DigiTap: An Eyes-Free VR/AR Symbolic Input Device

Manuel Prätorius* Dimitar Valkov† Ulrich Burgbacher‡ Klaus Hinrichs§
Visualization and Computer Graphics Research Group, University of Münster
Einsteinstraße 62, 48143 Münster, Germany

Abstract

In this paper we present *DigiTap*—a wrist-worn device specially designed for symbolic input in virtual and augmented reality (VR/AR) environments. DigiTap is able to robustly sense thumb-to-finger taps on the four fingertips and the eight minor knuckles. These taps are detected by an accelerometer, which triggers capturing of an image sequence with a small wrist-mounted camera. The tap position is then extracted with low computational effort from the images by an image processing pipeline. Thus, the device is very energy efficient and may potentially be integrated in a smartwatch-like device, allowing an unobtrusive, always available, eyes-free input. To demonstrate the feasibility of our approach an initial user study with our prototype device was conducted. In this study the suitability of the twelve tapping locations was evaluated, and the most prominent sources of error were identified. Our prototype system was able to correctly classify 92 % of the input locations.

CR Categories: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems: —Artificial, augmented, and virtual realities I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism: —Virtual reality

Keywords: virtual/augmented reality, character input, system control, wrist camera, optical glove

1 Introduction

Virtual and augmented reality (VR/AR) environments provide a versatile platform for communication and collaboration and show great potential for improving the work flow in various application domains such as 3D CAD/CAM design, architecture and in-situ information interfaces. Dedicated interfaces for *character input* and *system control*, as defined by Bowman et al. [2002], are vital to these systems, since in many cases the user is required to switch between different modes of interaction or to input additional data. For instance, in a CAD application the user has to specify whether an object itself or its vertices or faces are to be manipulated. Switching between object manipulation, sculpting, texture painting and other modes is also a common requirement in these applications as well as the precise specification of angles, lengths, object names and other properties, which requires a robust character input interface.

Augmented reality (AR) applications are currently proliferating, but are still lacking robust and unobtrusive input interfaces. For instance, some devices are using speech recognition as main input, which may be prone to errors due to ambient noise or differences in the pronunciation of names, and which may not be socially accepted in many situations. On the other hand, using a portable keyboard or a touch-enabled mobile device may be problematic, especially if eyes-free input is required.

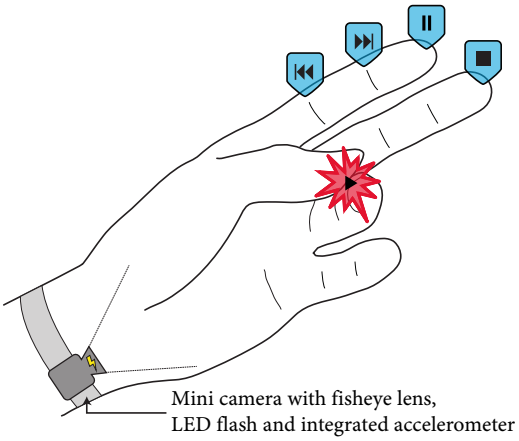

Figure 1: *Illustration of the DigiTap device. The controls of a music player are mapped to the fingertips and the knuckles. The music player is commanded to play by tapping with the thumb on the tip of the ring finger. The tap causes a vibrational signal, which is sensed by an accelerometer. Then a camera takes an image sequence of the hand to estimate the tapping location.*

Nevertheless, while navigation and object manipulation are addressed by many sophisticated VR interfaces, *character input* and *system control* are rarely being considered in depth and current VR/AR setups are mostly still using keyboards and GUIs [Bowman et al. 2004]. This holds especially if precise or numeric input is needed. Current strategies to achieve such input are to provide the user with some portable keypad device, such as the Twiddler [Lyons et al. 2004] and the Half-QWERTY [Matias et al. 1993] keyboards, or to map the touch positions on a pinch glove to characters and functions [Bowman and Wingrave 2001; Kuester et al. 2005], or to use a gesture-based sign language [Vardy et al. 1999].

All these strategies require additional intermediate devices, which may interfere with the use of equipment required by other interaction techniques, e. g., a magic wand or a Wiimote, and which may limit the user's natural hand movements and haptic sense, for instance because of the supporting cloth of a glove. Mobile, AR-based applications suffer from further issues. For instance, portable keypads are usually not readily available for casual input actions, and data gloves, which are often very expensive, might be socially inappropriate in some situations. Furthermore, the hardware setup is often very energy-demanding and not well suited for portable applications.

In this paper we present DigiTap—an eyes-free wrist-mounted input device, which can robustly detect thumb-to-finger taps at different locations on the fingers. These tap locations can be mapped arbitrarily to characters or system commands. Figure 1 shows an illustration of the DigiTap system, where the controls of a music

*e-mail:manuelpraetorius@uni-muenster.de
†e-mail:dimitar.valkov@uni-muenster.de
‡e-mail:burgbacher@uni-muenster.de
§e-mail:khh@uni-muenster.de

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
VRST 2014, November 11 – 13, 2014, Edinburgh, Scotland, UK.
2014 Copyright held by the Owner/Author. Publication rights licensed to ACM.
ACM 978-1-4503-3253-8/14/11 $15.00
http://dx.doi.org/10.1145/2671015.2671029

player are mapped to the tap locations. The idea to attach a camera to the wrist to enable system input with the user's hand has been used before. The first attempt was made by Vardy et al. [1999] where finger movements were mapped to a small set of input symbols. A recent approach is the Digits system developed by Kim et al. [2012]. They utilize a wrist-worn camera combined with an infrared laser to sense the user's hand and apply a new kinematic model to fully reconstruct the 3D pose of the hand. In contrast to their system we aim for discrete input and therefore constrain the input to taps with the thumb on the fingertips and the eight minor knuckles. Detecting the tap with an accelerometer allows to activate the other system components only when they are needed and therefore to decrease power consumption considerably. Thus, we speculate that the hardware may be integrated in a smartwatch-like device to provide unobtrusive, eyes-free, and always available symbolic input.

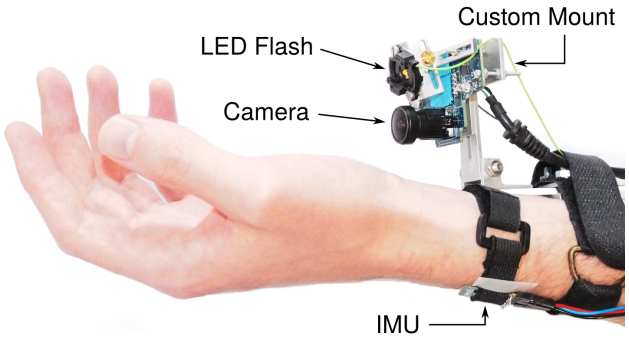

Figure 2: *The proof of concept DigiTap prototype.*

2 Related Work

Many particular features of VR/AR interfaces may render symbolic input a challenging problem, and there is a variety of different approaches to address this problem. Speech is, for instance, a common choice in this context [Bowman et al. 2002], but can lack precision in noisy or shared environments and may suffer from privacy and social acceptance issues. In addition, semi-random text input, as required by a route guidance AR system, is a difficult and usually error-prone task.

