RemoteTouch: Touch-Screen-like Interaction in the TV Viewing Environment

Sangwon Choi, Jaehyun Han, Geehyuk Lee
Department of Computer Science
KAIST, Daejeon, South Korea
{sangwonchoi7, jay.jaehyun, geehyuk}@gmail.com

Narae Lee, Woohun Lee
Department of Industrial Design
KAIST, Daejeon, South Korea
{narae, woohun.lee}@kaist.ac.kr

ABSTRACT
We explored the possibility of touch-screen-like interaction with a remote control in the TV-viewing environment. A shadow representing the user's thumb touches the screen, presses a button, flicks a cover-flow list, and draws a simple stroke, while the thumb stays and moves on and above the touchpad. In order to implement the concept we developed an optical touchpad for tracking the thumb hovering over its surface, and designed a TV application to demonstrate possible new interaction styles. Throughout two iterations of prototyping, we corrected some of our false expectations, and also verified its potential as a viable option for a TV remote control. This paper presents technical issues and requirements for the hover-tracking touchpad and a complete report of our user studies to explore touch-screen-like interaction for the TV.

Author Keywords
RemoteTouch interaction, hover-tracking touchpad, TV user interface, TV remote control.

ACM Classification Keywords
H.5.2. Information interfaces and presentation: user interfaces – input devices and strategies

General Terms
Design, Human Factors, Experimentation

INTRODUCTION
The TV in the future will become a terminal for many interactive applications such as Video-On-Demand (VOD) services, online games, and online shopping services. Smart TV projects, such as Google TV, Apple TV, and Samsung Internet@TV, provide a foretaste of the future of TV. In response to the new needs of this future TV, and replacing the conventional array of buttons, new remote control designs incorporating alternative input technologies, such as joystick, touchpad, or direct-pointing, are being explored and evaluated. For example, remote pointing devices such as a gyro-mouse [11] proved to be more efficient than a remote with an isometric joystick. However, as with other laser-pointer style direct-pointing devices, a gyro-mouse had problems such as fatigue, jittering, and response delay introduced by cursor stabilization [17, 18]. A remote with a touchpad is another common idea [4], but currently lacks a comparative study on its usability. In the context of pointing on a computer interface, a touchpad was shown to be less efficient than an isometric joystick [3]. Such studies focused on pointing efficiency, a typical requirement for desktop user interface, but did not assess the usability of such devices in the context of a TV viewing environment. Based on these studies, it is still difficult to predict which design will prevail in the future TV environment.

CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.

In an effort to design a better remote control for a TV, we have been studying different options in parallel. A remote with a touchpad as in [4] was an attractive option since a touchpad is familiar to users and users seemed to adapt quickly to using it with the thumb. A major concern with a touchpad was the indirectness due to the relative cursor control. Another attractive option was a remote with a touchscreen, which provides direct-touch control. This is becoming a popular idea especially due to the rapid spread of smart phones, and the new possibility to use them to control the TV. Google TV remote applications for smart phones are examples for this. However, a major concern with a remote with a touchscreen was the problem of divided visual attention between the remote and the TV screen, which are usually separated by a few meters. In addition, a smart phone as a remote, whether it has a touchscreen or not, may create the mismatch between the phone as a personal mobile device and the TV as a shared domestic device; for example, parents will be reluctant to let their child use their phones.

After examining possible options for a TV remote control, we became interested in the idea of realizing touch-screen-like interaction. Instead of touching the TV screen, which is too far to reach, one touches a sensor area on a remote, which offers a one-to-one mapping to the TV screen. For instance, one can select a button on the screen by moving the thumb over the sensor area and landing the thumb on the button. Also, one spins a scrolling list on the screen by flicking on a corresponding spot on the sensor. Such interaction techniques are familiar to users thanks to the popularity of touch-screen smart phones. Of course, the question remains whether users will be able to perform such techniques indirectly on the touchpad and away from the screen. We speculate that users will be able to develop the

skill to map the movement of the thumb on the remote to the movement on the screen if good on-screen visual feedback of the thumb is provided. We named this type of interaction RemoteTouch, and the primary goal of the present research is an iterative exploration of both the interaction design and the technical implementation of RemoteTouch, as well as verifying its effectiveness with users.

In the rest of the paper, we describe our early explorations and two main iterations of prototyping the RemoteTouch concept. The main focus of the first iteration was on the design of a hover-enabled touchpad for the basic realization of the RemoteTouch concept. The second iteration then focused on the design and evaluation of RemoteTouch-style interfaces. Before reporting our explorations, we first give a brief overview of the background of TV remote controls as well as previous research related with the RemoteTouch concept.

BACKGROUND AND RELATED WORK

The first wired remote control 'Lazy Bones' was developed by Zenith and it was soon replaced by its successor 'Space Command' whose form factor resembled that of a modern remote control. The basic structure of a TV remote, consisting of a series of buttons, each dedicated to a specific function, has remained until these days [4]. With more diverse TV usage patterns, especially driven by the connection to the Internet, researchers became interested in new TV user interfaces. Most of them, however, merely transferred user interface technologies from desktop computing to the TV viewing environment [4, 9, 11], thus lacking research for a novel remote control concept that takes into account the special characteristics of the TV viewing environment.

Some of these special characteristics are discussed in previous research on TV user interface [5, 9, 10]. TV viewers usually operate a TV at some distance in a relaxed posture without a stable working surface. A remote control and the output of a TV are spatially separated by a few meters. This spatial separation circumvents users from reaching the screen directly and requires them to alternate the focus of attention between the remote and the TV. As the TV becomes more interactive, the usability problem caused by the spatial separation of input and display will worsen. Furthermore, TV viewers take whatever posture that is comfortable for them while watching TV, and therefore it is not acceptable to design a TV remote control that requires a stable working surface. Also, users should be able to use a TV remote single-handed. The concept of RemoteTouch is especially motivated by the problem of the spatial separation of input and display.

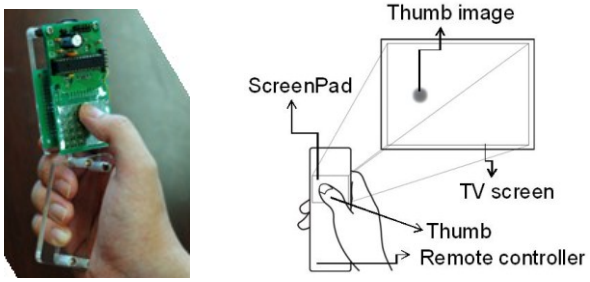

Figure 1. RemoteTouch controller (left) and the mapping from the RemoteTouch controller to the screen (right).

