Magic Finger: Always-Available Input through Finger Instrumentation

Xing-Dong Yang[1,2], Tovi Grossman[1], Daniel Wigdor[3], George Fitzmaurice[1]

[1]Autodesk Research
210 King St. East, Toronto,
ON, M5A 1J7, Canada
{firstame.lastname}@autodesk.com

[2]University of Alberta,
Edmonton, AB,
Canada, T6G 2E8
xingdong@cs.ualberta.ca

[3]University of Toronto,
Toronto, ON,
Canada, R3T 2N2,
dwigdor@dgp.toronto.edu

ABSTRACT

We present Magic Finger, a small device worn on the fingertip, which supports always-available input. Magic Finger inverts the typical relationship between the finger and an interactive surface: with Magic Finger, we instrument the user's finger itself, rather than the surface it is touching. Magic Finger senses touch through an optical mouse sensor, enabling any surface to act as a touch screen. Magic Finger also senses texture through a micro RGB camera, allowing contextual actions to be carried out based on the particular surface being touched. A technical evaluation shows that Magic Finger can accurately sense 22 textures with an accuracy of 98.9%. We explore the interaction design space enabled by Magic Finger, and implement a number of novel interaction techniques that leverage its unique capabilities.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design

Keywords: Touch input, always-available input, gesture input, contextual action, texture reorganization

INTRODUCTION

Recent years have seen the introduction of a significant number of new devices capable of touch input. While this modality has succeeded in bringing input to new niches and devices, its utility faces the fundamental limitation that the input area is confined to the range of the touch sensor. A variety of technologies have been proposed to allow touch input to be carried-out on surfaces which are not themselves capable of sensing touch, such as walls [16, 20], tables [20], an arbitrary piece of paper [18] or even on a user's own body [18, 26, 40].

Mounting cameras on the body has enabled these regions to become portable [17, 18, 34, 36]. However, like other vision-based implementations, the range of the sensor is limited to the viewing area of the camera, thus the capabilities of these sensors are in some ways more limited than are non-vision based techniques.

UIST'12, October 7-10, 2012, Cambridge, MA, USA.

These approaches to instrumentation have all focused on enabling touch capability for surfaces the user will touch with their finger. To overcome their inherent limitations, we propose *finger instrumentation*, where we invert the relationship between finger and sensing surface: with Magic Finger, we instrument the user's finger itself, rather than the surface it is touching. By making this simple change, users of Magic Finger can have virtually unlimited touch interactions with any surface, without the need for torso-worn or body-mounted cameras, or suffer problems of occluded sensors.

Our work has been inspired by earlier projects in always-available input [41]. Like those earlier projects, Magic Finger is capable of sensing the moment of touch and relative finger displacement on a surface. Further, inspired by the versatility of the human finger, Magic Finger can also sense the texture of the object a user touches, allowing for the use of a machine learning classifier to recognize the touched object. Thus, contextual actions can be carried out based on the particular surface being touched. Additionally, Magic Finger can also identify artificial textures and fiduciary markers. This extends its ability to recognize richer information, thus enabling many novel interaction techniques.

Magic Finger is a thimble-like device worn on the user's finger (Figure 1). It combines two optical sensors: a lower-resolution, high speed sensor for tracking movement, and a higher-resolution camera for capturing detail. In addition to this device, the contributions of this work also include an exploration of associated hardware and software design spaces, an evaluation of its texture sensing accuracy levels, and a set of applications and interactions that are enabled by Magic Finger.

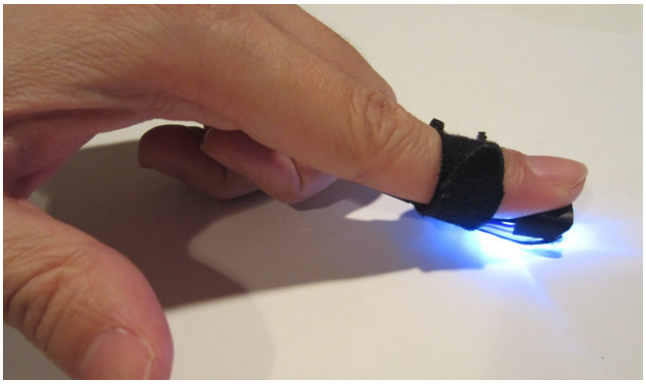

Figure 1: The Magic Finger device.

RELATED WORK

In this section, we review the related literature in the areas of always-available input, research on contextual actions, and augmentation of the user's finger.

Always-Available Input

Advanced sensing technologies have been used to help extend our ability to interact with computers from anywhere, supporting always-available input [41].

One way to accomplish this is to extend the range of devices. SideSight [12] uses an array of proximity sensors to detect a users' finger movement on the side of a cell phone. Therefore, touch input is no longer limited to the small region of the touch screen. Similarly, Portico [7] and Bonfire [30] used cameras installed on a laptop to capture the input occurring around it. PocketTouch allows capacitive sensors to function through fabric [40]. While these devices extend the range of sensors, they also are anchored to a device.

Efforts have also been made to instrument a user's surrounding environment to enable touch. Touché [42] enables touch on everyday objects, including humans and liquids, while Scratch Input [20] recognizes gestures on larger surfaces, but each require a sensor to be connected to the touched object. Wimmer and Baudisch [49] demonstrated that touch gestures can be enabled on an arbitrary object by having the object connect to an electric pulse generator and a Time Domain Reflectometry sensor.

Camera based sensors have also been used to sense input. Wilson [47] demonstrated that a properly positioned depth camera can be used to sense touch on surfaces. In the LightSpace system [48], a small room becomes interactive by combining multiple depth cameras and projectors.

While the instrumentation of environments is promising, it is also limited. Alternatively, the human body can itself be instrumented to sense input. Interactive clothing is one way to do so [28], but requires special clothing.

Many techniques enable body input of taps, postures, and whole body gestures. For example, Saponas et al. [41] used arm mounted electromyography sensors to sense pinch gestures made by different fingers. Cohn et al. [16] proposed a technique to sense whole-body gestures that utilizes existing electromagnetic noise from nearby power lines and electronics. Harrison et al.'s Skinput [26] used arm mounted bio-acoustic sensors to detect finger tap on the different locations of a users' forearm. Tiny sensors could also be embedded on [36], or implanted under [29] the user's skin. These techniques all open up new and interesting interaction opportunities, however they cannot sense detailed finger movements required to emulate touch input.