An alternative approach for symbolic input in VR/AR environments is provided by the variety of wearable keyboards and keypads proposed during the last decade. Popular examples for such devices are the one-handed chord keyboards, such as the "Twiddler" [Lyons et al. 2004], which are either held in or fastened to the user's hands and provide a set of buttons to enter characters or commands. Gloves or rings form another category of devices that have been used for symbolic input and system control in VR/AR. For instance, Bowman et al. have used a pinch glove to map thumb contacts on a user's hand to system menus [2001] or characters [2002]. A similar approach was proposed by Rosenberg and Slater in their Chording Glove device [1999] and was later extended by Kuester et al. within the KITTY project [2005]. Pratt [1998] has proposed a more generalized digital sign language, the "Thumbcode", for single-handed text input based on thumb-to-finger movements, suitable for these devices.

Another approach for mobile hand gesture recognition is to attach depth cameras to a user's body. Harrison et al. [2011] utilize a shoulder-worn depth camera to enable multitouch on different kinds of surfaces. In the work of Bailly et al. [2012] a Microsoft Kinect sensor is mounted on a shoe to sense different gestures performed with the fingers and arms. Systems where the sensor is not directly attached to the arm can suffer from occlusion issues and may not work when the user is walking. A commercial system for hand and finger tracking is the Leap Motion controller, which is meant to be placed on a desktop. Although its small size would allow to attach it to the wrist, the underlying recognition algorithm may not be suitable for this positioning.

The demand of less obtrusive user instrumentation and wearable computing has led to a variety of solutions for discrete and continuous input. The approach to use the human body as a touch surface is a popular concept in this context, since proprioception allows to naturally extend it for eyes-free input [Gustafson et al. 2013]. The Skinput sensor [Harrison et al. 2010], for instance, uses bio-acoustic signal acquisition to detect and locate taps on the user's arm and hand. In a user study the system was able to distinguish the four fingers by thumb-to-finger taps with an accuracy of nearly 90 %. Alternative approaches use accelerometers [Hrabia et al. 2013], electromygraphic [Chowdhury et al. 2013] or magnetic [Ashbrook et al. 2011] sensors, or a combination of these [Zhang et al. 2011]. Nevertheless, currently these approaches are in an early stage of development and robustly recognize only a limited set of input gestures.

An approach closely related to our work is to mount a camera and possibly a set of additional sensors to the user's wrist and use the data stream to reconstruct the hand posture [Kim et al. 2012; Vardy et al. 1999; Ahmad and Musilek 2006; Howard and Howard 2001]. For instance, the "WristCam" device proposed by Vardy et al. [1999] is able to detect a set of 3-finger gestures, which can be mapped to symbolic input, in an interface mimicking a sign language. The approach of Ahmad and Musilek [2006] utilizes a wrist-worn camera to track the finger positions. The device enables the user to type in the air like on a hardware keyboard. The finger positions are correlated to keystrokes by a machine learning algorithm. The Digits [Kim et al. 2012] sensor is a sophisticated depth-sensing device able to recover the full 3D pose of the user's hand. As it is designed as a general purpose interaction device it may potentially be suitable for symbolic input even though the idea was considered only rudimentarily by the authors. Furthermore, the underlying recognition algorithm might be unsuitable for precise detection of finger contacts and to determine whether a predefined end posture has been reached or the gesture is still in progress. In contrast, DigiTap is specifically designed for robust detection of a rich set of discrete inputs, but still allowing eyes-free interaction with minimal instrumentation.

3 The DigiTap System

To perform symbolic input with the DigiTap system, users have to tap with their thumbs on their fingertips, the 2nd knuckles or the 3rd knuckles. Each of these twelve tapping locations can be mapped to an arbitrary input action. Pinching thumb and a fingertip together is a common sensomotoric task. The 2nd and 3rd knuckles appear as distinct marks on our fingers and are therefore helpful to indicate eight additional tapping positions. Without considerable training, proprioception allows users to precisely touch their knuckles with the thumb without visual control.

Basically, the device consists of two sensors: 1) an accelerometer which is used to detect *if* a tap is performed, and 2) a camera with an LED flash which is used to detect *where* the user has tapped. One of the crucial differences to related wrist-mounted systems is that the camera does not need to capture a video stream, but only takes two consecutive images when triggered by the accelerometer.

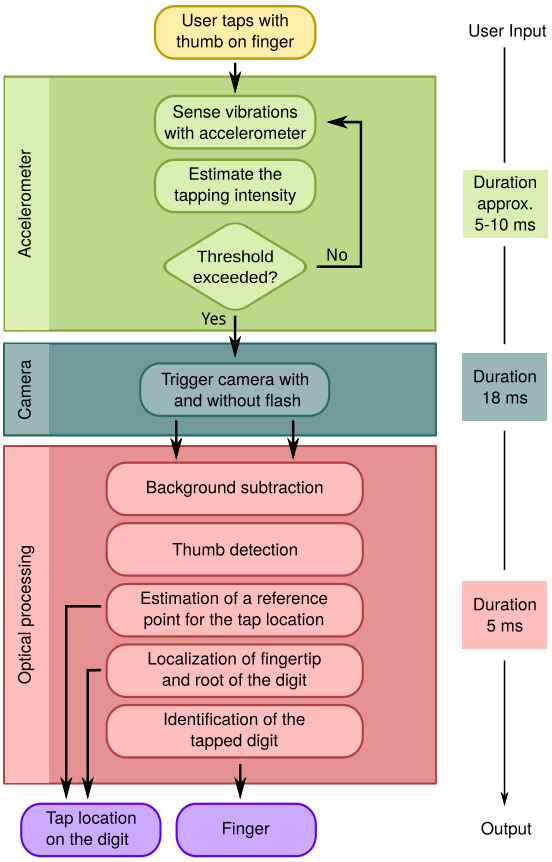

Figure 3: *Overview of the signal processing pipeline for a tapping recognition cycle.*

Triggering the system by detecting the vibrational signal of the tap is a power saving approach, since accelerometers are low-power devices and the camera and flash are activated for just a few milliseconds. By creating artificial shading conditions with an LED flash, it is possible to solve the detection of the tapping location with local appearance-based computer vision algorithms, which cause low computational effort. Thus, the DigiTap system has the potential to be embedded into a lightweight and stand-alone wrist-worn device.