There are two intriguing user interface concepts that are closely related to that of RemoteTouch: EZ Touch Remote by Panasonic [19] and TactaPad by Tactiva [20]. The EZ Touch Remote is an effort to overcome the indirectness of controlling a TV. The EZ Touch Remote allows users to see a virtual thumb on the screen that represents their thumb. With such visual feedback users feel the sense of directness in handling the TV. The EZ Touch Remote, however, utilizes a normal touchpad, and therefore fails to maintain the "continuous visual feedback" of the thumb as the visual thumb cannot follow the real thumb when it lifts off the touchpad. Intended for the desktop environment, the TactaPad is pursuing a very similar goal to RemoteTouch. It uses a multi-finger touchpad and a camera so that users can see the movement of their hands on the screen interacting with virtual objects. It is meant for users to feel enhanced directness of manipulation as if they touched objects directly. A very similar concept was explored for interacting with large displays while sitting around a table using a computer-vision based touchpad with hover information [15]. Although these bare much similarity with of the concept of RemoteTouch, they were intended for desktop usage with fixed setup, therefore not suitable for the versatility of the TV viewing environment.

Figure 1 illustrates the concept of RemoteTouch. The remote has a sensor area that is one-to-one mapped to the TV screen. The TV screen provides visual feedback about the position and hovering height of the thumb on the sensor area. For instance, when the thumb is above the sensor area, a small disk representing the thumb appears with a shadow as if it is hovering on the screen. When the thumb touches and slides on the sensor area, the disk lands on the screen and drags the object under the circle. The user sees the thumb moving on and above the screen while the thumb is moving on and above the sensor area. Eventually, this will make the user feel like using a touch-screen remotely.

EARLY EXPLORATIONS

At first, we tried to prototype the concept of RemoteTouch using a normal touchpad without hover information, and a one-to-one mapping from the touchpad to the screen. After a pilot study, we soon realized that the prototype suffers from the lack of the "continuous visual feedback of the thumb". For instance, when drawing a gesture on the screen, the thumb position remains unknown until touching the touchpad. Also, the visual feedback of the thumb is lost as soon as the thumb is lifted to start another stroke. Of course, the continuous visual feedback of the thumb would have been possible if we designed the interaction so that the thumb always needs to stay on the touchpad. However, confining the thumb on a touchpad restricts possible thumb

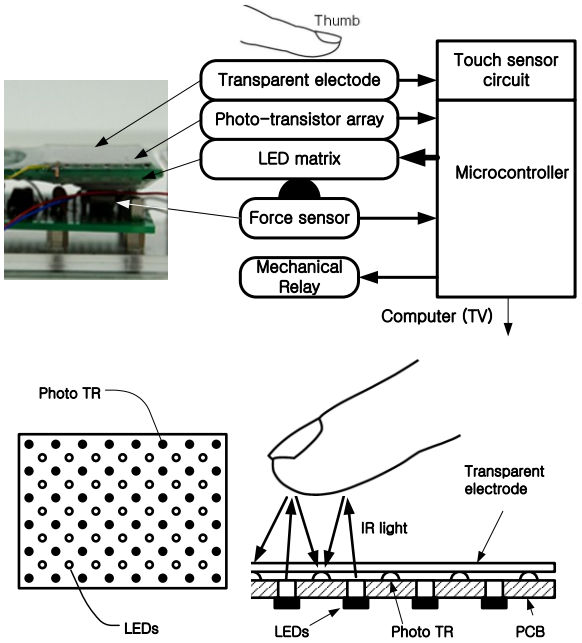

Figure 2. The structure of the ScreenPad: the functional blocks (top), and the top-view and the magnified cross-section of the optical sensor layer (bottom).

movements and gestures. We learned that we needed a hover-tracking touchpad that can sense the thumb above the surface as well as on the surface in order to realize the RemoteTouch concept.

We first considered capacitive sensing options, including GlidePoint and GlideSensor from Cirque [2], for hover-tracking, but could not find a satisfactory solution. For instance, GlideSensor in its highest sensitivity provided a sensing range of a few millimeters, which was too small to allow natural thumb movements. We observed that the thumb movement above the touchscreen can be as high as 10mm, which means that the sensor should have a hover-tracking range of at least 10mm. Hover-detection at this height may be possible with a capacitive sensing method, but, considering the physical nature of capacitive sensing, at this height we expected the image of the thumb being too diffuse for a precise localization of the thumb.

As a second option, we considered optical sensing methods using a camera. The prototype that we built was based on the principle of a diffusive IR touch screen [16]. We added an additional in-plane IR illumination to the prototype to enhance touch detection [6]. The resulting prototype successfully sensed the thumb, whether it was on or over the surface, yet a problem with the prototype was an excessive latency even though we used an IEEE 1394 camera with a frame rate of 60Hz. The visual feedback that we could achieve by processing the camera frames was not fast enough to make the visual feedback effective. The thumb by far outran the expected speed, causing a noticeable delay in its visual feedback. Another problem of the prototype was that it was too thick (about 6cm) to be housed in the size of a remote control.

FIRST PROTOTYPING

In order to overcome the latency problem of the camera-based prototype, we tested optical sensing using an LED array. The resulting optical touchpad, which we named as ScreenPad, finally satisfied our minimal requirements. Its sensing range was large enough to allow natural thumb movements (the sensor output at 10% of its maximum when the thumb is at 16mm from the pad), it was fast enough to allow a responsive visual feedback (about 12ms for sampling a frame and 4ms for transmission), and it was thin enough to be fitted in a remote mockup (about 5mm thick excluding a force sensor under the optical sensor). Only after building the ScreenPad prototype, we were able to start the experimentation of the RemoteTouch concept. Therefore, we will describe the implementation and experimentation of this prototype as our "first prototyping" and the next prototype in the following section "second prototyping".