Several techniques have supported richer input on the human body [24]. The Imaginary Phone [17] uses a 3D camera to detect taps and swipes on a user's palm. OmniTouch [18] uses a depth-sensing camera and a micro projector to turn a user's palm into a touch-sensitive display. SixthSense uses a small projector and camera, worn in a pendant device, to enable touch interactions on and around the body [34]. While such techniques enable always-available input, the input is generally restricted to the body, and can suffer from occlusion. In contrast, we propose *finger instrumentation*, enabling input on almost any surface.

Contextual Interactions

Magic Finger is able to recognize textures and use the information to understand the environment, or object the user is interacting with, and perform associated contextual actions. While texture has been previously explored as an output channel for touch input [8, 22, 31] we are unaware of work where texture is used as an input channel.

Relevant work by Harrison and Hudson used multispectral material sensing to infer the placement of mobile devices, and discussed potential uses, such as tunring off a cell phone when it is placed in a purse [19]. Alternatively a touch sensitive device can respond differently if it knows what caused the touch or who touched it [6, 25, 38, 50]. More broadly, research on contextual interactions in the ubiquitous computing literature is also relevant. We refer the readers to comprehensive surveys [10, 14].

Augment The User's Finger

Early work in augmenting the human finger lies in the virtual reality research, where data gloves are used to track the user's hand position in space [43]. Several projects also exist outside of the VR literature. Marquardt [32] augmented a user's finger with fiduciary markers to inform an interactive tabletop what part of the finger was in contact with the surface. With Abracadabra [21], users wear a magnet on their finger to provide input above a mobile device. FingerFlux [44] also uses a small magnet on the finger, to simulate haptic feedback above an interactive tabletop. While these projects are all related in that they instrument the finger, none had the goal of supporting always-available input.

Logisys's Finger Mouse, a cylinder-shaped optical mouse that can be mounted on a finger, is a relevant commercial product [5]. It is used to control an on-screen cursor with a relative input mapping. Magic Finger extends this basic capability to also support absolute targeting, texture recognition, and fiduciary marker identification. Furthermore, with Magic Finger, we explore a smaller form factor, and a rich interaction design space beyond cursor control.

HARDWARE DESIGN SPACE

In designing a device, our main goal is to enable the finger to sense touch on physical objects, without any environmental instrumentation. In order to emulate traditional touch techniques, the device must be able to sense contact and 2D positional movements. In this section, we present a design space of possible hardware implementations that could enable these goals. For a broader discussion of hardware for small on-body devices, we refer the reader to Ni and Baudisch' s work on *disappearing mobile devices* [36].

Scope of Recognition

We define three scopes of recognition: *none*, *class*, and *instance*. In its simplest form, the device could be able to sense finger movement without the capability of knowing

anything about the contact object (*none)*. However, when it is the finger itself performing the sensing, it would be helpful to know what the finger is touching, so the input could be directed to an appropriate device. The scope of recognition could be increased to recognize certain classes of objects (e.g. a table or a phone) (*class)*, or could be further extended to distinguish between specific instances of objects (e.g. my table vs. your table) (*instance)*.

Sensing X-Y Movements

To emulate traditional touch, the hardware must be able to sense X-Y movements. Previous work has shown that an optical flow sensor (found in optical mice) can be used as a small touch sensitive device [9, 36]. As such, 2D finger movements could be detected by affixing an optical flow sensor to the finger. Such sensors can be extremely small and offer reliable 2D motion sensing on a wide variety of surfaces, but a light source is required. Alternatively, a mechanical device, such as a small trackball could be used, but may not be as robust to varying types of surfaces. A mechanical device may also require a larger form factor.

Sensing Material

To increase the recognition *scope*, the device could sense the material that the finger is in contact with. This could be achieved with optics, using texture recognition algorithms [37]. Using an optical flow sensors may be difficult, as their resolutions are typically low (e.g. 18x18). Higher quality cameras may be required to improve the robustness of material recognition or to extend the recognition scope to *instance*. Alternatively, a Multispectral Material Sensor [19] could be used, but would only support a recognition scope of *class*. To sense properties such as roughness, softness, and friction, a texture sensor [35] could be used, but may require the user to touch the surface in specific ways.

Sensing Contact

To sense contact, a hardware switch, which connects a circuit when being pressed by a touching force, may be most accurate. While this would enable explicit sensing of contact, it would require the device to have an additional single-purpose sensor. Contact could instead be inferred from an audio sensor [25], but accuracy may not be as high. Alternatively, contact could be inferred from an optical sensor that is already present to sense movements and material.

Output

In the simplest form, the device could provide no output at all to the user. However, this could be severely limiting, especially in cases where periphery devices that provide feedback are not available. Tactile feedback is possible but may come with high power consumption, and would require an additional component. One or more LEDs could be used to display bit-wise low resolution information [36, 23]. If the device used optical sensing, it may already be equipped with an LED, preventing the need for additional components. Richer forms of output could be provided by a speaker or LCD display, but would require larger form factors and greater power consumption.

Form Factor

The device needs to be small so that it can fit on the users' finger. Ideally, the device would be undetectable when dormant [36]. There are a number of ways which the device could be affixed to the finger. It could be embedded on a ring or thimble like structure worn on the tip of the finger (Figure 2a). This would allow users to remove the device when desired, or twist it to deactivate sensing. Alternatively, if small enough, the device could potentially be embedded under the finger nail (Figure 2b), on the surface of the fingertip skin, or implanted under the skin with exposed components for sensing (Figure 2c) [36]. A larger form factor could be worn above the finger. However this may cause offset problems in the sensing, and could be susceptible to occlusion problems.

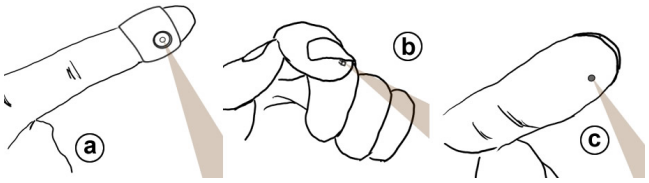

Figure 2: Possible form factors for the hardware.

MAGIC FINGER

After carefully considering the aspects of the hardware design space, we developed Magic Finger, a proof-of-concept prototype that enables always-available input through *finger instrumentation*. In this section we describe our implementation, and the basic capabilities of the device.

Hardware Implementation

To achieve sensing of positional movements *and* sensing of material, our prototype uses two optical sensors. A low resolution high frame rate optical flow sensor is used to sense 2D finger movements, and a higher resolution, lower frame rate camera is used to sense material (Figure 3).