3.1 Hardware

The DigiTap prototype shown in Figure 2 is based on off-the-shelf low-cost hardware. It is composed of two sensors and an LED flash: The optical sensor is a customized Sony PlayStation Eye camera, mounted with a 2.4 mm fisheye board-lens. The diagonal viewing angle is 132°. To allow for an adequate background subtraction the camera is capturing at 100 fps with a resolution of 320x240 pixels. Vibrations of the wrist are sensed with an LSM303DLHC 3-axis accelerometer integrated into the MinIMU-9 v2 board from Polulu sampling at 400 Hz. Data from the accelerometer is read via an I^2C interface connected to an Arduino Nano 3.0. The system is designed with an infrared flash LED and suitable band-pass filter for the camera, such that the flash remains invisible and the system is less obtrusive. In the current hardware prototype the LUXEON Rebel ES neutral white LED is used. Since the image processing algorithms are conceived for gray scale images, they do not need to be adapted. The sensors are attached to a custom mount made of aluminum, which is fastened to the forearm with hook-and-loop tape. For the user study the image processing algorithm was executed on

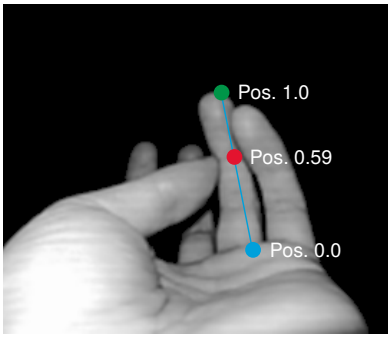

Figure 4: *Image of the tap point reconstructed by the DigiTap signal processing pipeline. The red dot represents the estimated reference position, and the blue line represents the reference line from the finger's root to its tip.*

an Intel® Core™ i7-2670QM CPU (single-threaded). To test if the optical processing is suitable for a stand-alone system, which can be worn on the wrist, we ran it on a Banana Pi (Allwinner A20: ARM Cortex-A7 dual-core, 1 GHz). On average the processing of one tap took 25 ms with this system (single-threaded).

The prototype is only meant for proof of concept purposes. In the current setup the Arduino (250 mW), the accelerometer (0.5 mW) and the camera (500 mW) are active all the time, whereas for each tap the LED flash (2 W) is only activated for about 18 ms. In a more sophisticated setup the camera should be replaced by a mini camera module (about 200 mW), which is only activated for the two snapshots. Instead of the neutral white LED an infrared LED (about 500 mW) should be sufficient for illumination. A smartphone with sufficient processing power for the optical algorithm consumes about 50 mW in sleep mode and about 2 W in normal mode (for each tap normal mode is just needed for less than 25 ms). Additionally, the Arduino which is constantly polling the accelerometer could be replaced by an interrupt-driven solution. In fact, the tap detection algorithm, which is described in section 3.3, can be efficiently implemented by an analog circuit. The output voltage of this circuit can be forwarded to an analog comparator which triggers an interrupt to wake up the remaining parts of the device. Considering an hour of work in a VR CAD environment, where a user types some annotations and does selections in menus, the flash and the camera are activated for approximately 1800 times. In this scenario the system would have a continuous energy consumption of about 70 mW h and additional 32 mW h for the recognition of the tapping positions. A smartwatch battery has a capacity of about 300 mA h, which is enough to supply the proposed system for about 12 h of continuous interaction.

3.2 Signal Processing Pipeline

The DigiTap's processing pipeline is outlined in Figure 3. If one of the fingers is tapped with the thumb, the impact produces a vibration signal which is sensed by the accelerometer. If the intensity of this signal exceeds a threshold value, a sequence of images is captured. Currently, the system takes two images: one without using the LED flash, and a second one immediately afterwards with activated LED flash. With the two captured gray scale images a background subtraction is performed, which results in a gray scale image of the illuminated hand. Qualitatively, the intensity of this gray scale image corresponds to the distance of the LED flash. This relation alleviates the detection of the tapping position and the identification of the touched finger. Thereafter, a reference point (red dot in Figure 4) for the thumb and a line (blue line in Figure 4)

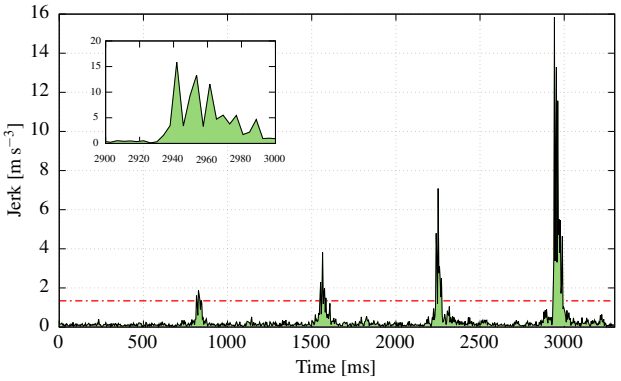

Figure 5: *Four different tapping intensities as measured by the L^1-norm of the jerk. The dashed red line indicates the tap detection threshold for triggering the image capture. The inset shows the 4th peak with stretched time axis.*

that extends from the fingertip to its root are estimated. The system output is a number D corresponding to the finger and a relative position L on the line as illustrated in Figure 4.

The duration of the processing stages is shown in Figure 3. Detecting the tap and triggering the camera takes about 10 ms. To capture the two consecutive images of the hand with and without flash takes 18 ms. This is the time the flash needs to fully illuminate the hand. All further image processing is done in about 5 ms. By adding additional 5 ms for transmission and synchronization the DigiTap system takes about 38 ms to process a tap. As discussed in the following section the accelerometer is muted for 40 ms after the first detection of a tap to avoid detecting the same tap multiple times. Therefore, the overall latency of the system is about 38 ms and the dead time is 50 ms. Considering the setup with the ARM Cortex-A7 mobile CPU the latency increases to 58 ms, which is still sufficient for discrete input.

3.3 Tap Detection

When the user performs a tap, mechanical vibrations are produced, which propagate through the user's hand. These vibrations are digitally captured on the wrist by the accelerometer and converted into an acceleration vector $\vec{a}(t)$, which is given in the accelerometer's coordinate system. Taps are characterized by abrupt variations of the absolute acceleration with respect to time and can therefore be detected as local extrema of the temporal derivative of $\vec{a}(t)$, the jerk $\vec{j}(t)$. To estimate the intensity I of the tap, the L^1-norm of $\vec{j}$ is calculated.

$$I = |j_x| + |j_y| + |j_z| = \left|\frac{\mathrm{d}a_x}{\mathrm{d}t}\right| + \left|\frac{\mathrm{d}a_y}{\mathrm{d}t}\right| + \left|\frac{\mathrm{d}a_z}{\mathrm{d}t}\right|. \qquad (1)$$

In Figure 5 the signal of four consecutive taps with increasing intensity is shown. To distinguish a tap from environmental noise a threshold is compared to the signal level, as indicated by the red line. Environmental noise mainly consists of vibrations from movements of the user's arm and body. Most hand movements like grasping or gesticulation are continuous and hence well distinguishable from tapping in the intensity signal. To find a reasonable threshold to distinguish even jerky movements from taps we have analyzed the noise produced by movements like walking, operating a phone, jogging, and walking up or down the stairs. The root mean squares (RMS) of the data captured from these movements are all less than $0.5\,\mathrm{m\,s^{-3}}$. However, descending stairs and jogging exhibited several peaks that reached up to $2.5\,\mathrm{m\,s^{-3}}$ and might interfere with light taps. To avoid wrong triggering in noisy situations the threshold may be calculated from the RMS of the signal when idling.

Considering the width of the tapping peaks, (40–60) ms, it is advantageous to immediately trigger the system when the rising edge of the signal reaches the threshold instead of analyzing the full peak so that the delay between a tap and the activation of the camera is as short as possible. It is assumed to be about (5–10) ms. As can be seen from the inset in Figure 5, which shows the fourth peak with stretched time axis, a single tap generates several consecutive falling and rising edges of similar intensity. To circumvent multiple triggering from a single tap the accelerometer is muted for approximately 40 ms. From preliminary tests we assume most users to be unable to perform two consecutive taps in less than 50 ms so that the "debouncing" of the accelerometer does not affect the tap rate.