Prototype Design

Hardware design

As shown in Figure 2, the sensor consists of three layers: a transparent conductor layer for touch sensing, a photo-transistor and LED matrix layer for position sensing, and a force sensor for thumb pressure sensing. Figure 2c shows the detailed structure of the second layer. The 5x7 LED matrix illuminates the thumb sequentially in the row-major order. The 6x8 photo-transistors, all of which are wired in parallel and act as a single planar sensor, measure reflection from the thumb. The operating principle is different from other implementations using a matrix of receivers [1] or a matrix of emitter-receiver pairs [7], but is similar to that of a laser barcode reader. Both use localized illumination instead of localized sensing to achieve imaging. We chose localized illumination since it is advantageous in terms of power conservation and chose global (non-localized) sensing since it is advantageous for sensing efficiency and sensitivity, i.e., more light scattered by the thumb can be gathered.

Using off-the-shelf components, the assembly of the sensor was rather simple except for the following three deliberations. First, the angular sensitivities of the LEDs and the photo-transistors had to be chosen carefully in order to obtain a smooth reflection image. The smoothness of the image is essential for the subsequent signal processing stage to produce sub-pixel resolution position data. Secondly, the placement of the LEDs and the photo-transistors had to be determined carefully in order to avoid reflection from the transparent cover (touch sensor electrode). Finally, the sensor circuit had to be designed to be resistant to AC line noise through capacitive coupling with the thumb. The 6x8 photo-transistors connected in parallel form a single planar photo-sensor with very low output impedance and therefore were almost insensitive to such electrical noise sources.

Signal processing

One of the main problems that we had to solve was the instability of the cursor due to the high control-display gain,

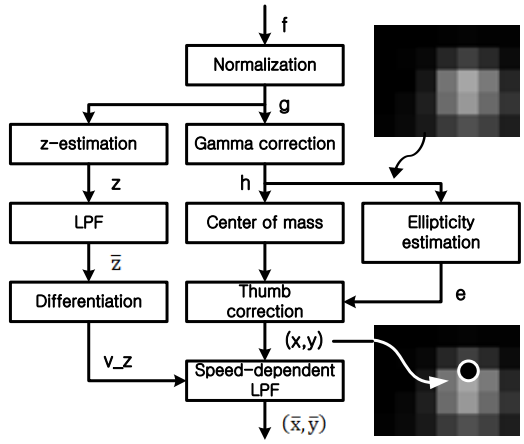

Figure 3. The signal flow diagram of ScreenPad.

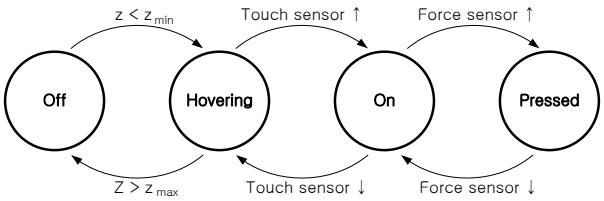

Figure 4. The state transition diagram of ScreenPad.

which is the consequence of a one-to-one mapping between the small sensor area and the large screen. A smoothing filter may reduce the instability but will also reduce the responsiveness of the device. In order to cope with the instability problem while preserving the responsiveness of the device, we designed a speed-dependent low-pass filtering (LPF) algorithm, where the strength of filtering is inversely proportional to the speed of the cursor. Another problem was the shift of the cursor during "down" movements. With a ScreenPad, one usually catches a target in two steps: moving to a point over a target and down to touch it. In the second step, one expects that the cursor will be stationary, but the cursor often moves and misses the target. The shift of the cursor during "down" movements may be due to the actual horizontal movements of the thumb or due to an error in the estimation of the center of the thumb. In either case, it was desired to reduce the cursor shift during down movements, and we answered this problem by incorporating the vertical velocity of the thumb into the speed-dependent LPF algorithm; the cursor was stabilized further when the thumb is in a rapid vertical motion.

Figure 3 shows the signal processing steps of the thumb image f from the sensor to the final determination of the thumb position $(\bar{x}, \bar{y})$. First, the raw sensor image f is transformed into the normalized value image $g \in [0,1]$. The individual variance of LEDs and photo-transistors are also compensated in this step. Next, the image g is gamma-corrected to emphasize the tip of the thumb ($\gamma = 2.0$). The center of mass of the resulting image h is the first estimate (x, y) of the thumb position. This first estimate (x, y) suffers from an error due to the light reflection from the proximal part of the thumb. This error is roughly proportional to the degree of ellipticity of the thumb projection in the image h. The degree of ellipticity is especially large when the thumb tries to reach the upper-left corner of the sensor area. In order to correct this error, we estimated the degree of ellipticity e by $\sqrt{\mu_{1,1}}$, where $\mu_{1,1}$ is the (1,1) central moment of the whole image, and adjusted (x, y) by $0.3 \times e$ to the left-up diagonal direction.

The vertical position z of the thumb is determined by the maximum value in g: $z = 1 - \max_{i,j} g(i,j)$. This value is low-pass filtered and then differentiated to give the vertical speed v_z of the thumb. The final step is the speed-dependent LPF that deals with jittering noise in the thumb position (x, y). It is a one-pole LPF whose pole is not constant but is dynamically determined by the variance (σ_x, σ_y) in the recent samples of (x, y) and the vertical speed v_z of the thumb:

$$(\bar{x}, \bar{y}) = \alpha_x(\bar{x}, \bar{y}) + (1 - \alpha_x)(x, y)$$

$$(\alpha_x, \alpha_y) = \min(1, \max(0, \alpha_0 - \beta(\sigma_x, \sigma_y) - \rho v_z))$$

where $(\bar{x}, \bar{y})$ is the output of the filter, and α_0,β, and ρ are positive constants that control the overall smoothing strength, the dependence on the horizontal thumb speed and the dependence on the vertical thumb speed, respectively. By a careful choice of these parameters ($\alpha_0 = 0.9$, $\beta = 1.0$, $\rho = 3.0$), pointing and selection operations became a lot more stable while the responsiveness of the device was not overly affected. Figure 4 shows the four states of the sensor and the transition conditions. The transition between "Off" and "Hovering" is triggered by the change of the vertical position estimation z ($z_{min} = 0.7$, $z_{max} = 0.8$, corresponding to about 10mm), while other transitions are triggered by the change of dedicated sensor outputs.

Visual feedback design

It is important to provide an effective visual feedback about the thumb position for the successful realization of the RemoteTouch concept. After a few trials, we chose a simple design not interfering with the current targets and tasks: a small circle with a translucent shadow as illustrated in Figure 5a. We used the disparity between the circle and the shadow to represent the height of the thumb above the touchpad.

