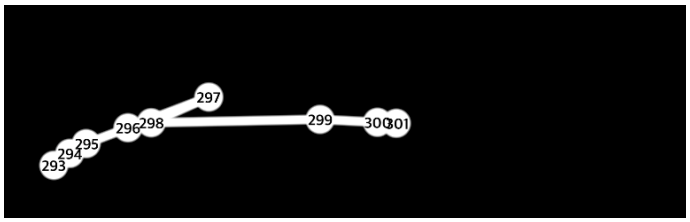


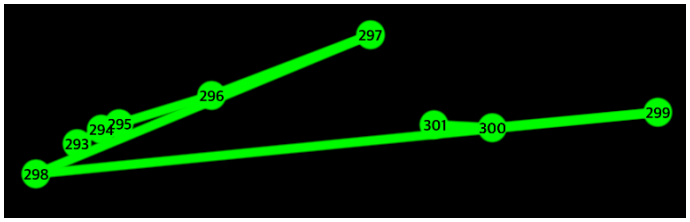
Figure 4.7: Captured air hockey trace

Results

Results show that the short-term linear model with a MAE of 6.57px performed best. Following is the quadratic model with a buffer length of five and a result of 7.22px. Therefore we are choosing the linear model for our user study.



(a) Glitch mixes up detection order of original input



(b) Unmodified prediction extrapolates the error

Figure 4.8: Excerpt of trace captured on multi-touch-display including glitch

Dealing with glitches

Playing over a longer period on the display that will be used in the study highlights some problems to be tackled before the study. Seldom input of one frame is first skipped and then only detected delayed or traces are temporally lost. Skipping a frame can be seen in figure 4.8. Glitches like that are very harmful to the users' perception of game play as they are extrapolated by prediction and lead to unexpected extreme behavior of the mallet especially concerning collision detection. Therefore they have to be detected and filtered out. To be capable of that, speed and direction of movement are permanently observed. If their changes from one frame to another are above a defined threshold, prediction is stopped for the next three frames. This avoids the extrapolation of these glitches. Unfortunately, defining the threshold is always a trade off between stopping prediction unnecessarily often and not detecting all extreme glitches. For the upcoming user study the filter is set in a way to ensure filtering out most extreme behavior, even if it overall leads to a slightly higher MAE because of stopping prediction a bit too often.

4.2 User Study

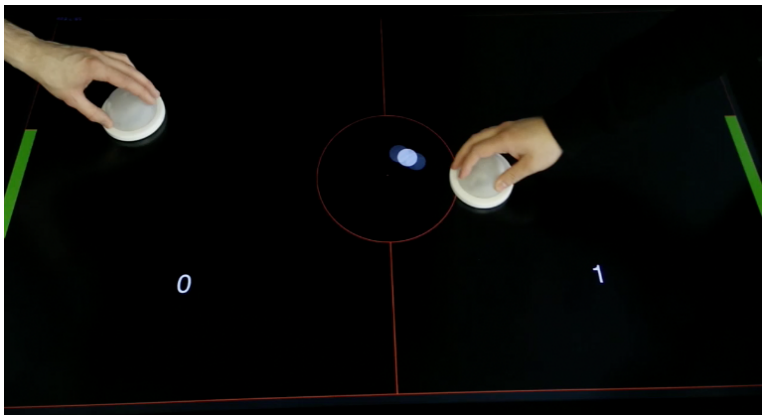


Figure 4.9: Two people playing air hockey, picture taken by Cherek et al. [2015]

The user study is performed to find out how people perceive the impact of touch prediction while playing air hockey as shown in figure 4.9. The specific application is chosen because latency is highly noticeable while playing this game due to the required fast movements to hit the puck. This is also supported by the already shown movement pattern in figure 4.7. Consequently there exists a large potential benefit with compensated latency in this scenario while the prediction model also faces a major challenge because of the frequent speed and direction changes. Because of that it is very interesting to analyze this particular application.