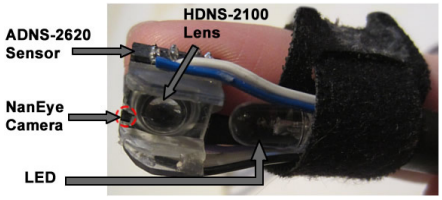

Figure 3. The Magic Finger Device.

Optical Flow Sensor

We used an ADNS 2620 optical flow sensor with a modified HDNS-2100 lens. This sensor is often used in optical mice, and is reliable on a large variety of surfaces. The sensor detects motion by examining the optical flow of the surface it is moving on. The ADNS 2620 sensor is a low-resolution black and white camera (18×18 $63\mu m^2$ pixels). Our initial tests showed that the sensor could be used to recognize textures, but only at the *class* scope of recognition, and only with a very small number of items (4).

Micro RGB Camera

To provide an enhanced scope of recognition, we used an AWAIBA NanEye micro RGB camera [4]. In comparison to the optical flow sensor, NanEye has a higher resolution and smaller pixels, and thus captures more subtle details of textures. The micro camera is 1mm × 1mm × 1.5mm (Figure 4). It captures color images at a resolution of 248 × 248, 3 μm^2 pixels, at 44 fps. For image processing, we converted the signal to greyscale. While it would be desirable to also use the NanEye for sensing optical flow, its low frame rate makes this difficult. We expect future solutions could be achieved with a single miniature sensor.

We embedded the NanEye on the edge of the optical flow sensor. To prevent occlusions from wiring, we crop to only use 70% camera's view to 175 × 175 pixels (Figure 4).

Light Source

In order for the camera and the optical flow sensor to function, an external light source is needed to illuminate the surface underneath it. We used a 5mm white LED, which has similar size and brightness with those used in optical mice. The LED can also be conveniently used for output.

Physical From

Magic Finger is affixed to the fingertip using an adjustable Velcro ring. A small plastic casing holds the components. The case was designed to control the distance between the RGB camera and surface during contact (Figure 1).

Computer Interfacing

The optical flow sensor and the LED were connected to an Arduino UNO board [3], which facilitates communication with the computer, an HP TouchSmart Tm2 running Windows 7. The NanEye is connected directly via USB. Tethering the device to a computer was an enabling technology for our prototype, and not our vision for future implementations. See Future Work for a discussion of power and communication capabilities required for an untethered implementation.

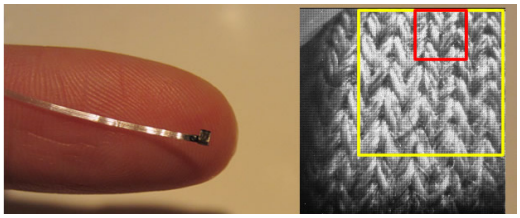

Figure 4: Left: the NanEye camera. Right: the field of view is a 175 x 175 rectangle (yellow region). The red region is used to detect changes in contrast.

Device Capabilities

Relative Positional Movement

Positional movement data is obtained directly from the optical flow sensor using the Arduino. Our program receives (X,Y) coordinates a resolution of 400cpi at 15 fps.

Sensing Contact

Magic Finger uses the micro camera to sense contact with a surface. Rapid changes in contrast (due to reflection of the illuminator) are used to signal changes in contact. To detect contact, we continuously calculate the overall contrast of a 60x60 pixel square of the sensor image, by averaging the square difference between each pixel and its neighbors (in grayscale space). When the Magic Finger is more than 5mm from a surface, light from its built-in LED is not reflected back, and a uniformly darker image is seen. A change within a small window of time form this darker image to a brighter image is measured against threshold values of our contrast metric for changes downward (lift-up) and upward (touch-down) (see Figure 5).

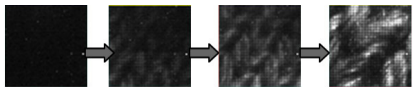

Figure 5. Contrast increases with the finger approaches a surface. Leftmost: the Magic Finger is more than 5mm from a surface. Rightmost: the Magic Finger is in contact with the surface

Identifying Material

When Magic Finger detects a contact event, the current frame of the RGB Camera is used for. texture recognition. Textures are classified using a machine learning algorithm. The first step is to describe the texture using a feature vector. In our implementation, we used Local Binary Patterns (LBP) to retrieve the unique features [37]. The algorithm detects 10 microstructures inside a texture (e.g. edges, lines, spots, flat areas). The histogram of the 10 microstructures uniquely describes a texture. This algorithm is orientation invariant, allowing recognition independent of the angle which the finger touches a surface. The feature vectors are also gray-scale invariant, making it robust in different lighting conditions. To train the classifier, we used Chang and Lin's LIBSVM algorithm using a Support Vector Machine (SVM) [13]. Before we trained the model, we tuned the required SVM parameters that gave high cross-validation scores.

Environmental vs. Artificial Textures

We classify the types of textures Magic Finger can sense as *environmental* and *artificial*. *Environmental* textures are naturally occurring in the user's environment, such as a table or shirt. In contrast, a*rtificial* textures are explicitly created for the purpose of being used by Magic Finger. We found that a simple way to create textures was to print a grid of small ASCII characters (Figure 6).

Fiduciary Markers

The above classification of textures allows Magic Finger to increase its scope to *class* recognition. If we are to further increase the scope to *instance* recognition, further capabilities are required. A common approach to tag specific objects is to use bar codes or fiduciary markers [2, 15]. We used the Data Matrix code, a two-dimensional barcode consisting of a grid of black and white cells. The Data Matrix codes we used were 10×10 cells, representing numerical values between 0 and 9999.

To be properly recognized, the entire Data Matrix must be in the field of view. To ease targeting, we print a cluster of identical Data Matrix codes, each exactly ¼ the size of the

RGB camera's field of view to ensure at least one code is fully visible. A library is used to decode the tags [1]. When Magic Finger detects a contact event, an API call is made to decode the frame. This call takes approximately 240ms, and is performed before texture classification occurs.

A future implementation of Magic Finger could utilize grids of fiduciary markers, such as is done by Anoto, in place of the ASCII-based artificial textures we designed. Nevertheless, we found that the ASCII-based technique was an effective way to easily generate artificial-textures.

