iGrasp: Grasp-based Adaptive Keyboard for Mobile Devices

Lung-Pan Cheng, Hsiang-Sheng Liang, Che-Yang Wu, Mike Y. Chen
Department of Computer Science
National Taiwan University
Taipei, Taiwan 10617
{r98922168, r01922036, b98902113, mikechen}@csie.ntu.edu.tw

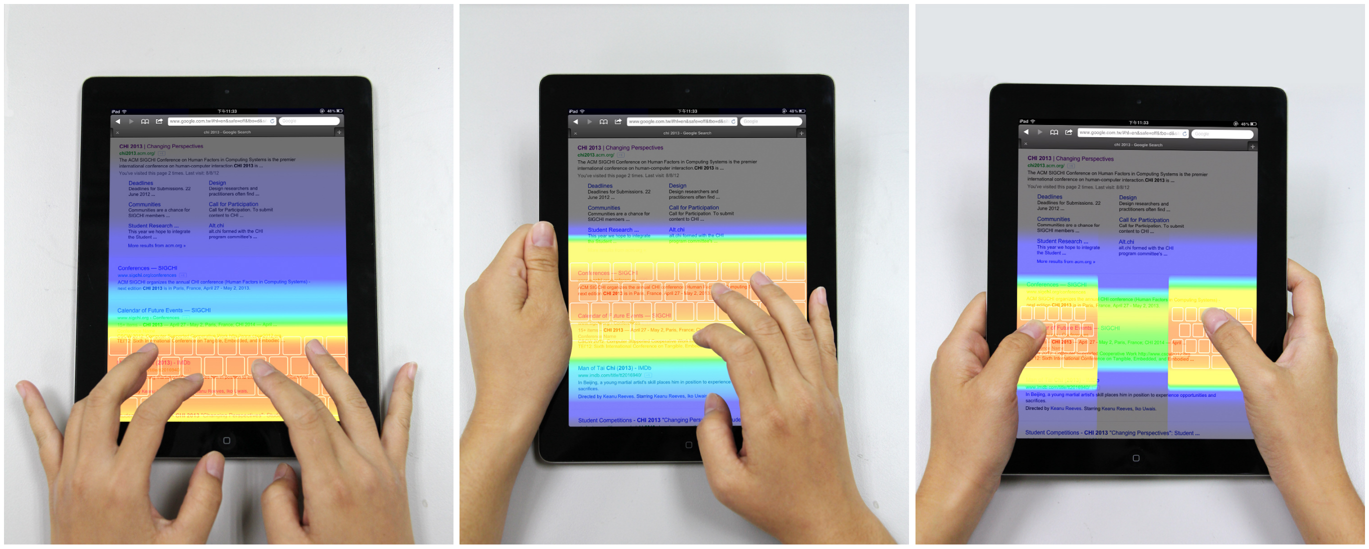


Figure 1. Heatmap visualization of preferred keyboard layouts and positions for three grasp conditions in our 64-user study: (left) not grasping the devices, (center) grasping the devices with one hand, (right) grasping the devices with both hands. The most preferred keyboard modes are merged+docked, merged+undocked, and split+undocked, respectively.

ABSTRACT
Multitouch tablets, such as iPad and Android tablets, support virtual keyboards for text entry. Our 64-user study shows that 98% of the users preferred different keyboard layouts and positions depending on how they were holding these devices. However, current tablets either do not allow keyboard adjustment or require users to manually adjust the keyboards. We present iGrasp, which automatically adapts the layout and position of virtual keyboards based on how and where users are grasping the devices without requiring explicit user input. Our prototype uses 46 capacitive sensors positioned along the sides of an iPad to sense users' grasps, and supports two types of grasp-based automatic adaptation: *layout switching* and *continuous positioning*. Our two 18-user studies show that participants were able to begin typing 42% earlier using iGrasp's adaptive keyboard compared to the manually adjustable keyboard. Participants also rated iGrasp much easier to use than the manually adjustable keyboard (4.2 vs 2.9 on five-point Likert scale.)

Author Keywords
Adaptive user interfaces; grasp detection; virtual keyboard; mobile devices.