3.4 Image Processing

The goal of the image processing algorithm is to detect the tap location of the thumb on a digit (see the red dot in Figure 4). The tap location consists of two parameters, the digit D that was tapped, and the location L of the tap on the digit. The latter value is measured relative to the estimated length of the finger.

An analytical approach was chosen, so there is no need for training for different users. The basic aspects, which were considered for the implementation, are low computational effort, simplicity and accuracy. As illustrated in Figure 3, the image processing consists of five steps, which are described in the remainder of this section.

Background Subtraction

The first step of the algorithm is to detect the hand in the picture taken by the camera. This can be a challenging task depending on the background and lighting conditions. To facilitate this process, two successive shots of the user's hand are taken, while the tapping is performed. The first image I_d is captured without the LED flash and the second image I_f is captured with the LED flash activated. Parts close to the LED, such as the user's hand, are illuminated and exhibit a measurable intensity change while the background intensity remains approximately constant. Pixels that do exhibit only a very small intensity change are considered as background and are being removed. The time between triggering the image capturing sequence and capturing the full illuminated image of the hand is about 18 ms. Although the delay between the two shots is short, it can cause a minimal displacement of the images, resulting in subtraction artifacts. To alleviate the effect of these artifacts, both images are blurred with a Gaussian Filter G, before the subtraction is performed. Thus, the output image I contains only the illuminated hand without background according to:

$$I(x,y) = \begin{cases} G * I_f(x,y) & \text{if } G * I_f(x,y) - G * I_d(x,y) > \tau_B, \\ 0 & \text{otherwise.} \end{cases}$$

The threshold τ_B depends on the preset photometric exposure and the environmental illumination conditions.

Utilizing Shading Aspects

As long as the additional illumination of the user's hand by the LED flash is stronger than the illumination by the environmental light, the intensity values of the image contain distance and orientation information in reference to the LED flash and camera position. Therefore, aspects of shading simplify the detection of the tapping

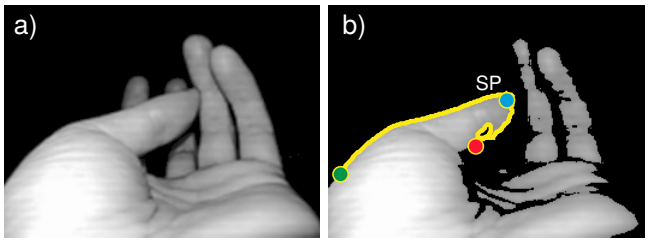

Figure 6: *Images after background subtraction. a) Without thresholding. b) After thresholding. For this image the thumb detection threshold τ_T is 140, which corresponds to the 25th percentile of all intensity values of a). The yellow path is the outline of the thumb, it is traced from the green point to the red point in search of a starting point (SP), which is marked blue.*

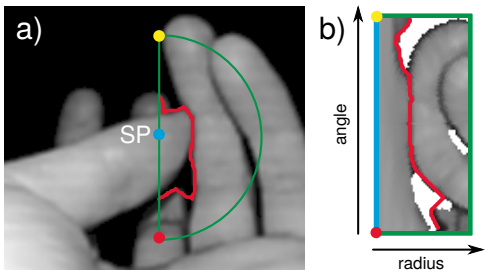

Figure 7: *Detection of the tapping intersection. a) The green half-circle frames the area, which is selected for transformation into a polar coordinate system. b) The intensity values of the image transformed into polar coordinates. In both images the trace with the minimum summed up intensities is marked in red.*

position. The thumb usually appears brighter than the fingers behind it, and it therefore is distinguishable by simple thresholding. Furthermore, the fingers have a horizontal intensity plateau, which is utilized to estimate the fingertip and the root of a finger.

Detecting the Thumb

Because with our system the thumb usually appears brighter than other parts of the hand, it is advantageous to detect the thumb first and then use it as a reference in the following processing steps. Further computations are based on the results of the thumb detection.

The thumb is separated from the fingers through the application of a threshold τ_T to the intensity image (see Figure 6). The thumb detection threshold τ_T is set to the 25th percentile of all intensity values in the background subtracted image. Considering various illumination conditions, the 25th percentile was chosen because it allows to reliably separate the thumb from the fingers without partitioning it into separated clusters of high intensity. After thresholding, the thumb's outline, marked yellow in Figure 6b, is traced beginning at the green point. For each point on the outline the euclidean distance to the lower left corner is calculated. If the distance of a point to the lower left corner of the image forms a local maximum and the vertical coordinate rapidly decreases while approaching a succeeding point with a local maximum distance, then the point is expected to be located on the tip of the thumb. If no point with these features can be found, the point with the maximum distance to the lower left corner is selected as the location of the tip. Depending on the illumination the thresholding might cut the tip of the thumb. Thus the selected point might be unstable for different thumb poses. Nevertheless, since this point is only used as starting point (SP) for the subsequent processing stages, this instability can be neglected.

Estimating a Reference Point for the Tap Location

To find the intersection of the thumb and the digit a circular area around the starting point is analyzed (see Figure 7 a). To ensure that the area contains the intersection, the original starting point, calculated by the thumb detection algorithm, is offset by a small distance towards the lower left corner, and then used as the center of the circular search area (see Figure 7a). The circular area is transformed into a polar coordinate system (see Figure 7b), and a dynamic programming approach is employed to find the trace with the minimum summed up intensities (marked red). To prevent the algorithm from following a trace in the black background it is set to white in the polar image.

To determine the (one-dimensional) location where the user has tapped the finger, a reference point (RP) is determined. This reference point should be stable among many different thumb poses. A point that meets this requirement lies on the topmost part of the intersection between the tip of the thumb and the digit. This point can be found reliably by a tracing algorithm proceeding from the starting point (SP) towards the intersection. In Figure 8a and 8b the red reference points for two different input locations are shown. The tracing algorithm starts at the green starting point and moves towards the pixel with the highest intensity, according to the allowed walking directions (N, NE, E, SE), until it hits the red intersection. The shading condition created by the LED flash constrains the path to the user's thumb and ensures that it heads towards the top of the thumb when the intersection is approached (see Figure 8b).