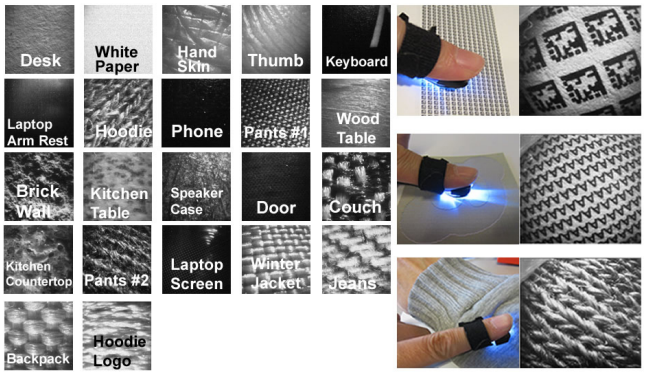

Figure 6: Left: 22 environmental textures used in the study. Right: Data Matrix, artificial texture, and environmental texture, as seen by a human and by the NanEye.

SYSTEM EVALUATION

We conducted a study in order to evaluate the accuracy at which Magic Finger can recognize *environmental* textures, *artificial textures*, and Data Matrix codes. This study was meant to serve as a technical evaluation of Magic Finger's capabilities, not a user study of the device.

For the *environmental* textures, we collected 22 different textures from a large variety of everyday objects, including public items, personal items, and parts of the user's body. The textures we sampled are shown in Figure 6.

To create *artificial textures*, we printed a grid of ASCII characters in black Calibri font, with a font size of 2pt and a line space of 1.6pt. The textures were printed on a Ricoh Aficio MP C5000 laser printer at 1200 DPI. We chose 39 characters found in a standard US English keyboard, including all 26 English capital letters (*A - Z*) and 13 other characters [*~ ! @ # $ % ^ & (- + = . "*]. For the data matrix, we randomly selected 10 out of 10,000 possible codes: 1257, 3568, 9702, 3287, 4068, 5239, 8381, 6570, 0128, and 2633, with each printed clusters of 72 × 15 identical codes.

For each *environmental* texture, *artificial* texture, and Data Matrix, 10 samples were collected twice a day for 3 days (as in [19]). Samples in the same class were obtained from the same object but from different locations. In total, we collected 60 samples for each material. The accuracy of recognition was tested using a 5-fold cross validation procedure. Because training and testing data may include points that were sampled in an adjacent time (e.g. points that tend to be similar), we randomized the order of the points prior to the test [19]. The Data Matrix recognition was evaluated by calculating the accuracy of decoding.

Results

Cross validation achieved an accuracy of 99.1% on the 22 tested *environmental* textures. This is a promising result given that we were only using 175 × 175 pixels (e.g. 70% of the camera's view). To better understand how the classification accuracy is affected by the resolution of the camera, we cropped the sample images from their center into smaller images of 18px^2, 70px^2, and 122px^2. The results of the 5-fold cross validation are shown in Figure 7, left. Samples of 70px^2 achieved 95% accuracy across the 22 textures. This gives future engineers some flexibility when choosing sensors for finger instrumentation.

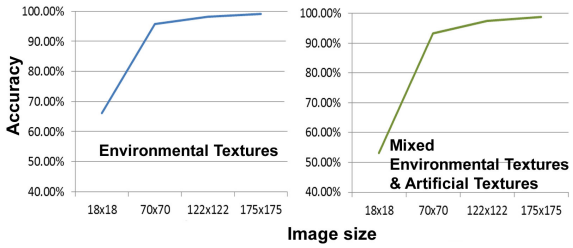

Figure 7: Cross validation accuracy. Left. On *environmental* textures. Right. On and mixed *environmental* and *artificial* textures.

For the ASCII *artificial* textures, cross validation yielded 83.8% accuracy with all the 39 tested characters. Given the large number of textures, this result is also promising. The lower accuracy is due to the larger number of textures, and also because the different textures are less visually distinguishable than the *environmental* textures we selected.

To see how the number of textures impact accuracy levels and find an optimal set of characters to use, we performed another 4 iterations of tests by progressively removing textures that resulted in lower accuracy rates. The accuracy levels through each iteration were 86.4% for 36 textures, 89.9% for 31 textures, 97.9% for 13 textures, and 99.5% for 10 characters [B G I ! # % ^ (+ .].

The above results indicate that Magic Finger can recognize 22 *environmental* textures *or* 10 *artificial* textures with an accuracy of above 99%. We also tested accuracy levels when combining these two sets, for a total of 32 textures. The Cross validation accuracy was 98.9% using the 175x175 pixel image. We also tested the classification accuracy with the smaller cropped images. The results of the 5-fold cross validation are shown in Figure 7, right.

The Data Matrix codes were correctly decoded for 598 of the 600 samples we collected, which yields an accuracy of 99.7%. In both failure cases (one for 3568, one for 9702) the API reported that no Data Matrix was recognized.

Summary

The results of our evaluation are very promising. Accuracy levels indicate that Magic Finger would be able to distinguish a large number of both *environmental* and *artificial* textures, and consistently recognize Data Matrix codes.

Compared to Harrison and Hudson's multispectral material sensing, which was able to classify 27 materials with an accuracy of 86.9% [19], we have obtained an accuracy of 98.9% across 32 textures. Overall, this demonstrates the feasibility of using a Magic Finger device for interactive tasks. In the following sections, we explore the interaction techniques and usage scenarios that Magic Finger supports.

INTERACTION DESIGN SPACE

Having developed a robust device, we sought to develop interaction techniques to facilitate its use. Here, we define the interaction design space for Magic Finger using 5 dimensions, described below.

Input Texture

The input textures vary according to the degree of specificity and information bandwidth they allow. We differentiate our interaction design space by 3 types of textures the user may encounter: *environmental* textures are textures of real objects, unmodified for Magic Finger; *artificial* textures refer to textures engineered for the Magic Finger, as we have discussed previously; and *dynamic* textures are those that change over time. Dynamic textures, for example, can be used to communicate between 2 Magic Finger devices.

Input Type

The possible input type depends on the specific capabilities of an implementation of Magic Finger. That is, our implementation is limited to having the Magic Finger act as a 2-state input device. Thus, input actions can either be *tap input* (sensing contact), or *gesture input* (positional movement).

Texture to Result Mapping

The ability to recognize different textures allows contextual actions to be carried out by the Magic Finger. The mappings of texture to result vary by the degree with respect to their artificiality: *Intrinsic* mappings rely on the semantics of the touched object to define the mapping. For example, tapping on the empty space of a phone always triggers a function of the phone, e.g. fast dial a number. Intrinsic mappings may suffer from a lack of flexibility, and conflicts in mappings. In contrast, *extrinsic* mappings associate commands to unrelated physical objects. Such mappings are flexible but may be less discoverable or memorable.