4.2.1 Method

Hypothesis

With touch prediction we are aiming at reducing the perceived latency to increase the overall responsiveness of the application. Therefore we are expecting to validate the following hypothesis with our user study: *Users prefer playing with touch prediction*

Participants

14 people (2 female) between 20 and 31 with a mean age of 24 take part in the study. All participants come from an academic background. Exactly half of the people have played air hockey before.

Hardware

Two PERCs as mallets are used. The game is executed on an iMac (Intel Core i7, NVIDIA GeForce GTX 780M, 24GB RAM) and displayed on a 55 inch capacitive multi-touch-display by Microsoft. The display has a resolution of 1920×1080 and a touch detection and display refresh rate of 60 Hz.

Procedure

Participants are asked to play two matches of air hockey, one with and one without touch prediction. In order to predict 33ms of movement the short-term linear model is used with an additional filter to detect glitches. Before participants start their first match they are shortly instructed to keep an eye on their perception of mallet control, the reactivity of the game and the collision detection between mallet and puck. Besides that, no additional information is given, especially not that the perceived latency between the games will differ. Any virtual representation of the detected tangibles is completely disabled. Every match has a duration of five minutes, participants are given the possibility to pause the game at any time if they want to. To avoid possible influences half of the participants start with activated touch prediction, the other half with deactivated touch prediction.

Questionnaire

After each match the participants have to fill out a short questionnaire containing five statements. These include their perception of mallet control (1), immediate movement detection (2), accurate collision detection (3), missing the puck even with a quick enough reaction (4) and an appropriate impact of the mallet's speed and movement on the puck (5). The level of agreement to each statement can be expressed on a Likert scale offering five options. Additionally, a comment section is provided. After finishing both matches participants should declare which game they preferred. The original questionnaire can be seen in the appendix.

4.2.2 Results and Discussion

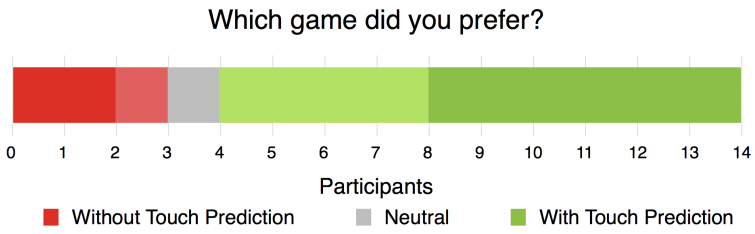


Figure 4.10: Proportional distribution of participants' game preference

	Average		Median		SD	
Statement	Org	Pred	Org	Pred	Org	Pred
1	3.64	3.64	4	4	1.11	1.29
2	3.14	3.64	3.5	4	1.12	1.23
3	3	3.07	3	3	1	1.22
4	3.93	3.43	4	3.5	0.80	1.29
5	3.79	3.57	4	4	0.94	1.29

Table 4.1: Results of detail questions including all participants

Overall preference

The distribution of game preferences is visualized in figure 4.10. Ten participants totally or slightly preferred the game with prediction, one had no preference and three totally or slightly preferred the game without prediction.

Approval concerning statements

Results of the detail questionnaires are presented in table 4.1. Concerning arithmetic mean and median a value of 1 stands for total disagreement, whereas a value of 5 represents total agreement regarding the given statement. Additionally the standard deviation of the answers is given to indicate the amount of variation. With prediction its value is higher on all five statements. If we take answers of all participants into account only small tendencies concerning their answers on questions 2, 3, 4 and 5 are detected. With prediction participants expressed a stronger approval concerning the statements of an immediate movement detection and an accurate collision detection and a weaker approval concerning the statements of not hitting the puck even when having reacted quickly enough and an appropriate transfer of mallet speed and direction to the puck.

The raw data with additional bar charts can be found in the appendix. There can also be seen that the tendencies concerning statements 2,3 and 4 are increased when only taking the participants into account that overall favored touch prediction.

Comments of participants

In the comments people mention that especially side movements were not detected in time when playing the original game and that they felt an improvement in the game with prediction. Without prediction the increased lag was recognized. With prediction some people complained that weak hits of the puck were interpreted as strong ones.