Locating the Fingertip and the Root of the Digit

The fingertip and the root of the digit have to be located to relate the reference point to the length of the tapped digit. At first, an initial point (IP) on the digit is selected by moving from the reference point onto the tapped digit. In Figure 9a the trace to the root of the digit is shown (marked yellow). To estimate its width, the finger has to be separated from the thumb and surrounding fingers. This is achieved by thresholding. Beginning at the initial point a tracing algorithm moves in the allowed directions (SW,S,SE,E), always selecting the pixel with the highest intensity. In each step, the width of the connected pixels in the current row with non-zero intensity is estimated. Assuming the user's fingers are aligned parallel to the vertical axis of the image, the width of the pixels with non-zero intensity corresponds to the width of the finger. If this value exceeds a certain threshold, the tracing algorithm terminates and returns the reference for the root of the finger, which is marked

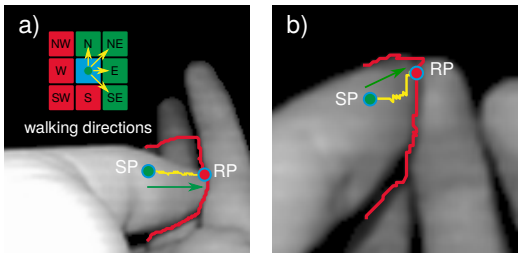

Figure 8: *Estimation of the reference point (RP) for the tapping position. In a) and b) two different tapping positions are shown. In search of the reference point, the yellow trace is followed starting at the green point and ending at the collision point with the tapping intersection (red trace). The inset shows the allowed walking directions.*

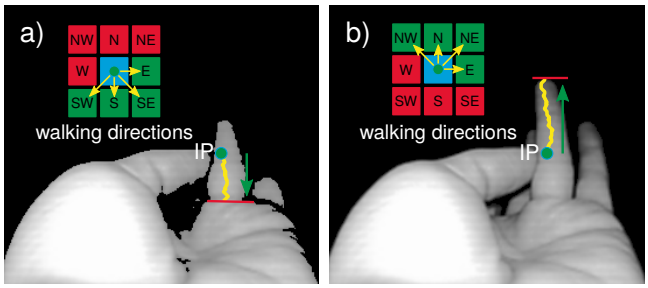

Figure 9: *Localization of the fingertip and the root of the finger. The path which is traced by the algorithm is marked in yellow. The green point indicates the initial position and the red line the fingertip b) or its root a). The insets show the allowed walking directions. a) Thresholded image to find the root of the finger. b) Non-thresholded image to find the fingertip.*

as a red horizontal line in Figure 9a. The fingertip is detected by a similar approach. In contrast to the detection of the root, the image is not thresholded because the fingertip should not be cropped. The tracing algorithm terminates when the intensity of all nearby candidate pixels is zero (see Figure 9b). To determine the tap location L the reference point is projected orthogonally onto the line connecting the fingertip and the root of the finger. The tap location L is then mapped to a single value by dividing the distance between the projected point and the root by the distance between the root and the fingertip (see Figure 4).

Identifying the Digit

The touched finger D is identified by a row scanning algorithm. The differentiation of the finger is achieved by the application of two thresholds. This is necessary because a high threshold guarantees the separation of all fingers but is at risk of cropping fingers, which are too low in intensity. In contrast, a low threshold alone does not crop but possibly leaves fingers connected. In Figure 10 the two thresholds are visualized. Just the bottom $40\,\%$ of the incident rows to the previously detected trace (colored green) of the tapped finger are scanned. This is because the pinky is smaller and in many testing cycles users tended to stretch the pinky away, which could result in not scanning it. The zero intensity pixels of all scanned rows are colored blue. In each scanned row the clusters of connected pixels with non-zero intensity are counted. Clusters which are below a certain width are not counted. If the row contains a cluster, which exceeds the assumed width of a finger, the whole row is neglected. In Figure 10 all valid clusters have a yellow mark at their end, and all invalid clusters and rows are marked red. Because of the directional illumination the fingers generally

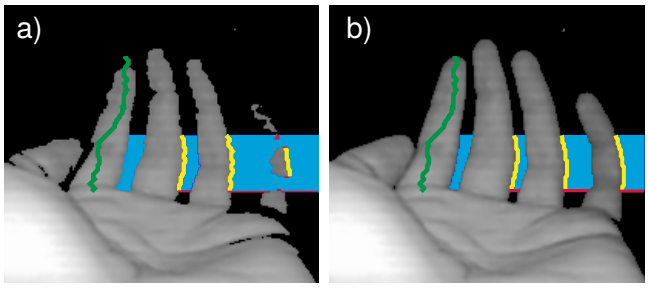

Figure 10: *Identification of the tapped finger. Thresholded images according to the 17th percentile a) and the 4th percentile b).*

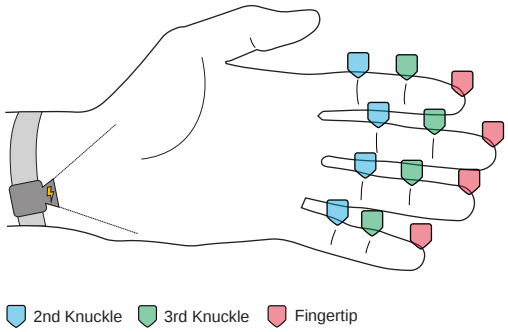

Figure 11: *All locations the participants had to tap on during the user study.*

do not tend to split into multiple clusters per row, when a threshold is applied. To circumvent the problems of cropping and not separated fingers, for each row the higher cluster count according to the thresholds is selected for identification.

4 Experimental Evaluation

To test the overall performance of our prototype implementation an initial user study was performed. In this study we decided to not separate the user's tap precision (based on her proprioception and finger control ability) and the system's recognition precision. Instead, we tested the combined overall system-user performance. Therefore the participants had to tap on either a fingertip or one of the knuckles with the thumb, and the acquired data and detection rates were evaluated.

4.1 Participants

Twelve volunteers (two female) in age from 24 to 34 (median 30) participated in the experiment. All participants were right-handed, and all were students or professionals at our university. Most of them reported to be experienced in computer games, whereas the self-reported familiarity with VR/AR input devices was rated as either average, or low. Each test session, including training, debriefing and breaks, took approximately one hour. The participants were allowed to take a break at any time.

4.2 Task

The participants were asked to tap at twelve different input locations on their fingers. These locations are the fingertips and the eight minor knuckles (see Figure 11) resulting in a total of twelve locations. The interaction is restricted to taps on the side of the finger which is faced into the direction of the thumb to enable the image processing algorithm to correctly identify the finger as shown in Figure 8. The fingertips and the eight minor knuckles were chosen because they provide easily identifiable and discrete interaction points for the user. Although it could be possible to use the intermediate positions, they were neglected because in a preliminary test users were confused by too many locations, and the positions were too close to be reliably distinguished by the system.

During the study the right-handed users had the DigiTap prototype placed on their left wrist. As the device is intended to be integrated into a smartwatch the location on the left wrist enables these users to operate the buttons and the screen of the watch with their dominant hand. We expect that the performance of the tap interaction would be equal or even better with the dominant hand.

4.3 Procedure and Setup

Participants were introduced to the interaction technique with a brief demonstration and an explanation of the underlying principle. Afterwards, the DigiTap prototype was placed on their wrist and tightened with a wristband. With our prototype it was possible to adjust the orientation and position of the camera and the flash regardless of the placement of the wristband in order to enable users to perform the taps comfortably. In foregone tests of the system we found that some users intuitively tended to stretch their hands away instead towards the camera while performing a tap. To capture these taps the camera was positioned higher above the wrist than necessary because the focus of the users should lie on the tapping position instead of the wrist posture. In the user study the distance between optical axis and base of the camera mount was 3 cm and the distance to the heel of the hand, depending on the appearance of the users arm and hand, about 2.0 - 2.5 cm. Experienced users should be able to tap on the right position and to keep their hand in the field of view even when the camera touches the heel of their hand (distance to optical axis 0.8 cm).