Resulting Action Type

Many types of actions could result from user input. We classify three main types: perform a *command execution*, provide *continuous control*, or serve as a *mode delimiter*.

Feedback

Based on needs, feedback can be *internal* or *external* to Magic Finger. For example, *internal* feedback could provide information using an LED while richer e*xternal* feedback could be provided by an attached or secondary device.

AUTHORING ENVIRONMENT

We created an application to help users manage their registered and recognized textures (*environmental* and *artificial)* and their associated functional mappings. A window displays a list of the defined texture classes (Figure 8 left). Each texture class in the list contains a sample image of the texture and a user defined title for that texture class. Users can double-click the sample images to view all the samples of that object that are used in training the SVM model. A button in the user interface can be used to create a new texture class. Users can also add samples to a new or existing texture class as input to the recognizer.

Each texture class also has an associated dropdown menu, which shows a list of functions which are preset resulting actions that could be mapped to any texture class (Figure 8 middle). For the purposes of demonstration, we have predefined a variety of resulting actions which can be mapped to Magic Finger input. In some cases, the authoring environment allows the user to adjust parameters of an action. For example, if the user selects the "Send SMS" action (which automatically sends an SMS message) from the drop down list, a pop up dialog is displayed where the user can enter a recipient number and message. To automatically launch external applications, the user can select "external application" from the action drop down list. The user can then drag-and-drop an icon of a desired application onto the menu item (Figure 8 right). This allows the user to link Magic Finger actions to existing applications, as well as to define their own scripts in their preferred languages and link them to Magic Finger.

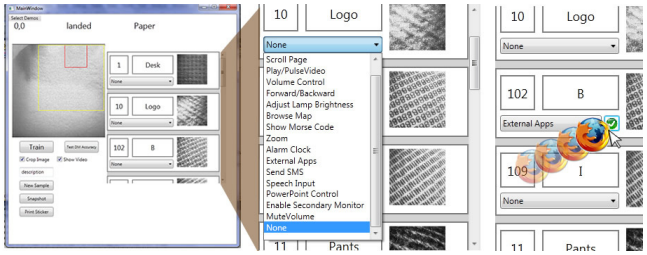

Figure 8: Left: list of texture classes. Middle: the drop down list of functions; Right: external applications

Authoring Artificial Textures

The authoring application also allows users to generate ASCII-based *artificial* textures which can be used, for example, as stickers to be placed on objects. In these textures, the individual ASCII characters are so small that they are indiscernible to the human eye, and can be used as pixels to create two-colour graphical images. While each "pixel" is the same character, the pixels can take-on a foreground or background color. We chose two colors that are distinguishable by the human eye, but are indistinguishable in greyscale during recognition (right).

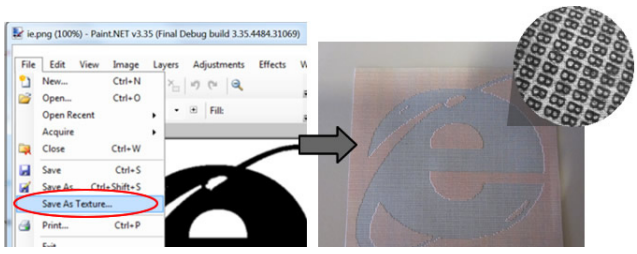

Figure 9: Left: the paint.net application; Right: the resulting texture.

To author these textures, we modified Paint.NET, an image editing application, to include a "*save as texture*" option (left). This option prompts the user to enter a character, and then converts the current image into an HTML document consisting entirely of that character rendered in either the foreground or background color. Such images can be used to give the *artificial textures* user friendly appearances, such as icons representing their functions.

INTERACTION TECHNIQUES

In order to explore the interaction design space we have described, we implemented a number of interaction techniques. Each of the following techniques serves as an exemplar of a point in our design space, as well as a portion of a coherent set of possible uses for Magic Finger. Some of the interactions are novel while others show how Magic Finger can be used to implement previously published techniques that required numerous different hardware configurations. In the example interaction techniques below, we explicitly mention the corresponding design dimension.

Some of the below interactions involve multiple Magic Fingers. To implement these scenarios, we constructed a second Magic Finger without the RGB camera. This device was unable to sense textures, but was sufficient for demonstrating the desired multiple-device interactions.

Tap Input

PocketTouch

Magic Finger can be used to remotely access or control mobile devices. We implemented a resulting action of muting an incoming Skype call (*command execution*), and mapped this to touching a handbag. If the user receives an unwanted call when their Smartphone is in the handbag, the user can simply tap the handbag to mute the notification. This interaction is much like PocketTouch [40], but does not require a phone's capacitive sensor to be placed in a special mode, or require the phone to be placed at a specific location (Figure 10a).

Tap-to-Talk

Because Magic Finger can identify the object that users touch, tapping different surfaces can trigger different *command executions*. In our implementation, the empty surface of a phone is used as a shortcut 'button' to send a frequent SMS message to a predefined recipient, e.g. *"I'm on my way home."* (b). This is an example of an *intrinsic mapping*. Users may also take advantage of inherent features of physical artifacts to define input textures. We created an *extrinsic mapping* of tapping on the logo of a t-shirt, to launching the Windows Voice command app. Thus, tapping on this special area of the shirt serves as a *mode delimiter* for entering speech input (Figure 10c).

Skinput

By recognizing skin as a texture, Magic Finger provides a method for using skin as an input surface, without requiring additional sensors. We utilized this to create an "am I late?" gesture – when the user taps their wrist (where a watch might have been – *intrinsic mapping*), Magic Finger checks for an upcoming appointment and either displays it on-screen (*external feedback),* or blinks its built-in LED (*internal feedback*) to indicate "time to head out!" (Figure 10d).

Pinching Finger to Thumb

The finger to thumb pinch action has been shown to be a useful gesture for freehand input [46]. By recognizing the thumb as an *environmental* texture, Magic Finger can easily detect a pinch when the thumb is tapped (Figure 10e). We mapped pinch to a *command execution* of advancing slides in a presentation, as a replacement for holding a wireless presentation mouse.