Discussion of detail questions

The increased standard deviation indicates that the answers are more spread with prediction and not represented well by the arithmetic mean value. This could explain why we only detect small tendencies concerning the detailed questions while a larger majority overall preferred prediction. Participants' perception of touch prediction was strongly connected to their playing style. We observed matches between two participants where at the end one person preferred prediction while the other person preferred the original game play. People that tended to play more extreme most of the time just trying to hit the puck with as much speed as possible were the ones favoring original game play because prediction had problems handling these extreme changes of speed. This potentially weakened their experience and caused a strong approval towards the original game play reflected in their answers. Maybe a decrease of the puck bounciness or an overall cap of its speed could help here. The results of the people that preferred touch prediction show that they favored it because they felt a more immediate movement detection as well as a more accurate collision detection. Additionally for these participants the occurrences of not hitting the puck were reduced, which is also supported by some comments.

Discussion of comments

The mentioned lag during side movements was a problem with latency as the exact position here made a difference if the puck was hit at all or completely missed, so compensated latency made a perceptible difference. Hitting the puck during forward movement was not that latency sensitive as the puck was just sliding under the tangible but still was hit.

Conclusion

To come to a conclusion, a significant majority of people preferred the match with reduced perceived latency. Therefore our hypothesis is validated. Nevertheless, prediction in combination with air hockey can be further improved to handle even more playing styles.

Chapter 5

Summary and Future Work

5.1 Summary and Contributions

Summary

The thesis was located in the area of tangibles tackling the problem of latency with a software approach. By referring to related work we pointed out that latency can have a strong influence on performance. Especially with dragging tasks as performed with tangibles a lagging detection harms the system responsiveness and applications like air hockey suffer from that. To address the problem first the existing environment provided by the MTK had to be understood to then fully integrate touch prediction into it. With that done we evaluated our prediction considering the absolute euclidean distance between predicted and actual position as well as detailed distance and direction errors. The ability to capture traces was required to afterwards run different models on the same data for an accurate comparison. Furthermore, original and predicted traces were visualized to illustrate the behavior of different prediction models. Finally we took up the initial thesis motivation and performed a user study to get to know how participants perceive the effects of touch prediction while playing air hockey.

Contributions

Our developed touch prediction gathers the original touch input to execute different prediction models on the data in real time. The software does not require additional hardware tweaks and is designed to be compatible with iOS and Mac. Reducing the perceived latency it works with tangibles as well as direct touch. It includes the linear approach by Cattan et al. [2015a] continuously calculating speed and direction out of the last two positions and extrapolating this movement for prediction. Additionally, the framework is extended using polynomial regression with a variable degree and buffer length. Our detailed evaluation allows us to give recommendations on which prediction model to choose under what circumstances. Quadratic prediction works best with a short buffer length of four on input containing smooth curves. Using cubic or quartic regression overall causes a higher prediction error and is therefore not recommended. Short-term linear prediction handles straight lines with abrupt speed changes best. Independent of the chosen model an increased prediction length causes a beyond linear increase of the prediction error, so we advise to only carefully increase the prediction length. The results of our user study indicate that touch prediction is a promising approach, because ten out of 14 participants favored the air hockey game with activated touch prediction.

5.2 Future Work

New displays

With the spread of displays with higher refresh rates like the Microsoft Surface Hub significant hardware improvements are made. With its 4K resolution and most importantly a 120 Hz refresh rate touch prediction on this display can potentially be more precise. This assumption can be made, because with the increased refresh rate we can get twice as much values in the same time to improve quadratic prediction or shorten our time intervals even more with the short-term linear model to react faster to sudden changes.

Performance improvements

At the moment every recognized trace on the scene is processed. If a tangible is used it creates three traces. Therefore the amount of traces to be analyzed can become quite large. The current implementation has been tested with

maximum two tangibles simultaneously while playing air hockey. For better performance the implementation of prediction could be changed to just take the center point of a tangible for prediction. This would decrease the processing effort significantly, but would need additional changes to handle for example tangible rotation.