To set the threshold for the tap intensity above which the camera is triggered a calibration procedure was carried out. Users had to move their arm and fingers slightly without tapping. From the RMS of the environmental noise during the calibration process the threshold was calculated. Then the participants had to test if they were able to trigger the system comfortably at each input location. If triggering caused difficulties, the threshold was lowered manually.

In order to become familiarized with the system each participant had to complete a training period, during which additional instructions about the usage and pose corrections were given. Especially, users were told to bend their fingers slightly towards the device to bring the digits into the field of view of the camera. The training period was terminated after 15 minutes.

In the study users had to tap 15 times at each location, resulting in a total of 180 taps. The input locations for each trial were presented on a computer screen by a blue ellipse on an image of a real hand similar as shown in Figure 11. The input order of the locations was randomized.

The participants were not given direct feedback where exactly the tapping position was located or if they successfully hit the right spot. The user was supposed to act as natural as possible without adapting too much to the characteristics of the system.

4.4 Results and Discussion

In this section, we report on the performance of the DigiTap prototype. Overall, 2160 trials were performed. 91.9 % of the tap locations were correctly detected and classified. The remaining 8.1 % of the trials were erroneous due to different reasons, which are described in the following section. Additionally, the repeatability of the different tap locations is analyzed.

Sources of Error

In the present version, the image processing algorithm does not check the plausibility of the results of its stages. These checks were omitted to keep the computational demands of the algorithm as low as possible. Thus, errors in the image processing result in the wrong estimation of the input location. These errors are either that the wrong digit is detected, or that the finger has been identified correctly but the distance d between the tap and the estimated input location is unexpectedly large. We consider an estimated input location as erroneous if d exceeds three times the interquartile range (IQR) of the distribution of tap locations for the desired input location.

		Predicted Class		
		Fingertip	3rd Knuckle	2nd Knuckle
Actual Class	Fingertip	669	4	0
	3rd Knuckle	5	677	12
	2nd Knuckle	0	20	638

Table 1: *Confusion Matrix resulting from a 10-fold cross-validation with a k-nearest neighbor ($k = 5$) classifier.*

During the study, the first error (wrong digit) occurred in 110 cases (5.1 % of the trials). The second error was encountered 25 times (1.2 % of the trials). The data of the erroneous trials were examined manually to find the underlying source of error. The amount of errors are categorized by their source and visualized in Figure 12, bottom. Five different sources of error are shown:

Capture Error Covers all cases, where the camera captured an incomplete or no image.

Posture Error A trial is classified as posture error, if the hand is not completely visible to the camera, or regions of the image that are relevant for the estimation of the input location are hidden by other fingers or the wrist.

Trigger Error Sums up all accidentally triggered trials.

System Error A trial is classified as a system error, if no other cause can be specified. These errors mainly appear if the illumination conditions are such that the system is not able to detect the intersection between thumb and finger properly or to identify the tapped finger.

User Error If the tap is located correctly by the system but the user simply tapped at another input location than expected, the trial is classified as a user error.

On average a study session contains 6.3 % error trials. User #4 had problems to comfortably trigger the system on the pinky. To facilitate the triggering, the threshold for the tapping intensity was lowered manually. This enabled peaks from the background noise, which are sometimes caused by jolting the cable connection of the IMU, to unintentionally trigger the system, which contributed strongly to the high trigger error rate of user #4. In contrast, user #8 triggered the system accidentally in 6.7 % of the trials. This value was unexpectedly high compared to the median trigger error rate of 1.4 %. Scrutinizing the accelerometer and image data of user #8 revealed that additional peaks where generated because the user released the tap with an abrupt movement. Most of these wrong triggerings could be avoided by increasing the "dead time" of the accelerometer to 150 ms.

Repeatability of Input Locations

The output of the DigiTap prototype is the two-dimensional tap location (D, L). The discrete value D corresponds to the digit which was tapped, and the continuous value L represents the location of the tap on the digit. L is the relative position of the reference point compared to the fingertip and the root of the finger. The value of L for a distinct input location depends upon several factors. These factors are the shape and size of the finger, the thickness of the thumb, the posture of the finger relative to the optical axis, and the

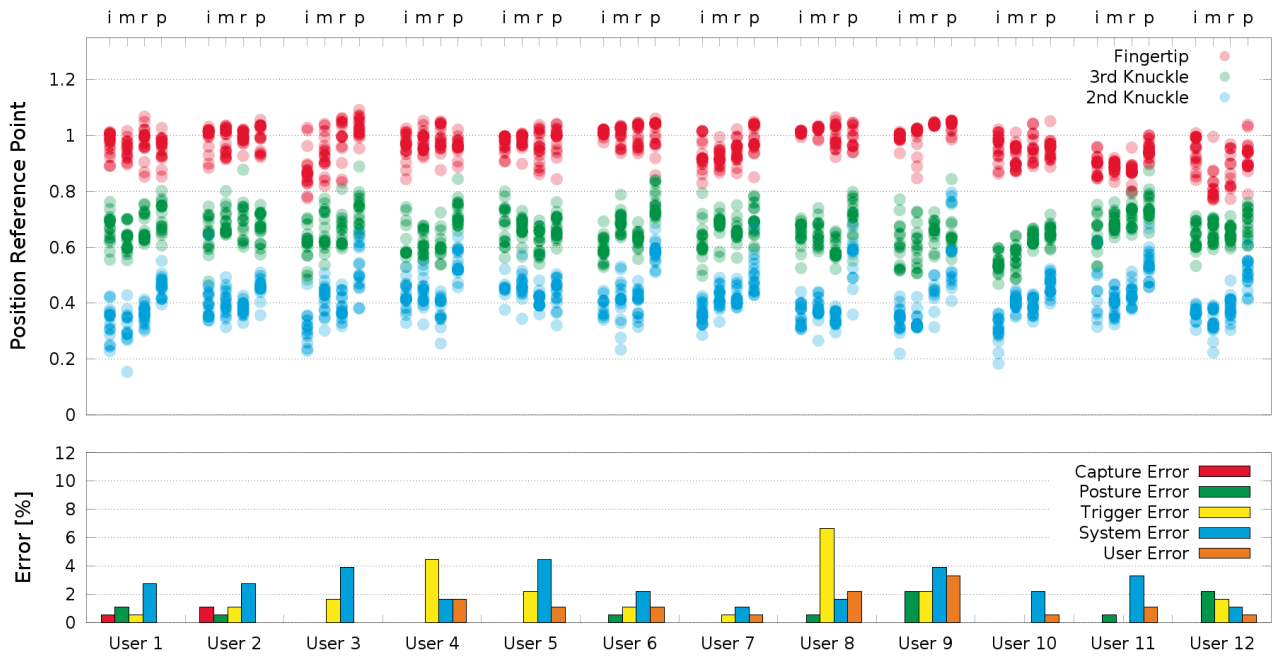

Figure 12: *The upper chart shows the raw data for the individual users and each of their fingers (i = index, m = middle, r = ring, p = pinky). The lower chart shows the sources of error for each user.*

illumination conditions. The distance between the fingertip and the root changes for each trial even when the same tapping location is given. Therefore this distance can not be used to directly measure the accuracy of a tap or the spatial precision. Nevertheless, we can determine the repeatability of the tap location L. The repeatability is crucial in order to employ DigiTap as a robust interaction device. It is important to keep in mind that L accumulates the errors of the user and the system.