Data Matrix Buttons

In addition to tapping *environmental* textures, users can also tap *artificial* ones. A grid of printed Data Matrix codes can be used as physical buttons. The decoded numeric data can be used to trigger a command or to retrieve contextual information. In our implementation, we assigned specific numbers to PowerPoint documents. A user can print the

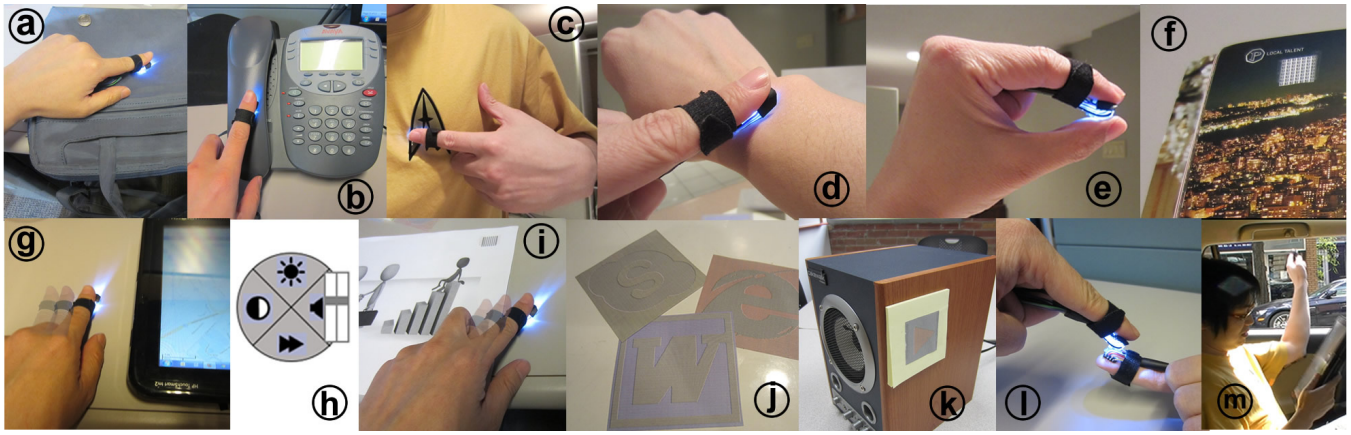

Figure 10. Tap and Gestural-based Interactions. (a) Tap on the bag to mute a Skype call; (b) Tap on the phone to send a frequent SMS message; (c) Using the logo of a t-shirt as a mode delimiter; (d) Tap the wrist to check an upcoming appointment; (e) Pinch gesture (f) Data Matrix Buttons; (g) Occlusion Free Input; (h) FaST slider widget; (i) Multi-surface crossing gesture; (j) Application launch pad; (k) Sticker as disposable remote controls; (l) Passing texture to another Magic Finger; (m) Use Magic Finger as a periscope to check out traffic.

PowerPoint slides with the associated Data Matrix code at the bottom of each page. Tapping the code opens the presentation on the user's computer. We also associated codes on magazine articles with webpages (Figure 10). Tapping the Data Matrix opens a URL associated with the article.

Gesture Input

Occlusion Free Input

Occlusion Free Input allows users to interact with a touch screen device by gesturing on a nearby surface. This eliminates the occlusion of the screen by a users' hand. We implemented a navigation application, where users can pan or zoom a map displayed on a tablet device (*continuous control)*. By mapping this action to the back of a tablet, users can carry out back-of-device input [45], without needing a specially instrumented tablet device. Additionally, SideSight [12] can be emulated by mapping the action to the table that the device is resting on (Figure 10g).

FaST Sliders

We implemented a FaST slider widget to control volume and playback of multi-media [33] (Figure 10h). FaST sliders allow manipulation of multiple, discrete or continuous parameters from a single menu. Once activated, the user can select a parameter to manipulate with a directional mark. While the finger remains down, the user can then adjust the value of the selected parameter by dragging. If the user is close to a display device, visual feedback for the menu can be displayed onscreen (*external feedback*). When a display is not available, we blink the Magic Finger's LED to indicate when a parameter has been selected, so the user knows to begin parameter adjustment (*internal feedback*).

Multi-Surface Crossing Gestures

Magic Finger can detect a crossing gesture from one surface to another (Figure 10i). This is accomplished by performing texture recognition any time the finger dwells while still in contact with a surface. We mapped this gesture to activating a slide deck on a projector in a meeting room. The user can tap a Data Matrix on their slide printout to open the PowerPoint, and tap the table to connect the laptop to a secondary monitor (i.e. projector). If a crossing gesture is detected from the Data Matrix to the table, the PowerPoint window is moved from the laptop display to the projector. In this case, the crossing gesture has an *intrinsic mapping*, as it represents a relationship between the two mapped surfaces. These multisurface gestures are similar to stitching [27], but do not require inter-device communication between the surfaces.

Input Stickers

Artificial textures can be used to increase the number of distinguishable surfaces in an area. By printing the artificial textures on adhesive strips, we create Input Stickers. We describe the associated interaction techniques below.

Application Launch Pad

We printed a set of Input Stickers that take-on the appearance of desktop icons (Figure 10j). The user can place these stickers in a convenient location in their work area, and tap the stickers to launch the associated desktop application. This emulates functionality shown in MagicDesk without requiring the entire desk surface to be a touch sensitive surface [11].

Disposable Remote Controls

Input stickers can also be used as remote controls. We printed a stack of stickers and put them beside the physical on-off switch for the speakers in a public room (Figure 10k). When entering a room, a user can pick up a sticker before sitting down. Tapping the sticker plays and pauses music on the room's sound system. When leaving the room, the user can replace the sticker on the stack, or dispose of it.

Additional Features

Identify Awareness

An interesting property of Magic Finger is that it is inherently user-identity aware. We configured the music remote control, described above, to play a user's favorite type of music, based on the identity of the user tapping the sticker. Traditional touch devices are not able to recognize users, without specialized hardware and making potentially error prone inferences [39, 50].

Passing Textures

In some cases, a user may want to virtually pass a texture that has been read by the Magic Finger to a remote receiver or to another user. We use the pinch gesture as a *mode delimiter* for "picking up" a texture. If a pinch is detected immediately after a Data Matrix is tapped, then the Data Matrix code is temporarily stored. When the pinch is released, the LED emits a Morse code pattern, representing the 4 digit number associated with the Data Matrix. We implemented a simple product scanner application that uses a webcam to read the Morse code and display information about the associated item, such as its price. We also implemented a second Magic Finger that can read the Morse code as a *dynamic texture* and carry out the associated functions of the Data Matrix code after it is passed to the user from the source user (Figure 10l).