New prediction models

Prediction quality could be improved through preventing extreme predictions. These extreme predictions could be recognized by setting a maximum prediction distance based on the cap of executable speed by humans. The movement speed while conducting has been analyzed by Koster [2008] and could motivate such an analysis for tangible interaction. All in all reducing latency by software is a promising approach where research continues. Henze et al. [2016], for example, have a paper to release that is dealing with software-reduced touch screen latency comparing linear and polynomial prediction to an approach using neural networks. When new approaches or prediction models like that are explored, the current implementation is open for further extensions.

Appendix A

User Study Questionnaire and Results

ID: _____

First Match

Gender _____
Age _____
Air Hockey Experience? _____

1. I have full control over my mallet

Totally disagree		Neither nor		Totally agree
◯	◯	◯	◯	◯

2. My movements are detected immediately

Totally disagree		Neither nor		Totally agree
◯	◯	◯	◯	◯

3. The collision detection between mallet and puck is accurate

Totally disagree		Neither nor		Totally agree
◯	◯	◯	◯	◯

4. Sometimes I do not hit the puck even if it feels like having reacted quickly enough

Totally disagree		Neither nor		Totally agree
◯	◯	◯	◯	◯

5. Speed and direction of movement of the mallet have an appropriate impact on the puck movement

Totally disagree		Neither nor		Totally agree
◯	◯	◯	◯	◯

Comments?

ID: _____

Second Match

1. I have full control over the mallet

Totally disagree		Neither nor		Totally agree
◯	◯	◯	◯	◯

2. My movements are detected immediately

Totally disagree		Neither nor		Totally agree
◯	◯	◯	◯	◯

3. The collision detection between mallet and puck is accurate

Totally disagree		Neither nor		Totally agree
◯	◯	◯	◯	◯

4. Sometimes I do not hit the puck even if it feels like having reacted quickly enough

Totally disagree		Neither nor		Totally agree
◯	◯	◯	◯	◯

5. Speed and direction of movement of the mallet have an appropriate impact on the puck movement

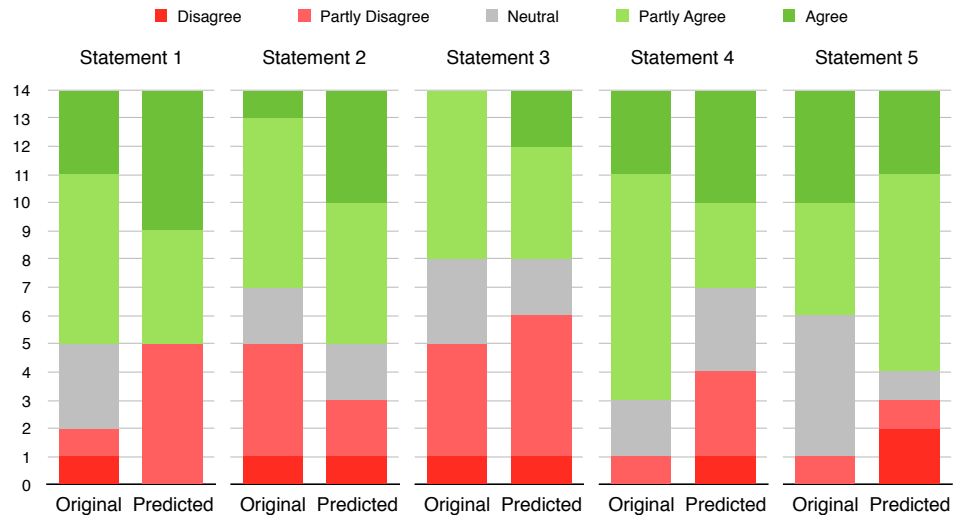
Totally disagree		Neither nor		Totally agree
◯	◯	◯	◯	◯

Comments?

Concluding: Regarding the asked aspects, which match did you prefer?