The results for all trials where the finger is identified correctly and which are not considered as outliers are plotted in the upper chart in Figure 12. The plotted position of the reference point corresponds to the output shown in Figure 4. The red clusters belong to the fingertip positions, the green ones to the 3rd knuckle and the blue ones to the 2nd knuckle. Considering each finger, the data clusters for the three tapping positions are almost always linearly separable.

To evaluate the performance as an input system, a 10-fold cross-validation with a k-nearest neighbor ($k = 5$) classifier was conducted. Table 1 shows the resulting confusion matrix. Only the taps where the identification of the tapped finger was correct were used for classification. Among these values 41 taps were misclassified according to their location on the finger. Taking into account all misclassifications and errors discussed above the system is able to correctly classify 91.9 % of all trials. It has to be mentioned that the classification performance depends on the considered digit as can be seen in Figure 13b.

Comparison of the Input Locations

According to the anatomy of the hand and the individual dexterity some input locations are easier to tap than others. To evaluate the subjective usability of the respective tap positions, the users were asked to rate each input location on a 5 point Likert scale. They were presented the following statement: "It is comfortable to tap at the indicated location". The results of the questionnaire are shown in Figure 13a, where 1 corresponds to "strongly agree" and 5 means "strongly disagree". As can be seen, the comfort of tapping generally decreases from the fingertip towards the root of the finger. For the index, the middle and the ring finger ratings range from "strongly agree" to "neutral". The pinky shows a stronger gradient in comfort according to the input locations. Considering this, the layout of possible user interfaces should be arranged such that actions which are scarcely invoked are mapped to the 3rd and 2nd knuckle of the pinky.

To compare the subjective ratings with the results of the study, we visualized the misclassified trials for the respective input locations in Figure 13b. These data show that there is also a small increase of the misclassification rate going from the tip to the root. In correspondence with the subjective ratings the 2nd and 3rd knuckle of the pinky show the largest classification errors. This could be due to two aspects: Tapping at these locations is more challenging than

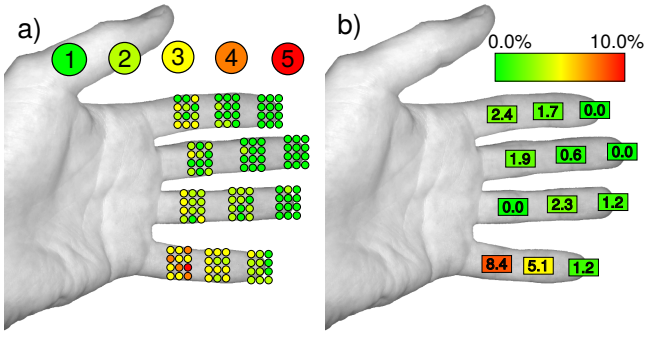

Figure 13: *a) Subjective user ratings about their ability to tap at the respective locations. b) Percentage of misclassified trials*

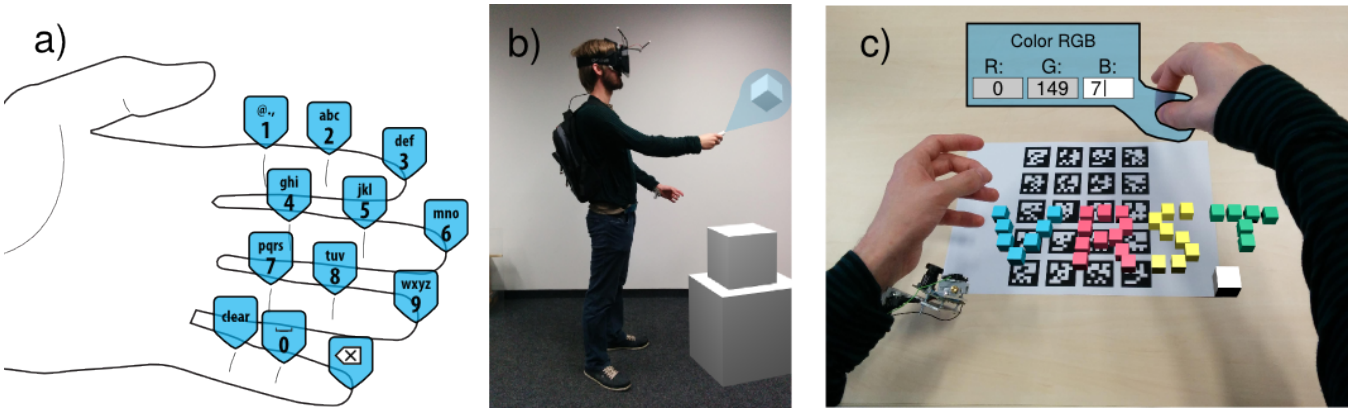

Figure 14: *a) A custom multi-tap keyboard is mapped to the fingertips and the knuckles. b) Conceptual illustration of a possible point-and-click interaction scenario, where DigiTap could be used to switch the mode of interaction. c) Conceptual illustration of an AR scenario, where the user could perform symbolic input to set object attributes.*

the others, and therefore the users were not as precise. Some of the participants stretched the pinky away from the camera while tapping at these locations, which reduces the accuracy of the reference position relative to the finger.

5 Design Implications

The availability of discrete input enables a variety of additional functions in many VR/AR applications [Bowman et al. 2002]. Especially character input is compelling in many situations, such as object labeling in CAD or medical applications, leaving a message in a collaborative environment, or invoking and specifying particular functionality in mobile AR interfaces based on see-through displays. A straightforward implementation of a character keyboard with DigiTap may be to map the buttons of a common multi-tap keyboard (known from keypad equipped mobile phones) to knuckles and fingertips of the index, middle and the ring fingers, as shown in Figure 14a. In this setup up to five characters are mapped to each available finger location and are cycled when the user taps at a location. A character is selected, if a predefined delay has been exceeded, or if the user taps at a different input location. In a preliminary evaluation the authors were able to achieve a text entry rate of approximately 10 words-per-minute (WPM) after only a couple of hours training. Though these results were not confirmed by a rigorous user study, the achieved entry rates are very promising, especially in comparison to the average typing speed on a standard 12-key mobile phone keypad (11.05 WPM according to Wigdor and Balakrishnan [2004]).

In general, an arbitrary soft-keypad layout could be mapped to the tap locations on the fingers. Nevertheless, since no visual feedback is provided, the user has to remember the mapping for each used keyboard layout and constantly keep in mind which layout is currently active. In cases where eyes-free input is not required, the application could compensate for this by providing a visual feedback on a display device, but in many cases such an approach is not applicable or undesirable.