User Test

We evaluated the efficiency of the RemoteTouch controller for pointing tasks and dragging tasks by comparing it with a "clickable" touchpad on the one hand and a "clickable" joystick on the other. Dragging tasks were included in the test as we thought that they would be crucial in RemoteTouch interactions, e.g., for moving a slider or for flicking a page. The "clickable" touchpad resulted from combining Cirque's GlideTouch touchpad with a button switch under the pad similar to the touchpad of an Apple MacBook. There were three states: off-state (over-the-pad), on-state (touched), and pressed state (touched and pressed).

For the "clickable" joystick, we used the right stick of the Xbox 360 controller, which can be pressed as well as tilted.

Pointing with the RemoteTouch controller is achieved by moving the thumb in the hovering state and landing it on the target displayed on the TV screen. Dragging, on the other hand, is done by touching a "start circle", moving the thumb with contact to the pad, and lifting the thumb at the on-screen target. In contrast, pointing with the clickable touchpad is done by moving in the touched state and clicking on the target, while dragging is done by pressing on the start circle, moving in the pressed state, and releasing at the target. Finally, pointing with the clickable joystick is realized by moving in the rate control mode and clicking on the target, while dragging is done by pressing on the start circle, moving in the pressed state, and releasing at the target.

The experiment was a 3(device) x 2(task) x 5(session) within-subjects factorial design. We asked participants to perform pointing and dragging tasks with the three devices for 5 sessions. Twelve university students (average age 25.7, 10 females) were recruited and the devices were presented to them in a counterbalanced order. Fifty random-circle-tracking trials were allotted to each condition, and one session (3x2 conditions) took about an hour. We measured throughputs and error rates by device and task for each session, and evaluated participants' task workload by the NASA-TLX questionnaire of workload after the experiments. The throughput for both tasks was computed by fitting the experimental data to Fitts' law and estimating the reciprocal of the slope of the line (time vs. index of difficulty) as done in [13].

The means and standard deviations of the measures are presented in Table 1. The effect of device was statistically significant for throughput in the pointing tasks (ANOVA, $F(2,118) = 66.053$, $p < 0.001$). In the dragging tasks, there was a significant effect of device for throughput (ANOVA, $F(1.3, 76.5) = 56.303$, $p<0.001$). In this case, we applied Greenhouse-Geisser correction since Mauchly's p-value was greater than 0.05. There was no significant difference between the throughput of the RemoteTouch controller and the touchpad in pointing tasks. However, the RemoteTouch controller outperformed others in dragging tasks.

		RemoteTouch	Touchpad	Joystick
Pointin	Throughput(bps)	3.1(1.1)	3.1(1.0)	1.6(0.3)
	Error rate(%)	6.2(4.8)	2.5(2.4)	1.1(1.9)
	NASA-TLX	293(84)	363(105)	255(104)
Draggin	Throughput(bps)	3.9(2.1)	2.5(0.9)	1.6(0.4)
	Error rate(%)	5.5(4.4)	5.5(4.9)	20(13)
	NASA-TLX	234(72)	292(85)	356(107)

Table 1. Throughputs, error rates, and workloads of the three devices for the pointing task and the dragging task (standard deviations in parentheses).

The throughput of RemoteTouch was better in dragging tasks than in pointing tasks ($t=-2.851$, $p=0.003$) contrary to the results of a previous research [13]. In addition, interestingly, the score of NASA-TLX of RemoteTouch were also significantly lower in dragging tasks than in pointing tasks ($t=3.495$, $p=0.002$). These results are persuading us to put more emphasis on dragging-based operations than clicking-based operations in the user interface design of RemoteTouch interaction.

SECOND PROTOTYPING

The experimental results of the first prototyping did not live up to our expectations, but gave us a chance to verify and correct our assumptions about RemoteTouch. First of all, we learned through observation that moving in the hovering state of RemoteTouch is not as easy as doing it on a real touchscreen. With a touchscreen phone, for example, we can maintain the thumb over the surface without much difficulty. This is not the case anymore when we look at the feedback on the TV screen. While there may be other important differences between these two cases, we could observe that lack of depth cue was a major problem in the second case. RemoteTouch seemed to share the same problem due to the low quality of the visual feedback of the thumb. The quality of visual feedback is affected by the following three factors: the latency of the sensor, noise in the thumb position estimation, and the visualization method. In the second prototyping, we focused on improving on these three factors. Also, in the design of the RemoteTouch interaction, we tried to minimize the time staying in the hovering state, since staying in a hovering state may be still stressful compared with the case of a real touchscreen.

Prototype Design

Hardware design

The two main improvements in the hardware were resistance to ambient light noise and an increased frame rate. Ambient light noise may be from low-frequency sources such as daylight and incandescent lamps, or from high-frequency sources such as fluorescent lamps. In order to cope with ambient light noise, we changed the ScreenPad firmware so that it uses differential sampling; it measures ambient light noise levels between successive pixel samplings with LEDs off, and use them to subtract noise levels from pixel values. Since pixel samplings and noise samplings were close in time (about 150μs) and linear interpolation was used to estimate noise levels at pixel sampling times, the differential sampling method could make the prototype almost insensitive to high-frequency ambient light noise as well as to low-frequency ambient light noise. Next, we increased the frame rate of ScreenPad from 25 Hz in the first prototyping to 50 Hz. The speed of the hardware in the first prototype was already fast enough as the total amount of time for acquiring a thumb image and transmitting it to the PC took less than 20ms. This remained the same when the frame rate was increased to 50Hz. Also, the signal processing on the PC leveled off at 5ms after testing various signal processing algorithms. In short, the total time delay due to the processing pipeline remained mostly unchanged when the frame rate was raised from 25Hz to 50Hz. Nevertheless, the increased update rate of the thumb's visual feedback made the hardware appear

more responsive. Another change is the replacement of the force sensor by a button switch (a tact switch). We used a force sensor in the first prototyping in order to adjust the click threshold. We realized that the high-fidelity haptic feedback of a click operation is more important and decided to use a mechanical switch.