The first match		No differences		The second match
◯	◯	◯	◯	◯

ID	Gender	Age	AH Experience	Statement 1		Statement 2		Statement 3		Statement 4		Statement 5		Prediction prefered?
				Org	Pred	Org	Pred	Org	Pred	Org	Pred	Org	Pred	
42	M	20	No	5	5	4	5	4	4	5	5	5	5	**5**
43	M	31	Yes	4	5	4	4	4	5	3	4	4	4	**4**
44	F	28	Yes	4	2	4	2	4	2	4	2	5	4	**1**
45	M	23	Yes	5	5	4	4	3	4	4	2	3	1	**4**
46	M	28	No	4	4	2	3	2	3	5	4	3	3	**5**
47	M	23	No	3	2	3	1	4	2	2	5	4	2	**1**
48	M	23	Yes	4	4	2	5	3	2	4	3	3	4	**4**
141	M	22	Yes	3	5	2	4	2	2	4	2	4	4	**5**
142	M	22	No	4	4	4	4	3	3	4	3	3	4	**4**
143	M	21	Yes	4	4	5	5	2	4	3	3	3	4	**5**
144	M	24	No	5	5	4	5	4	5	5	1	5	5	**5**
145	M	25	Yes	1	2	3	3	1	2	4	5	2	5	**5**
146	F	26	No	2	2	2	4	4	4	4	4	5	4	**2**
147	M	22	No	3	2	1	2	2	1	4	5	4	1	**3**
Overall average				3.64	3.64	3.14	3.64	3.00	3.07	3.93	3.43	3.79	3.57	3.79
Average of participants who prefered prediction				3.90	4.30	3.40	4.20	2.80	3.40	4.10	3.20	3.50	3.90	
Overall Median				4.00	4.00	3.50	4.00	3.00	3.00	4.00	3.50	4.00	4.00	
Median of participants who prefered prediction				4.00	4.50	4.00	4.00	3.00	3.50	4.00	3.00	3.00	4.00	
Overall standard deviation				1.11	1.29	1.12	1.23	1.00	1.22	0.80	1.29	0.94	1.29	



Bibliography

Glen Anderson, Rina Doherty, and Subhashini Ganapathy. *Design, User Experience, and Usability. Theory, Methods, Tools and Practice: First International Conference, DUXU 2011, Held as Part of HCI International 2011, Orlando, FL, USA, July 9-14, 2011, Proceedings, Part I*, chapter User Perception of Touch Screen Latency, pages 195–202. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-21675-6. doi: 10.1007/978-3-642-21675-6_23. URL http://dx.doi.org/10.1007/978-3-642-21675-6_23.

Michelle Annett, Albert Ng, Paul Dietz, Walter F. Bischof, and Anoop Gupta. How low should we go?: Understanding the perception of latency while inking. In *Proceedings of Graphics Interface 2014*, GI '14, pages 167–174, Toronto, Ont., Canada, Canada, 2014. Canadian Information Processing Society. ISBN 978-1-4822-6003-8. URL http://dl.acm.org/citation.cfm?id=2619648.2619677.

Takeshi Asano, Ehud Sharlin, Yoshifumi Kitamura, Kazuki Takashima, and Fumio Kishino. Predictive interaction using the delphian desktop. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*, UIST '05, pages 133–141, New York, NY, USA, 2005. ACM. ISBN 1-59593-271-2. doi: 10.1145/1095034.1095058. URL http://doi.acm.org/10.1145/1095034.1095058.

François Bérard and Renaud Blanch. Two touch system latency estimators: High accuracy and low overhead. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces*, ITS '13, pages 241–250, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2271-3. doi: 10.1145/2512349.2512796. URL http://doi.acm.org/10.1145/2512349.2512796.

Pradipta Biswas, Gokcen Aslan Aydemir, Pat Langdon, and Simon Godsill. *Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data: Third International Workshop, HCI-KDD 2013, Held at SouthCHI 2013, Maribor, Slovenia, July 1-3, 2013. Proceedings*, chapter Intent Recognition Using Neural Networks and Kalman Filters, pages 112–123. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-39146-0. doi: 10.1007/978-3-642-39146-0_11. URL http://dx.doi.org/10.1007/978-3-642-39146-0_11.