The Periscope

People often use their fingers to sense areas that they cannot see. For example, we may use our hand to search for a pen dropped between the cushions of a couch. Magic Finger allows users to enhance the capabilities of such probing. By viewing its real-time video feed on a handheld device, the Magic Finger becomes an extension of the user's eyes. For example, a user can look behind them, around a corner or out the window of a car by simply pointing their finger in the associated direction (Figure 10m).

DISCUSSION AND FUTURE WORK

Having used Magic Fingers, we have made several observations which may be of benefit to those seeking to implement one. In this section, we discuss some of the insights we have gleaned from our experiences.

Device Size and From Factor

While our form factor is small enough to fit on a fingertip, it is still larger than what we envision for the future.

The hardware does have room to be made much smaller. Our prototype is limited by the size of the ADNS 2620 chip. The chip was designed to be used in a mouse, and so its form factor is optimized to fit with the other components of the mouse. The core sensor of the chip, however, is only 2 × 3 mm wide. Therefore, reengineering the chip to fit the sensor will significantly reduce the size of Magic Finger.

The NanEye camera has an ideal form factor. However, it is limited by its frame rate. Ideally, Magic Finger would need only one optical sensor to handle all of its functionality. For optical flow to be detected reliably, the sensor would need to have a frame rate of approximately 1500fps. Based on our evaluation, the resolution should be at least 70 × 70px to enable both optical flow and texture recognition. Given existing technologies, we are optimistic that a sensor matching these specifications will soon be available.

The other component that would need to be miniaturized is the LED. Micro LED's are available today; testing would be needed to determine minimum brightness requirements.

Power and Communication

A limitation of our hardware is that it is tethered to a nearby host PC. This simplified our implementations, but for Magic Finger to become a completely standalone device, power and communication needs to be considered. Holz et al. provide a thorough review of potential technologies that can be used for power and communication in a micro form factor [29]. Powering can be accomplished through rechargeable batteries, or harvesting power from the body or the environment [29]. Communication can be provided through a Bluetooth protocol, which consumes little power. Processing responsibilities could be performed by a Bluetooth-tethered mobile phone.

Evaluations

We presented a technical evaluation that demonstrated promising results for sensing accuracy of the Magic Finger. The results should be considered a high bar of what a Magic Finger could achieve, since the samples were collected under controlled conditions. Future evaluations should look at how differences in contact angles may impact captured image quality. Design considerations of the form factor of the device should aim to facilitate capturing good quality samples.

Magic Finger could also benefit from more formal user studies to understand how the device would be used by end users. For example, a user study may help us find a preferred location to mount the device, or identify important topics we have not explored, such as the "Midas touch" and false activations.

Interactions

The interactions which we implemented and demonstrated should be considered exploratory in nature, and only represent a small sample of what is possible with the Magic Finger. In our actual implementation, issues such as conflicts between operating modes and resulting actions would require a more careful examination. Introducing aids to help users learn or remember mappings would also be a fruitful topic for future work.

Magic Finger in its current implementation is limited in absolute positioning. One known technique to enable absolute positioning for artificial textures would be to encode location into the textual pattern (eg: Anoto [2]). For natural textures, absolute positioning may never be possible, although with improved sensing, location could be sensed at higher granularity (e.g. recognize locations of skin rather than simply "skin").

CONCLUSION

In this paper, we introduced the concept of finger instrumentation and our prototype Magic Finger, a novel device that extends users' touch capability to everyday objects. Instead of requiring instrumentation of the user's environment, or external cameras which may be prone to occlusion, Magic Finger is unique in that the finger itself is instrumented to detect touch. Our system evaluation showed that Magic Finger can recognize 32 different textures with an accuracy of 98.9%, allowing for contextual input. We presented a design space of interactions enabled by Magic Finger, and implemented a number of interaction techniques to explore this design space. Future work will focus on investigating topics such as "Midas touch" and false activations, and on understanding how users will use the Magic Finger. We will also explore hardware solutions for miniaturizing the form factor of the Magic Finger.

REFERENCES

[1] 2D Technology *http://www.2dtg.com/*

[2] ANOTO *http://www.anoto.com/*

[3] Arduino *http://arduino.cc*

[4] AWAIBA *www.awaiba.com*

[5] Finger Mouse *http://www.logisyscomputer.com*

[6] Annett, M., Grossman, T., Wigdor, D. and Fitzmaurice, G. (2011). Medusa: A Proximity-Aware Multi-touch Tabletop.*ACM UIST*. 373-382, 2011.

[7] Avrahami, D., Wobbrock, J. O. and Izadi, S. (2011). Portico: Tangible interaction on and around a tablet. *ACM UIST*. 347-356.

[8] Bau, O., Poupyrev, I., Israr, A. and Harrison, C. (2010). TeslaTouch: electrovibration for touch surfaces. *ACM UIST*. 283-292, 2010.

[9] Baudisch, P., Sinclair, M. and Wilson, A. (2006). Soap: a Pointing Device that Works in Mid-Air. *ACM UIST*. 43-46.

[10] Bellavista, P., Corradi, A., Fanelli, M. and Foschini, L. (2013). A Survey of Context Data Distribution for Mobile Ubiquitous Systems. *ACM Computing Surveys*. 1-49, 2013.

[11] Bi, X., Grossman, T., Matejka, J. and Fitzmaurice, G. (2011). Magic Desk: Bringing Multi-Touch Surfaces into Desktop Work. *ACM CHI*. 2511-2520, 2011.

[12] Butler, A., Izadi, S. and Hodges, S. (2008). SideSight: multi-"touch" interaction around small devices. *ACM UIST*. 201-204.

[13] LIBSVM--A Library for Support Vector Machines. *http://www.csie.ntu.edu.tw/~cjlin/libsvm*.

[14] Chen, G. and Kotz, D. (2000). A Survey of Context-Aware Mobile Computing Research (TR2000-381).

[15] Cheng, K.-Y., Liang, R.-H., Chen, B.-Y., Laing, R.-H. and Kuo, S.-Y. (2010). iCon: utilizing everyday objects as additional, auxiliary and instant tabletop controllers. *ACM CHI.* 1155-1164, 2010.

[16] Cohn, G., Morris, D., Patel, S. N. and Tan, D. S. (2011). Your noise is my command: sensing gestures using the body as an antenna. *ACM CHI.* 791-800, 2011.

[17] Gustafson, S., Holz, C. and Baudisch, P. (2011). Imaginary Phone: Learning Imaginary Interfaces by Transferring Spatial Memory from a Familiar Device. *ACM UIST* 283-292, 2011.