Signal processing

The overall flow in Figure 3 was maintained except for the following two changes. First, the method to determine the center of the thumb was changed. The center of mass, which earlier was used to represent the center of the thumb, caused problems when the thumb was in a strongly inclined posture. We resized the 5x7 image, h, into 20x28 image, H, using a bi-cubic interpolation method before we calculated the center of mass, $(\bar{x}, \bar{y})$. Also, we did not use all pixels but only pixels with brightness above a threshold determined by the peak value, and weighted by their brightness. That is,

$$(\bar{x}, \bar{y}) = \sum_{w_{ij}>0} w_{ij}(x_{ij}, y_{ij}) \Big/ \sum_{w_{ij}>0} w_{ij}$$

$$w_{ij} = H(i,j) - \emptyset H_{max}$$

where (i,j) is the index to the image, H, (x_{ij}, y_{ij}) is the position of the pixel (i,j), H_{max} is the maximum value in $H(i,j)$, and $\emptyset$ is a parameter that controls the dynamic threshold and was set to 0.7 experimentally.

Secondly, the step to correct the thumb position using the ellipticity of the thumb was now omitted. This algorithm could not cope well with individual differences. Also, the thumb position correction was only necessary when users touched the pad with the bottom of the thumb. It seemed to be more practical to expect users to learn that using the tip of the thumb is better than using the bottom of the thumb.

In the hope to further improve noise tolerance, we tried Kalman filters in place of the speed-dependent LPF. We tried different dynamical models, including a constant velocity model and a constant acceleration model, and did our best to optimize the noise parameters of the models. Nonetheless, the speed-dependent LPF performed better in terms of both stability and responsiveness of the thumb movement. Therefore, we decided to keep the speed-dependent LPF.

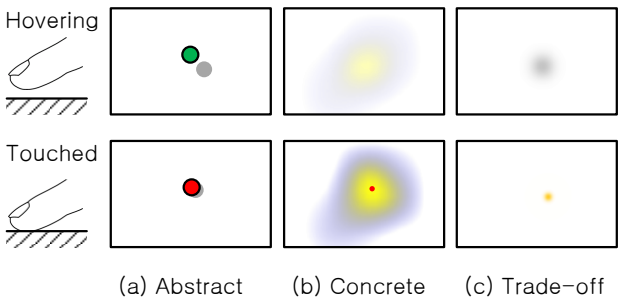

Figure 5. The three visual feedback designs that were considered in the second prototyping.

Visual feedback design

The quality of the visual feedback of the thumb does not only depend on the quality of the position information, but also on an effective visual design. These two are in fact dependent on each other. A very realistic visual feedback can result in a less stable control when the position information is very noisy. The level of realism and precision that a visual feedback design should pursue depends on the quality of the available information.

Figure 5 shows the three designs of visual feedback that we considered in the second prototyping. Figure 5a shows the visual feedback design that we used in the first prototyping. A solid disk represents the position of the thumb, and the distance to its shadow (the half transparent disk under the solid disk) presents the distance between the thumb and the touchpad. When the thumb touches the surface, the solid disk overlaps the shadow and changes color to indicate transition into the touched state. The problem with this design was that its positional feedback appears more precise than the actual information available from the hardware. As the thumb hovers higher, the positional signal becomes noisier and less accurate, but the visual feedback appears the same, i.e., crisp and definite. This is misleading and seems to have a negative influence on the motor control of the thumb. This problem is not only due to the limited quality of the sensor information, but also due to the ambiguity of the "perceived thumb position" [8]. The hot-spot of the thumb may be perceived as at the tip of the thumb for some, and at the center of the thumb for others.

Figure 5b shows a feedback design that was intended to cope with this problem. The image of the thumb acquired by the sensor is shown as is with pseudo-color, and the center of the thumb is not provided until the thumb touches the surface. The reflected light values are mapped to the hue. Pixels touching the surface are painted in solid yellow, while pixels above the surface are pointed in transparent gray. This design seemed to solve the problem with design (a), but was visually too distracting; the image of the thumb was too large and too dynamic. Figure 5c shows the final design that we settled with. This may be regarded as a tidy abstraction of design (b). A Gaussian profile is used to represent the thumb. Its size, color, and transparency are controlled to show the proximity of the thumb. When the thumb is in contact with the surface, it is represented by a small, solid, yellow Gaussian dot. When the thumb is hovering, it is represented by a large, transparent, gray Gaussian shadow. The size, transparency, and color of the image are also consistent with the accuracy of the position information. The final design may be regarded as a trade-off between the two extreme designs (a) and (b).

The TV Application

We implemented a TV application that would allow us to experience and experiment with a RemoteTouch-style interface. Figure 6a shows the main screen of the TV. "Edge-bars" frame the screen on each of its four edges. They remain inactive until a user clicks on one of the edges.

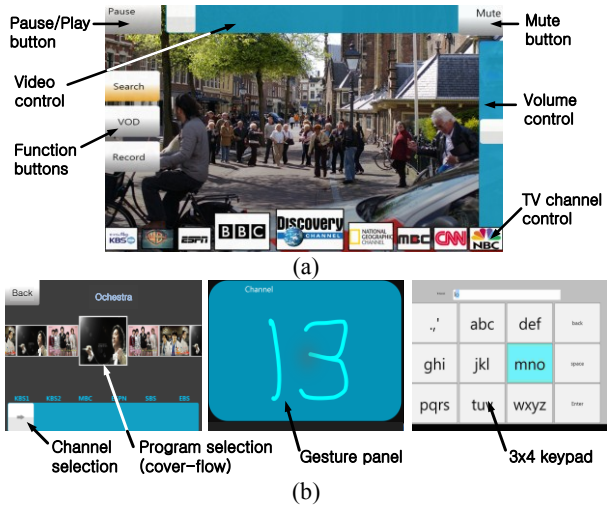

Figure 6. (a) The four "edge-bars" and (b) three sub-screens for VOD selection(left), gesture input (center), and text entry (right), respectively.

Only one bar at a time is visible and active. For illustrative purposes, we showed them all activated in this picture. The bottom edge-bar is for selecting a channel. A user can slide on the edge and press on an icon to select a channel. The top edge-bar is for controlling playback of the content; a user can slide on the edge to move forward or backward in the program. The right edge-bar is for controlling the volume and the left edge-bar is for a short-cut menu. Figure 6b shows three sub-screens. The first is the VOD selection interface invoked by the VOD button in the left edge-bar. A user switches a category using the bottom slider control and selects a program in a list in the center, similar to the cover-flow interface on the iPod. The second is the gesture panel invoked by clicking in the center of the main screen. A user can write a channel number to switch channels directly. The third is the text-input interface invoked by the Search button in the left edge-bar. These three interfaces are the three test tasks in the user test, to be explained in more details in the next section.