An interesting result from our initial evaluation is that the perceived effort differs with the tap location (see Figure 13a), i. e., the participants of the study generally rated taps on the pinky finger and on the inner knuckles as less comfortable than taps on the fingertips of the index or middle finger. While not surprising, the results might have a strong impact on the *user induced* input error rates and on the learnability of particular function mappings. For instance, it might be more adequate to map the fingertips of the index, middle and the ring finger to functions that are commonly invoked and the inner knuckles to support functions. In the example music player interface (see Figure 1) the "main" functions, e. g., *play*, *stop* and *pause*, are mapped to the fingertips, while the secondary functions (*next* and *previous* song) are mapped to the second and third knuckles of the index finger. In this interface design, we considered *pause* as most critical, demanding fast and easy access and robust detection, and we therefore mapped it to the fingertip of the index finger.

The strong difference in the perceived effort to tap on the pinky finger compared to all other locations makes these taps very distinguishable, as some of the participants in the experiment reported. Thus, it might be appropriate to map functions with heavy payload, e. g., *delete an object*, *backspace*, *undo*, to these locations (see Figure 14a). Furthermore, the detection rates for these locations are also considerably worse (see Figure 13b). While in most cases the user can simply tap again at the desired location, some functions (most prominently *undo*) require very robust detection. In these cases the fingertip of the pinky would be most appropriate, since it combines a distinguishable muscle effort with very high detection rates (see Figure 13).

One of the main design considerations in the development of the DigiTap device was to allow easy co-usage of further devices, such as a magic wand, laser pointer or a free-hand gesture-based interaction metaphor (e. g., Song et al. [2012]). For instance, in a point-and-click interaction technique a DigiTap device might be used to trigger different actions (e. g., *(de)select*, *delete*, *move-to*), while a 3 or 6 DoF linear input device could be used to position and orientate a virtual cursor (see Figure 14b). Furthermore, the fact that the user does not need to wear a glove or other instrumentation on her fingers makes DigiTap perfectly suited for free-hand interaction techniques. An interface concept for this use case is illustrated in Figure 14c. Here a user's hand gestures are captured and processed by an external tracking system and used for object manipulations. Grasping the color panel would pause the external hand tracking and enable string input from the DigiTap. Furthermore, we suppose that although DigiTap relies on an IR flash for proper operation, it will not interfere significantly with an external tracking system. The reason for this is that (in contrast to similar systems) the flash is only enabled for a very short interval (currently 18 ms with proper synchronization), and there is an interval of at least 200 ms between two consecutive taps. Thus, even a high speed camera would capture the flash light in only a few frames before it disappears, which can easily be filtered out by a simple temporal filter.

6 Conclusion and Future Work

In this paper we have presented DigiTap, an approach to provide discrete input in virtual and augmented reality environments. The proposed system is specifically designed to enable freehand, eyes-free, and unobtrusive interaction. We have described the design of a prototype device built from low-cost off-the-shelf hardware. Results from a user study indicate that the system is capable of reliably detecting 12 different input locations on a user's hand with an accuracy of nearly 92 %.

The proposed prototype is still in a preliminary stage, we plan to miniaturize it and turn it into a stand-alone system. This will allow to conduct user studies of AR/VR applications in a more convenient way. The possibility to measure the intensity of a tap could be utilized to extend the input modalities. Therefore we would like to test users upon their ability to repeat different tapping intensities.

References

AHMAD, F., AND MUSILEK, P. 2006. A keystroke and pointer control input interface for wearable computers. In *Fourth Annual IEEE International Conference on Pervasive Computing and Communications*, 10–pp.

ASHBROOK, D., BAUDISCH, P., AND WHITE, S. 2011. Nenya: subtle and eyes-free mobile input with a magnetically-tracked finger ring. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2043–2046.

BAILLY, G., MÜLLER, J., ROHS, M., WIGDOR, D., AND KRATZ, S. 2012. Shoesense: a new perspective on gestural interaction and wearable applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 1239–1248.

BOWMAN, D. A., AND WINGRAVE, C. A. 2001. Design and evaluation of menu systems for immersive virtual environments. In *Proceedings of IEEE Virtual Reality*, 149–156.

BOWMAN, D. A., RHOTON, C. J., AND PINHO, M. S. 2002. Text input techniques for immersive virtual environments: An empirical comparison. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 46, SAGE Publications, 2154–2158.

BOWMAN, D. A., KRUIJFF, E., LAVIOLA JR, J. J., AND POUPYREV, I. 2004. *3D user interfaces: theory and practice*. Addison-Wesley.

CHOWDHURY, A., RAMADAS, R., AND KARMAKAR, S. 2013. Muscle computer interface: A review. In *ICoRD'13*. Springer, 411–421.

GUSTAFSON, S. G., RABE, B., AND BAUDISCH, P. M. 2013. Understanding palm-based imaginary interfaces: The role of visual and tactile cues when browsing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 889–898.

HARRISON, C., TAN, D., AND MORRIS, D. 2010. Skinput: appropriating the body as an input surface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 453–462.

HARRISON, C., BENKO, H., AND WILSON, A. D. 2011. Omnitouch: wearable multitouch interaction everywhere. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 441–450.

HOWARD, B., AND HOWARD, S. 2001. Lightglove: Wrist-worn virtual typing and pointing. In *Proceedings of the Fifth International Symposium on Wearable Computers*, IEEE, 172–173.

HRABIA, C.-E., WOLF, K., AND WILHELM, M. 2013. Whole hand modeling using 8 wearable sensors: biomechanics for hand pose prediction. In *Proceedings of the 4th Augmented Human International Conference*, ACM, 21–28.

KIM, D., HILLIGES, O., IZADI, S., BUTLER, A. D., CHEN, J., OIKONOMIDIS, I., AND OLIVIER, P. 2012. Digits: freehand 3d interactions anywhere using a wrist-worn gloveless sensor. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, 167–176.

KUESTER, F., CHEN, M., PHAIR, M. E., AND MEHRING, C. 2005. Towards keyboard independent touch typing in vr. In *Proceedings of the ACM symposium on Virtual reality software and technology*, 86–95.

LYONS, K., STARNER, T., PLAISTED, D., FUSIA, J., LYONS, A., DREW, A., AND LOONEY, E. 2004. Twiddler typing: One-handed chording text entry for mobile phones. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, 671–678.

MATIAS, E., MACKENZIE, I. S., AND BUXTON, W. 1993. Half-qwerty: A one-handed keyboard facilitating skill transfer from qwerty. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 88–94.

PRATT, V. R. 1998. Thumbcode: A device-independent digital sign language. In *Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science*.

ROSENBERG, R., AND SLATER, M. 1999. The chording glove: a glove-based text input device. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 29*, 2, 186–191.

SONG, P., GOH, W. B., HUTAMA, W., FU, C.-W., AND LIU, X. 2012. A handle bar metaphor for virtual object manipulation with mid-air interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 1297–1306.

VARDY, A., ROBINSON, J., AND CHENG, L.-T. 1999. The wristcam as input device. In *Proceedings of the IEEE 16th International Symposium on Wearable Computers*, 199–199.

WIGDOR, D., AND BALAKRISHNAN, R. 2004. A comparison of consecutive and concurrent input text entry techniques for mobile phones. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, 81–88.

ZHANG, X., CHEN, X., LI, Y., LANTZ, V., WANG, K., AND YANG, J. 2011. A framework for hand gesture recognition based on accelerometer and emg sensors. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 41*, 6, 1064–1076.