Elie Cattan, Amélie Rochet-Capellan, and François Bérard. A predictive approach for an end-to-end touch-latency measurement. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces*, ITS '15, pages 215–218, New York, NY, USA, 2015a. ACM. ISBN 978-1-4503-3899-8. doi: 10.1145/2817721.2817747. URL http://doi.acm.org/10.1145/2817721.2817747.

Elie Cattan, Amélie Rochet-Capellan, Pascal Perrier, and François Bérard. 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*, ITS '15, pages 205–214, New York, NY, USA, 2015b. ACM. ISBN 978-1-4503-3899-8. doi: 10.1145/2817721.2817736. URL http://doi.acm.org/10.1145/2817721.2817736.

Christian Cherek, Simon Voelker, Jan Thar, Rene Linden, Florian Busch, and Jan Borchers. PERCs demo: Persistently trackable tangibles on capacitive multi-touch displays. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces*, ITS '15, pages 389–392, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3899-8. doi: 10.1145/2817721.2823474. URL http://doi.acm.org/10.1145/2817721.2823474.

S. Mealla Cincuegrani, S. Jordà, and A. Väljamäe. Physiopucks: Increasing user motivation by combining tangible and implicit physiological interaction. *ACM Trans. Comput.-Hum. Interact.*, 23(1):4:1–4:22, February 2016. ISSN 1073-0516. doi: 10.1145/2838732. URL http://doi.acm.org/10.1145/2838732.

Jonathan Deber, Ricardo Jota, Clifton Forlines, and Daniel

Wigdor. How much faster is fast enough?: User perception of latency and latency improvements in direct and indirect touch. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 1827–1836, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3145-6. doi: 10.1145/2702123.2702300. URL http://doi.acm.org/10.1145/2702123.2702300.

Niels Henze, Markus Funk, and Alireza Sahami. Software-reduced touchscreen latency. 2016.

Zenja Ivkovic, Ian Stavness, Carl Gutwin, and Steven Sutcliffe. Quantifying and mitigating the negative effects of local latencies on aiming in 3D shooter games. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 135–144, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3145-6. doi: 10.1145/2702123.2702432. URL http://doi.acm.org/10.1145/2702123.2702432.

Ricardo Jota, Albert Ng, Paul Dietz, and Daniel Wigdor. 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*, CHI '13, pages 2291–2300, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1899-0. doi: 10.1145/2470654.2481317. URL http://doi.acm.org/10.1145/2470654.2481317.

Anna Koster. Wiicon:acceleration based real-time conducting gesture recognition for personal orchestra. Diploma thesis, RWTH Aachen University, Aachen, December 2008.

Edward Lank, Yi-Chun Nikko Cheng, and Jaime Ruiz. Endpoint prediction using motion kinematics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 637–646, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-593-9. doi: 10.1145/1240624.1240724. URL http://doi.acm.org/10.1145/1240624.1240724.

S. M. LaValle, A. Yershova, M. Katsev, and M. Antonov. Head tracking for the oculus rift. In *IEEE International Conference on Robotics and Automation*, 2014.

Darren Leigh, Clifton Forlines, Ricardo Jota, Steven Sanders, and Daniel Wigdor. High rate, low-latency multi-touch sensing with simultaneous orthogonal multiplexing. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 355–364, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3069-5. doi: 10.1145/2642918.2647353. URL http://doi.acm.org/10.1145/2642918.2647353.

Daniel Leithinger, Sean Follmer, Alex Olwal, and Hiroshi Ishii. Physical telepresence: Shape capture and display for embodied, computer-mediated remote collaboration. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 461–470, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3069-5. doi: 10.1145/2642918.2647377. URL http://doi.acm.org/10.1145/2642918.2647377.

Ming Li and Leif Kobbelt. Actui: Using commodity mobile devices to build active tangible user interfaces. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, MobileHCI '15, pages 592–597, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3653-6. doi: 10.1145/2786567.2792895. URL http://doi.acm.org/10.1145/2786567.2792895.

Rene Linden. Multitouchkit: A software framework for touch input and tangibles on tabletops and. Master's thesis, RWTH Aachen University, Aachen, September 2015.