Focus Group Interview feedback

We conducted a focus group interview (FGI) with 4 undergraduate students (average age 24.8, 1 female). Two of them had about three years' experience with an Internet protocol TV (IPTV) interface, and others have none. We gave them a 5-minute instruction on the use of the current IPTV interface (B-tv, SK Broadband) and the use of the RemoteTouch interface for the TV application. The IPTV interface that we used for a reference was based on a traditional remote with directional keys. We gave them an additional 15 minutes for trying the interfaces themselves. The FGI continued for 50 minutes afterwards.

The participants seemed to feel comfortable with the current IPTV interface, but complained about difficulty in selecting an item in the deep hierarchical menu and inconvenience in using a search function. They all showed interests in RemoteTouch interaction style but seemed to need time to become used to it. A lot of useful comments were collected and representative ones are summarized in Table 2.

Questions	Comments
Pros and cons of RemoteTouch?	It is refreshing; Its use is intuitive; Cursor movement is too sensitive because the sensor area is small; Cursor movement is a lot more continuous than expected; Pressing a button would be easier than writing on the pad.
Able to use after a short introduction?	A short introduction was enough; I felt fatigue at the finger since the device was very sensitive.
Compared with pointing devices like Wii remote?	They are more intuitive, but are more tiresome since I have to move the arm all the time; It would be better to move on a small area like ScreenPad.
Additional comments?	It makes me feel like really "remote-controlling"; The playback control of a movie would be done better with gestures than manipulating the slider control; RemoteTouch may be useful for a vehicle UI where visual attention to the device is limited.

Table 2. User comments from an FGI with the TV application.

User Test

The user test has two aims. One is to obtain user feedback after giving the participants sufficient time to get used to the RemoteTouch interaction style. The other is to assess the advantage of hover-tracking in touch-screen-like interactions. For this purpose, we let participants try three tasks corresponding to the three sub-screens in Figure 6b in two conditions using the RemoteTouch device: Hover-Tracking (HT) and No-Hover-Tracking (NHT). In the NHT condition, it was impractical to use a flick gesture or write by touch, because the initial position of the thumb is not visible, and therefore, dragging and writing in the pressed state had to be used instead. A more detailed description of the differences of the three tasks for the two conditions is given in Table 3.

	HT	NHT
Task 1: Numeric gesture	Writing in the touched state; visual feedback in the hovering state	Writing in the pressed state; no visual feedback in the hovering state
Task 2: Program selection	Dragging in the touched state; visual feedback in the hovering state; selection by a click	Dragging in the pressed state; no visual feedback in the hovering state; selection by a double-click
Task 3: Text entry	Key input by a click; visual feedback in the hovering state	Key input by a click; no visual feedback in the hovering state

Table 3. The differences of the three tasks in the two conditions, HT and NHT.

Participants and procedure

We started with 12 university students, but one of them failed to complete the whole tests for a personal reason. All of the 11 successful participants (average age was 25.4, 4 females) had some experience with a touchpad.

One session consisted of three blocks, one for each of the three tasks. A short break for a few minutes was allowed between the tasks. Since one session took less than 20 minutes and did not cause too much fatigue, two or three sessions were conducted per day. A practice session for about 10 minutes was allowed every day before test

sessions began. The experiment was a within-subject design. Six of them started with the HT condition first, and others with the NHT condition. In order to reduce the interference between two conditions, we let them have at least a two days' break between conditions. In the end, each participant completed 5 sessions for each condition.

In the following, we describe each experiment task with reference to Figure 7.

(a) Numeric gesture (Task 1): A two-digit number is presented in the left text box, and a participant writes it on the screen. The gesture recognizer available from Windows .NET Framework was used for gesture recognition. The recognition result is displayed in the right text box. This task is repeated for 20 two-digit random numbers in a block.

(b) Program selection (Task 2): This task is to select a program from a channel. The slider bar at the bottom shows 8 channels, and the cover-flow-like list in the center contains 20 programs from the selected channel. A target to select is presented by a channel-program pair (represented as letters in the experiment) on the top of the screen, and a participant selects the channel and the program in succession. This task is repeated for 10 random targets in a block.

(c) Text-entry (Task 3): This task is to enter a sentence using a 3x4 screen keypad (multi-tap, alphabetic layout). A sentence from the MacKenzie and Soukereff's English phrase dictionary [14] is presented in the first text box, and the text-entry result is displayed in the second text box. This task is repeated for 5 randomly chosen sentences in a block.

Results

The experimental results for the three tasks are summarized in the following with reference to Figure 8.

(a) Numeric gesture: Figure 8a shows the average time for writing a two-digit number in the two conditions. The stroke time in HT was significantly shorter than in NHT ($t = 4.3785$, $p < 0.0001$). Figure 8b shows the recognition error rates in both conditions. The error rate in HT was slightly lower than that of NHT but the difference was not significant.

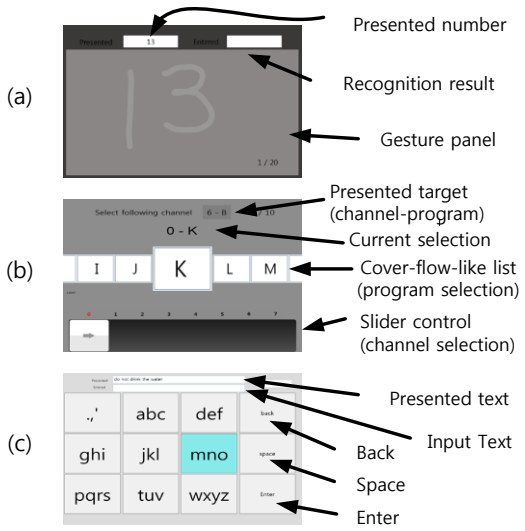

Figure 7. The three tasks: (a) Numeric gesture, (b) Program selection, and (c) Text-entry.

The gesture recognizer appeared to be the source of most errors. The recognizer frequently misread a numeric gesture despite its high legibility. In particular, '7' was often mistaken as '17' since the recognizer expected it to be written in a single stroke, while in Korea it is usually written in two strokes. Furthermore, we expected the recognizer to identify gesture inputs better in the HT case, since it also provided a superior quality of writing. Yet, contrary to our expectations, the recognizer performance was similar in both cases.