[18] Harrison, C., Benko, H. and Wilson, A. D. (2011). OmniTouch: Wearable Multitouch Interaction Everywhere. *ACM UIST.* 441-450, 2011.

[19] Harrison, C. and Hudson, S. E. (2008). Lightweight material detection for placement-aware mobile computing *ACM UIST.* 279-282, 2008.

[20] Harrison, C. and Hudson, S. E. (2008). Scratch input: creating large, inexpensive, unpowered and mobile finger input surfaces. *ACM UIST.* 205-208, 2008.

[21] Harrison, C. and Hudson, S. E. (2009). Abracadabra: wireless, high-precision, and unpowered finger input for very small mobile devices *ACM UIST.* 121-124, 2009.

[22] Harrison, C. and Hudson, S. E. (2009). Texture displays: a passive approach to tactile presentation. *ACM CHI.* 2261-2264, 2009.

[23] Harrison, C., Lim, B. Y., Shick, A. and Hudson, S. E. (2009). Where to locate wearable displays?: reaction time performance of visual alerts from tip to toe. *ACM CHI.* 941-944, 2009.

[24] Harrison, C., Ramamurthy, S. and Hudson, S. E. (2012). On-Body Interaction: Armed and Dangerous. In *Proceedings of the International Conference on Tangible, Embedded, and Embodied Interaction* 19 - 22, 2012.

[25] Harrison, C., Schwarz, J. and E., H. S. (2011). TapSense: Enhancing Finger Interaction on Touch Surfaces. *ACM UIST.* 627-636, 2011.

[26] Harrison, C., Tan, D. S. and Morris, D. (2010). Skinput: appropriating the body as an input surface. In *ACM CHI.* 453-462, 2010.

[27] Hinckley, K., Ramos, G., Guimbretiere, F., Baudisch, P. and Smith, M. (2004). Stitching: pen gestures that span multiple displays. In *Proceedings of the AVI.* 23-31, 2004.

[28] Holleis, P., Schmidt, A., Paasovaara, S., Puikkonen, A. and Häkkilä, J. (2008). Evaluating capacitive touch input on clothes In *Proceedings of the Mobile HCI.* 81-90, 2008.

[29] Holz, C., Grossman, T., Fitzmaurice, G. and Agur, A. (2012). Implanted User Interfaces. *ACM CHI.* 2012.

[30] Kane, S. K., Avrahami, D., Wobbrock, J. O., Harrison, B. L., Rea, A. D., Philipose, M. and Lamarca, A. (2009). Bonfire: a nomadic system for hybrid laptop-tabletop interaction *ACM UIST.*. 129-138, 2009.

[31] Lévesque, V., Oram, L., MacLean, K. E., Cockburn, A., Marchuk, N. D., Johnson, D., Colgate, J. E. and Peshkin, M. A. (2011). Enhancing physicality in touch interaction with programmable friction. *ACM CHI.* 2481-2490, 2011.

[32] Marquardt, N., Kiemer, J. and Greenberg, S. What Caused That Touch? Expressive Interaction with a Surface through Fiduciary-Tagged Gloves. . In *Proceedings of*.

[33] McGuffin, M., Burtnyk, N. and Kurtenbach, G. (2002). FaST Sliders: Integrating Marking Menus and the Adjustment of Continuous Values. *GI.* 2002.

[34] Mistry, P. and Maes, P. (2009). SixthSense: a wearable gestural interface. In *Proceedings of the CHI EA.* 2009.

[35] Mukaibo, Y., Shirado, H., Konyo, M. and Maeno, T. (2005). Development of a Texture Sensor Emulating the Tissue Structure and Perceptual Mechanism of Human Fingers In *Proceedings of the International Conference on Robotics and Automation.* 2565-2570, 2005.

[36] Ni, T. and Baudisch, P. (2009). Disappearing mobile devices *ACM UIST.*. 101-110, 2009.

[37] Ojala, T., Pietikäinen, M. and Mäenpää, T. (2002). Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24 (7), 971-987, 2002.

[38] Richter, S., Holz, C. and Baudisch, P. (2012). Bootstrapper: Recognizing Tabletop Users by their Shoes. I *ACM CHI. (in press).* 2012.

[39] Richter, S., Holz, C. and Baudisch, P. (2012). Bootstrapper: Recognizing Tabletop Users by their Shoes. *ACM CHI. (in press).* 2012.

[40] Saponas, T. S., Harrison, C. and Benko, H. (2011). PocketTouch: Through-Fabric Capacitive Touch Input. *ACM UIST.*. 303-308, 2011.

[41] Saponas, T. S., Tan, D. S., Morris, D., Balakrishnan, R., Turner, J. and Landay, J. A. (2009). Enabling always-available input with muscle-computer interfaces. *ACM UIST.* 167-176, 2009.

[42] Sato, M., Poupyrev, I. and Harrison, C. (2012). Touché: Enhancing Touch Interaction on Humans, Screens, Liquids, and Everyday Objects. *ACM CHI.*

[43] Sturman, D. J. and Zeltzer, D. (1994). A Survey of Glove-based Input. *IEEE Computer Graphics and Applications*, 14 (1), 30-39, 1994

[44] Weiss, M., Wacharamanotham, C., Voelker, S. and Borchers, J. (2011). FingerFlux: Near-surface Haptic Feedback on Tabletops. *ACM UIST.* 615-620, 2011.

[45] Wigdor, D., Forlines, C., Baudisch, P., Barnwell, J. and Shen, C. (2007). Lucidtouch: a see-through mobile device. *ACM UIST.* 269-278, 2007.

[46] Wilson, A. (2006). Robust Vision-Based Detection of Pinching for One and Two-Handed Input. *ACM UIST.* 255-258, 2006.

[47] Wilson, A. D. (2010). Using a depth camera as a touch sensor. *Tabeltop.* 69-72, 2010.

[48] Wilson, A. D. and Benko, H. (2010). Combining Multiple Depth Cameras and Projectors for Interactions On, Above, and Between Surfaces. *ACM UIST.*. 273-282, 2010.

[49] Wimmer, R. and Baudisch, P. (2011). Modular and Deformable Touch-Sensitive Surfaces Based on Time Domain Reflectometry. *ACM UIST.*. 517-526, 2011.

[50] Zhang, H., Yang, X. D., Ens, B., Liang, H. N., Boulanger, P. and Irani, P. (2012). See Me, See You: A Lightweight Method for Discriminating User Touches on Tabletop Displays. *ACM CHI. (in press).* 2012.