ACM Classification Keywords
H.5.2. User Interfaces: Interaction styles; Input devices and strategies

INTRODUCTION
Multitouch mobile devices, such as iPhone, iPad, and Android tablets, support virtual keyboards for text entry. While text entry performance for virtual keyboards has been extensively studied and improved, such as through gestures [17] and adaptive techniques including predicted target zones [10, 13] and motion compensation [7], most work has focused on fixed keyboards. This paper explores automatic adaptation versus manual adjustment of the keyboard layouts (horizontally *split* and *merged*) and vertical keyboard positions (*docked* at the very bottom and *undocked*).

To better understand users' preferences for the layout and position of virtual keyboards, we conducted our first user study: a 64-person observational study. We asked participants to perform typing tasks under the three grasp conditions of using a tablet: 1) not grasping the device, 2) grasping the device with one hand, and 3) grasping the device with both hands. For each grasp condition, participants used all four possible keyboard modes and vertical positions currently available on iPad: 1) merged+undocked, 2) merged+docked, 3) split+undocked, and 4) split+docked[1].

We asked the participants to report their preferred keyboard layout and position for each of the three grasp conditions. Figure 1 shows a heatmap visualization of the results. Most users preferred merged+docked mode when not grasping the devices, merged+undocked when grasping the devices with one hand, and split+undocked when grasping the devices with both hands (preferred by 74%, 88%, and 70% of the users, respectively). In addition, 98% of the participants preferred two or more distinct keyboard modes, for an average of 2.49 distinct keyboard modes across the three grasp conditions. Furthermore, among the 17 iPad owners in the study, 76% were not aware that its keyboard were adjustable, indicating a potential discovery problem for manually adjustable keyboards.

We present iGrasp, which automatically adapts virtual keyboard's layout and position based on users' grasps without requiring explicit user input. We built an iGrasp prototype using 46 capacitive sensors placed on the sides of an iPad, which are connected to an Arduino board and then wired into iPad's serial port. It is capable of sensing the users' grasp position in 1 cm resolution, and can distinguish between the three grasp conditions.

iGrasp supports the following two adaptation modes to improve performance and ease-of-use of virtual keyboards.

- *iGraspSwitch*: senses the user's grasp condition by grouping the 46 sensors into four sensor groups and adapts the keyboard to the same layout and position that was last used for the currently sensed grasp condition.

- *iGraspPosition*: additionally senses the current grasp position and continuously adapts the keyboard to the grasp location by utilizing all 46 sensors.

We conducted the other two user studies to evaluate iGrasp for two adaptation modes. For *iGraspSwitch*, our results show that participants strongly preferred iGrasp over manually adjustable keyboards (4.2 vs 2.9 on five-point Likert scale). Also, the time to type the first character improved by 42%, from 2.57 seconds to 1.49 seconds. For *iGraspPosition*, our results show no statistically significant improvement in performance nor preference over *iGraspSwitch*. Based on our findings, its possible to implement iGrasp using only four capacitive sensors, with each sensor having a sensing area that covers half of each side of the device and having similar width as our prototype.

[1]We used private iOS APIs to enable split+docked mode, because the default iOS behavior would merge the keyboard when it is docked.

Our contributions include the following: 1) we show that most users have different preferred virtual keyboard layouts and positions for different grasp conditions, 2) we propose iGrasp, a novel approach that uses implicit grasps of a device to automatically adapt the on-screen keyboard's layout and position to match users' preferences, 3) we show that the grasp-based adaptive keyboard significantly helps users to type earlier and is strongly preferred by users over current manual approaches.

The rest of the paper is organized as follows: first, we discuss related work, then describe the first observational user study and finding. We then present iGrasp design and implementation, and discuss evaluation methodology and results for the second and third user studies. Last, we conclude with discussion, contributions, and future work.

RELATED WORK

Modern mobile devices often have different kinds of sensors such as accelerometers, gyroscopes and cameras, to support contextual sensing [27]. The users' context then can be used for adapting interfaces. For example, WalkType [7] uses touch and accelerometer data while users are sitting and walking to built displacement models that can be used to compensate for imprecise input during walking. Researchers also use accelerometers and cameras to rotate screen orientation [3, 14, 28]. iGrasp sees users' grasps as their context to adapt the keyboard interface. In the following sections, we will discuss related work about 1) text entry and keyboard adaptation, 2) sensing grasp and 3) grasp-based user interface.