(b) Program selection: Figure 8c shows the average completion times for a program selection. Contrary to our expectation, participants performed better in the NHT condition ($t = 1.715$, $p < 0.05$). The curves in the figure show a learning effect (ANOVA, $F(4, 40)=9.562$, $p < 0.0005$), and the HT curve, at the end, appears to outrun the NHT curve, but the difference in the final block is not significant ($p = 0.07$). This stood in contrast to the post-test survey result and the NASA-TLX test result that are presented later. One of the possible reasons for this result was the difference in the way participants use the cover-flow-like list. While they reported that they liked the 'flicking' operation in the HT condition, flicking induced more overshoots. The average number of overshoots in a block was 3.1 for HT and 2.6 for NHT, and the difference was significant ($t = 1.741$, $p < 0.05$). The average error rates in Figure 8d are also reflecting this difference. Participants complained of accidental clicks that they made when they actually tried to flick the list. An interesting observation in Figure 8c is that there was a significant interaction between conditions and blocks (ANOVA, $F(4, 40)=4.722$, $p = 0.003$) implying a stronger learning effect in HT than in NHT as evident in the graph.

(c) Text-entry: There was no difference in text-entry speed

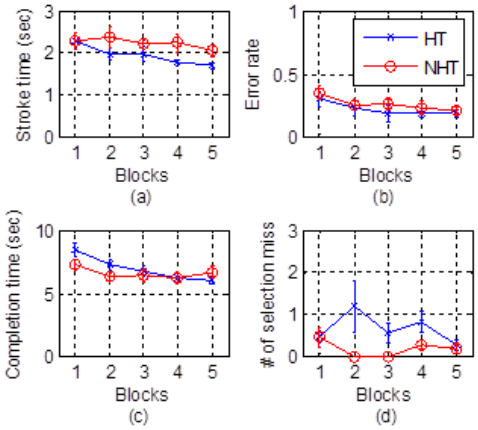

Figure 8. Test results: (a) the stroke time and (b) error rates for task 1, and (c) the completion time and (d) the number of errors for task 2.

between the two conditions ($t = 1.1710$, $p > 0.05$). This result was in fact anticipated from the participants' behavior in the test. Unlike the case of a physical keypad, they rarely moved the thumb off the touchpad. They repeated 'slide and click' instead of 'fly and land'. The interfaces in the two conditions are in fact identical when the thumb is always in contact with the pad, and therefore we could not expect any difference in the experimental result. We expected that their thumb would start to leave the surface as their typing becomes faster, and, if this happens, hover-tracking would be able to make a difference. However, they did not seem to have sufficient time to reach a typing speed of that level. No learning effect was observed in the result and the best speeds that they could achieve were about 8.5 wpm in both conditions. This is comparable with the text entry speed of novice users of a mobile phone keypad, which is about 8 wpm [12]. All the participants were indeed novice users since they are all Koreans and do not usually use an alphabetic keypad for text entry.

Post-test feedback

We conducted a brief questionnaire survey after the test, and the main results are summarized in Table 4. For the first five questions, the average values of a 5-point Likert scale feedback were shown. For the remaining questions the numbers of votes were shown. Except for the fourth question, their rating and votes were in favor of the HT condition. The difference in stroke time in task 1 between HT and NHT was only about 15%, but all participants in the experiment answered that writing a gesture in the HT condition was more natural, which may be more important than the small difference in the stroke time. Participants also favored the HT condition for manipulating the cover-flow-like list but favored the NHT condition for manipulating the slider in task 2. They seemed to favor the NHT condition for operations requiring precision. In task 2, the pointing operation to catch the tracker of the slider control required precision, and this was easier in the NHT than in the HT condition.

	Questions	HT	NHT	No idea
Task 1	1. Was the digit recognition function satisfactory?	3.36	2.64	
	2. Could you easily get used to the device?	3.55	3.27	
	3. Did you find the writing action natural?	3.27	2.27	
Task 2	4. Could you easily manipulate the slider control?	3.18	3.45	
	5. Could you easily manipulate the Cover-Flow control?	3.73	3.45	
Over-all	6. Which is better for task 1?	9	2	0
	7. Which is better for task 2?	5	4	2
	8. Which is better for task 3?	5	3	3

Table 4.The result of the post-test questionnaire survey.

For task 3, only 6 of the participants reported that they could perceive a difference between the two conditions. They said that they tried harder to maintain the thumb in contact with the pad for concern that they might lose visual feedback of the thumb on the screen in the NHT condition. Participants reported that they felt less fatigue and faster visual feedback in the HT condition. In order to assess the overall task load in the two conditions, we conducted NASA-TLX tests after each condition, and summarized the result in Table 5. All numbers are in favor of the HT condition except for the item 'Frustration'. A frequent complaint was about the shift of the thumb image when they moved fast to catch a target on the screen. Errors caused by the less-than-perfect visual feedback might have been the main cause for the frustration.

NASA-TLX	HT		NHT	
	Rating	Weight	Rating	Weight
Mental demand	35(7)	1.6(0.4)	41(6)	1.6(0.4)
Physical demand	42(9)	2.6(0.6)	57(8)	3.1(0.6)
Temporal demand	39(7)	2.5(0.5)	40(7)	2.2(0.4)
Effort	38(9)	2.9(0.5)	38(8)	3.0(0.6)
Performance	47(9)	3.0(0.5)	53(7)	3.0(0.4)
Frustration	42(9)	2.6(0.6)	39(8)	2.2(0.6)

Table 5. The average scores of the NASA-TLX test (standard errors in the parentheses). A lower score means lower demand or better performance.

DISCUSSION

An important yet unaddressed issue in this paper is the ergonomic aspect of the RemoteTouch controller. Indeed, we considered the movement range of the thumb when we determined the size of ScreenPad. Moreover, we rotated the orientation of ScreenPad by 20 degrees to the CCW direction as shown in Figure 1 in order to align the horizontal axis of ScreenPad better to the natural movement direction of the thumb. We also tried to make the shape of the controller handle fit the grip better. There is still a lot of room for improvement in the ergonomic aspects of the RemoteTouch controller. A related issue is considerations for handedness. Throughout our study on the RemoteTouch interaction, we assumed right-handedness. For instance, the rotation angle of ScreenPad in Figure 1 is for right-handed users. Also, the lighting model that we assumed in the design of the visual feedback shown in Figure 5a is also biased for right-handed users. These designs may be easily mirrored for left-handed users, but a single design for both types of users will be a challenge.