I. Scott MacKenzie and Colin Ware. 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*, CHI '93, pages 488–493, New York, NY, USA, 1993. ACM. ISBN 0-89791-575-5. doi: 10.1145/169059.169431. URL http://doi.acm.org/10.1145/169059.169431.

Michael Meehan, Sharif Razzaque, Mary C. Whitton, and Frederick P. Brooks, Jr. Effect of latency on presence in stressful virtual environments. In *Proceedings of the IEEE Virtual Reality 2003*, VR '03, pages 141–, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1882-6. URL http://dl.acm.org/citation.cfm?id=832289.835964.

Albert Ng, Julian Lepinski, Daniel Wigdor, Steven Sanders, and Paul Dietz. Designing for low-latency direct-touch input. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 453–464, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1580-7. doi: 10.1145/2380116.2380174. URL http://doi.acm.org/10.1145/2380116.2380174.

Albert Ng, Michelle Annett, Paul Dietz, Anoop Gupta, and Walter F. Bischof. In the blink of an eye: Investigating latency perception during stylus interaction. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 1103–1112, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2473-1. doi: 10.1145/2556288.2557037. URL http://doi.acm.org/10.1145/2556288.2557037.

Phillip T. Pasqual and Jacob O. Wobbrock. Mouse pointing endpoint prediction using kinematic template matching. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 743–752, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2473-1. doi: 10.1145/2556288.2557406. URL http://doi.acm.org/10.1145/2556288.2557406.

Andriy Pavlovych and Wolfgang Stuerzlinger. The tradeoff between spatial jitter and latency in pointing tasks. In *Proceedings of the 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '09, pages 187–196, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-600-7. doi: 10.1145/1570433.1570469. URL http://doi.acm.org/10.1145/1570433.1570469.

Orit Shaer and Eva Hornecker. Tangible user interfaces: Past, present, and future directions. *Foundations and Trends® in Human–Computer Interaction*, 3(1–2):4–137, 2010. ISSN 1551-3955. doi: 10.1561/1100000026. URL http://dx.doi.org/10.1561/1100000026.

Peter Tsoi and Jacob Xiao. Advanced touch input on iOS. Technical report, Apple Inc., 2015.

Philip Tuddenham, David Kirk, and Shahram Izadi. Graspables revisited: Multi-touch vs. tangible input for tabletop displays in acquisition and manipulation tasks. In

Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '10, pages 2223–2232, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-929-9. doi: 10.1145/1753326.1753662. URL http://doi.acm.org/10.1145/1753326.1753662.

P. P. Vaidyanathan. *The theory of linear prediction*. Number 3 in Synthesis Lectures on Signal Processing. Morgan & Claypool, 2008. doi: 10.2200/S00086ED1V01Y200712SPR03. URL http://resolver.caltech.edu/CaltechBOOK:2008.004. Copyright © 2008 by Morgan & Claypool.

Simon Voelker, Kosuke Nakajima, Christian Thoresen, Yuichi Itoh, Kjell Ivar Overgård, and Jan Borchers. Pucs: Detecting transparent, passive untouched capacitive widgets on unmodified multi-touch displays. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces*, ITS '13, pages 101–104, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2271-3. doi: 10.1145/2512349.2512791. URL http://doi.acm.org/10.1145/2512349.2512791.

Simon Voelker, Christian Cherek, Jan Thar, Thorsten Karrer, Christian Thoresen, Kjell Ivar Overgård, and Jan Borchers. PERCs: Persistently trackable tangibles on capacitive multi-touch displays. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, UIST '15, pages 351–356, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3779-3. doi: 10.1145/2807442.2807466. URL http://doi.acm.org/10.1145/2807442.2807466.

Haijun Xia, Ricardo Jota, Benjamin McCanny, Zhe Yu, Clifton Forlines, Karan Singh, and Daniel Wigdor. 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*, UIST '14, pages 205–214, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3069-5. doi: 10.1145/2642918.2647348. URL http://doi.acm.org/10.1145/2642918.2647348.

Index

Typeset September 21, 2016