Text Entry and Keyboard Adaptation

Visible and invisible key-target adaptation have been widely studied to improve text entry performance of virtual keyboards. Examples of visible adaption include Himberg et al. [13], who propose key shape, size and position adaptation based on users' previous touches. Text Text Revolution [23] is a typing game that helps users improving typing performance. The game provides targeting practice, highlights areas for improvement and generates training data for key-target resizing. BigKey [5] expands the key size of the next entry character predicted from tables of single letter and diagram frequency counts [19]. Fitrianie et al. [6] use n-grams language model to predict the next most likely character and resized a circular text input interface. iGrasp is also a type of visible adaptation, but instead of resizing keys on a keyboard, iGrasp changes the layout and position of the entire keyboard.

Invisible input adaptation changes the touch size, shape, and location of keys, but does not modify their on-screen appearances to minimize the possibility of visual distraction. Gunawardana et al. [10] propose an anchored method to avoid overly aggressive key-target resizing. Each key's target area would change invisibly without crossing other key's anchor. Because iGrasp changes the layout and position of the entire keyboard, it can augment visible and invisible key-target adaption techniques.

Sensing Grasp

Many sensing technologies have been used to sense grasp. GripSense [8] leverages the built-in inertial sensors, vibration motors, and touchscreens of smartphones for grip and pressure detection. After monitoring device rotation, touch size, and thumb swiping, users' hand postures could be inferred.

Capacitive sensing is commonly used for researching touch sensing input devices [15] and users' grasp [16,30,35]. Many new technologies are invented based on capacitive sensing. Touché [25] proposes *Swept Frequency Capacitive Sensing* (SFCS) which sweeps through a range of frequencies to obtain a multitude of responses to know how a user is touching the object. Midas [26] is a software and hardware toolkit for automatically synthesizing capacitive touch sensor that enables designers to build touch sensitive prototype in any shape. Time domain reflectometry [21, 34], which originally was used to diagnose cable faults, can be used to locate a users touch on form factors such as guitar strings and conductive ink. Hand-Sense [35] uses four capacitive sensors on two long edges to recognize left-hand and right-hand usage. Our iGrasp prototype uses 46 capacitive sensors.

Pressure sensors [11,22,31] and impedance sensors [20] have also been used to sense grasps. Tango [22] uses 256 pressure sensors to make a hand-size ball as a haptic interface for 3D objects. The sensors use two layers of electrically conductive strips separated by a compressible foam rubber. Pressure compresses the two layers, thus increasing the capacitance. Arrays of piezo-resistive elements have been used identify the grip patterns on smart guns in order to identify the user for safety purposes [31].

Light and infrared sensors have also been used to sense grasps. TouchString [9] concatenates units of LED phototransistor pairs to form a multitouch sensor rail that can surround surfaces of objects such as bottles and mobile phones. SideSight [2] uses infrared proximity sensors embedded along the side of small devices to detect the presence and position of fingers. FlyEye [33] also uses infrared light and a camera to detect touch and proximity by measuring the changes in light reception through optical fibers that are embedded in the surface. Rock-paper-fibers [24] uses a bundle of optical fibers observed by a webcam and recognized how the bundle is shaped and touched by matching the resulting graph with its widget database.

Grasp-based User Interfaces

Graspables [30] uses several capacitive touch sensors as discrete, binary input to sense touches. They built two prototypes that can switch between the multiple functions of a device based on grasp: Bar of Soap switches between different applications, including camera, gamepad, phone, and remote control. Ball of Soap selects different pitches such as fastball and curveball. Kim et al. [16] report the accuracy of different machine learning methods on classifying eight grip patterns, and use the detected grip launch mobile device applications: call, SMS, cam-era, video, and game. There is further research discussing grasp interaction on mice and pens such as MT mouse [1] and MTPen [29]. Based on users' grasps, they could decide whether to switch to the other modes for hovering and drawing. GRASP [34] proposes a model of human grasping that describes five meaningful factors - goal, relationship, anatomy, setting and properties, and offers a basis and framework for further research and discussion. BiTouch and BiPad [32] studies users' holds of tablets and designed bi-manual interaction techniques such as chords and gestures on the bezel of devices. iRotateGrasp [4] proposes using users' grasps to adapt the screen orientation to users' viewing orientations on mobile phones. They collected users' grasps in different postures and orientations by the capacitive sensors on the edges and the back of their phone-sized prototype, and then they used support vector machine (SVM) to classify grasps into viewing orientations. iGrasp uses capacitive sensors as well. However, our paper focuses on investigating how users' typing experience can be improved by our proposed adaptation — changing the virtual keyboard layout and position based on how and where users are grasping their tablets.