Choosing between absolute (AC) and relative (RC) position control of the cursor is another important discussion point. This question deliberately has not been addressed in this paper, since AC is an essential part of the RemoteTouch concept. A pilot study to compare the two cases for a clickable touchpad was done in the early investigation phase as RC was a common choice for a touchpad. AC has advantages when it comes to utilizing one-to-one mapping to access targets on the edges or the vertices of the screen directly. A disadvantage with AC was the instability of the cursor due to a high control-display gain. RC uses a smaller control-display gain minimizing the instability problem. A smaller gain, however, resulted in frequent clutching. Another difference was the role of the cursor. The cursor in AC was perceived as an image of the thumb, but in RC was perceived as an object to move. As such, RC requires the cursor as a mediator, which compromises the directness of

control. The final choice between AC and RC remains an open question and should be determined by the requirement of the usage scenario.

One of the considerations for the RemoteTouch controller to be a practical solution is the power consumption of ScreenPad. While ScreenPad consists of a matrix of LEDs, it turns on only one LED at a time, meaning that the power consumption of the whole matrix is the same as that of a single LED. In fact, we designed the current through the LEDs to be below 20 mA in order to drive them directly by a microcontroller. The capacity of a common alkaline AA battery is about 2500mAh, and therefore a single battery cell (in fact, two or more cells to provide a required voltage) can power the LED array for at least 125 hours. The power consumption of the LED array is comparable with the LED illumination of an optical mouse. Considering the usage pattern of a TV remote, we believe 125 hours of continuous operation will not be too short to make it a practical solution. Of course, this implies that LEDs can be turned off when the remote is not in use. The ScreenPad has a button switch (tact switch) for clicking and a touch sensor for detecting a touch by the thumb. The first option will be using the mechanical button switch for waking up the microcontroller. If clicking is too laborious for users, we may consider leaving the touch sensor active in a low-power state.

CONCLUSION

This paper introduced the RemoteTouch concept proving its validity by iterations of prototyping and a user study. Through these iterations we could correct some of our false expectations, and also verify its potential as a viable option for a TV remote control. While the main focus of the second iteration was to enable better visual feedback of the thumb, the user study results indicate that it is still the most important factor to work on further. Immediate future work is to improve the accuracy of the sensor, especially the accuracy of the hover distance, and pursue a realistic, 3-dimensional visual feedback of the thumb.

The contributions of this study are twofold. Foremost, it is the first high-fidelity prototyping study that examines the problems and possibilities of touch-screen-like interactions in a TV remote control environment. Secondly, Screen Pad, a hover-enabled touchpad, is a practical design example for implementing such interactions. The advantages of its operating principle, such as low power consumption by successive lighting of the LEDs and high noise immunity by binding photo-transistors into a single sensor, may also be exploited in the future designs of other hover-enabled optical touchpads.

REFERENCES

1. Abileah, A. and Green, P. Optical sensors embedded within AMLCD panel: design and applications. ACM SIGGRAPH 2007 Emerging Technologies.
2. Cirque – Innovative Touch Solutions. http://www.cirque.com.
3. Douglas, S. A., Kirkpatrick, A. E., and MacKenzie, I. S. Testing pointing device performance and user assessment with the ISO 9241, Part 9 standard. *In Proc. CHI 1999*, 215-222.
4. Enns, N. R. and MacKenzie, I. S. Touchpad-based remote control devices. *In Proc. CHI 1998*, 229-230.
5. Ginny J., et al. The challenges of designing a user interface for consumer interactive television, *In Proc. Int. Conf. on Consumer Electronics*, (1994), 114-115.
6. Han, J. Y. Low-cost multi-touch sensing through frustrated total internal reflection. *In Proc. UIST 2005*, 115-118.
7. Hodges, S., Izadi, S., Butler, A., Rrustemi, A., and Buxton, B. ThinSight: versatile multi-touch sensing for thin form-factor displays. *In Proc. UIST 2007*, 259-268.
8. Holz, C. and Baudisch, P. The generalized perceived input point model and how to double touch accuracy by extracting fingerprints. *In Proc. CHI 2010*, 581-590.
9. Ishiyama, K., and Yano, S. A study of characteristics of pointing devices for television operation, *In Proc. IEEE SMC 2000*, 1307-1312.
10. Lekakos, G., Chorianopoulos, K., Spinellis, D. Information systems in the living room: a case study of personalized interactive TV design. *In Proc. the 9th Euro. Conf. on Information Systems*, (2001), 319-329.
11. MacKenzie, I. S. and Jusoh, S. An evaluation of two input devices for remote pointing. *In Proc. IFIP Int. Conf. on Eng. for HCI 2001*, 235-250.
12. MacKenzie, I. S., Kober, H., Smith, D., Jones, T., and Skepner, E. LetterWise: Prefix-based disambiguation for mobile text input. *In Proc. UIST 2001*, 111-120.
13. MacKenzie, I. S., Sellen, A., and Buxton, W. A. A comparison of input devices in element pointing and dragging tasks. *In Proc. CHI 1991*, 161-166.
14. MacKenzie, I. S. and Soukoreff, R.W. Phrase sets for evaluating text entry techniques. *CHI 2003 Extended Abstracts*, 754-755.
15. Malik, S., Ranjan, A., and Balakrishnan, R. Interacting with large displays from a distance with vision-tracked multi-finger gestural input. *In Proc. UIST 2005*, 43-52.
16. Matsushita, N. and Rekimoto, J. HoloWall: designing a finger, hand, body, and object sensitive wall. *In Proc. UIST 1997*, 209-210.
17. Myers, B. A., Bhatnagar, R., Nichols, J., Peck, C. H., Kong, D., Miller, R., and Long, A. C. Interacting at a distance: measuring the performance of laser pointers and other devices. *In Proc. CHI 2002*, 33-40.
18. Olsen, D. R. and Nielsen, T. Laser pointer interaction. In Proc. CHI 2001, 17-22.
19. Panasonic's EZ Touch Remote prototype, Exhibited at CEATEC Japan 2008, Makuhari Messe, Japan.
20. TactaPad – See, Touch, Feel. http://www.tactiva.com/tactapad.html.