OBSERVATIONAL STUDY

We designed a study to understand users' preferences for keyboard layouts and positions. We targeted the portrait orientation because it is a common usage and users' grasp positions would have more variation. We developed a custom text entry application for iPad that recorded the preferred keyboard layout (split or merged) and the Y-axis position of the keyboard. We recruited 64 participants (age 16 to 57, 39% female) and 27% owned iPads (average ownership 13.6 months).

Participants first learned how to change the keyboard layout and position, and tried all four keyboard modes in the portrait orientation: 1) merged+undocked, 2) merged+docked, 3) split+undocked, and 4) split+docked. Using the same manual adjustment methods as the built-in iPad keyboard, participants learned to change the keyboard's position by dragging the bottom-right button up and down. In addition, participants learned to split and merge the keyboard by performing pinch-in and pinch-out gestures on the keyboard. 76% of iPad owners reported that they did not know how to adjust the keyboard, indicating a potential discovery problem for manual keyboard adjustment.

Next, participants explored keyboard layouts and positions for each of these three conditions: 1) placing the device on a surface and not grasping the device, 2) grasping the device with one hand and typing with the other hand, and 3) grasping the device with both hands. We did not constrain how users grasped the devices. For example, in the single-handed grasp condition, most users grasped an edge of the tablet, but a few users supported the tablet with their forearms instead. For each condition, we asked the participants to try all four keyboard modes, and to type at least three phrases that were randomly selected from the Makenzie and Soukoreff phrase set [18] . After participants felt they had found a preferred keyboard layout and position for each condition, the layout and position were recorded. The average session duration was 10.4 minutes.

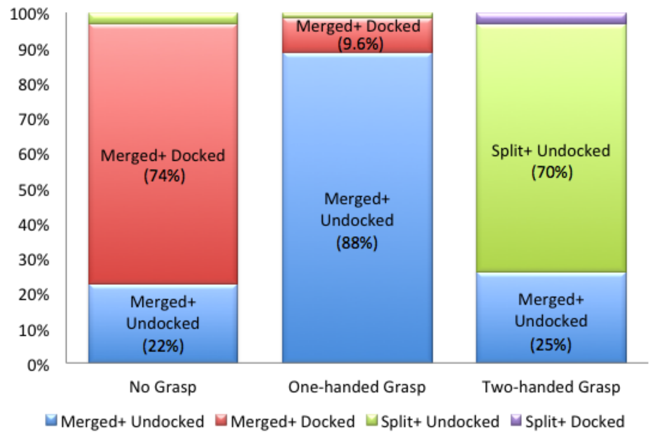


Figure 2. Distribution of the most preferred keyboard mode by grasp condition.

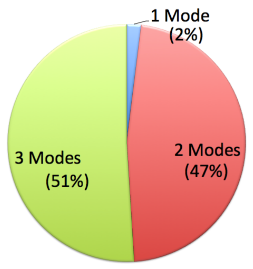


Figure 3. Distribution of the number of distinct keyboard modes preferred by each user for three different grasp conditions.

Keyboard Layout and Position Preferences

Figure 1 shows the heatmap visualization of keyboard layout and position preferences for all participants. Figure 2 shows the distribution of the most preferred keyboard mode for the 64 participants for each of the three usage conditions, showing that the most preferred keyboard mode is different for each condition. When typing without holding the devices, 74% of the participants preferred the merged+docked keyboard. This is the default keyboard mode supported on all the tablets that we surveyed, including iPad, Android tablets, RIM Playbook, and Palm TouchPad. When grasping the device by one hand and typing using the other hand, 88% of the participants preferred merged+undocked keyboard. When holding the devices using both hands, 70% of the participants preferred the split+undocked keyboard while 25% of the participants preferred merged+undocked keyboard. Participants commented that the split keyboard design had the following disadvantages: 1) keys were too small, 2) some keys such as 'B' were placed on the side they were not used to, and 3) they needed more time to visually seek characters across the middle of the keyboard.

Participants chose an average of 2.49 distinct keyboard modes across the three grasp conditions, and 98% of the participants chose at least two distinct keyboard modes. The detailed distribution is shown in Figure 3.

	No Grasp	One-Handed Grasp	Two-Handed Grasp
Average (pixel)	63	245	223
Median (pixel)	0	255	215
Standard Deviation	105	92	73

Table 1. Average and median keyboard position offset for the three usage conditions.

Table 1 shows the mean and the median of keyboard Y-axis position offset of all participants for each of the grasp conditions. The Y-axis offset value is 0 for the docked position. The high standard deviations in all three conditions show that users preferred very different keyboard positions. The mean of Y-axis position when users grasped the devices by one hand was 245 pixels (*SD*=92), and 223 pixels (*SD*=73) when users grasped the devices with both hands. These values may be used by iGrasp as the default keyboard positions.

DESIGN

The results from the observational study show that iGrasp needs to adapt both the layout and position of the keyboard. We designed two types of grasp-based adaptation: *iGrasp-Switch* and *iGraspPosition*.

iGraspSwitch

Adaptive layout switching augments manual keyboard adjustments by helping users automatically switch between different keyboard modes. It senses the current grasp condition (none, one-handed, and two-handed) and then automatically shows the keyboard using the same layout and position that was last used for that grasp condition. This requires the system to have sufficient sensor resolution to distinguish between different types of grasp. Also, the system needs to save the keyboard layout and position for each grasp condition. Although this approach may not always position the keyboard to exactly where the users' hands are, it provides users with the option to manually adjust the keyboard positions and layouts.

iGraspPosition

Instead of showing the keyboard at the last-used position, *iGraspPosition* senses the current grasp position and shows the virtual keyboard at that location. It shows the same keyboard layout that was last used for the current grasp condition. To avoid constant movement of the keyboard while typing, the system stops re-positioning the keyboard once the user has started typing. There are two key challenges to this adaptation. First, the system must be able to locate the grasp position with sufficient resolution in order to accurately position the keyboard. Second, the system must have an accurate model that maps the user's grasp position to an optimal keyboard position. Because users' hand sizes may vary significantly, personalized calibration can be used to improve the system's performance. The trade-off of this approach is that it replaces manual position adjustment so that if the keyboard is poorly positioned due to an inaccurate model, it will always be positioned poorly for that user.

IMPLEMENTATION

Our goal is to build a tablet-sized device that is capable of sensing the grasp position and distinguishing between different grasp conditions: no grasp, one-handed grasp, and two-handed grasp. The form factor and weight should be similar to typical tablets to minimize changes in users behavior. After exploring several sensing approaches, including light sensors and clip-on sensors [36], we decided to use capacitive sensors which are not sensitive to lighting conditions and can be placed more densely than clip-on sensors.

iGrasp Prototype

Our iGrasp prototype consists of the following components: an Arduino Pro Mini 328 circuit board, four Freescale capacitive touch sensor controllers MPR121, an iPad case, and an iPad 2. As shown in Figure 4, the circuit board and the controllers are placed in the center on the back side of the iPad. We placed 46 copper foils along both of the long sides of the iPad, with 0.2 cm gaps between the foils. Each copper foil is 4.0 cm $\times$ 0.8 cm in size and is connected to the capacitive sensor controllers, and the Arduino board samples each sensor as binary readings at 60Hz, then transmits the data to iPad via the serial port. All subsequent processing is done on the iPad, which runs iOS 5.1 and is jailbroken to enable the serial port. The dimension of the prototype is 18.9 cm (W) $\times$ 24.2 cm (H) $\times$ 1.3 cm (D). Its weight is 662 g, 18 g lighter than iPad 1 and 10g heavier than iPad 3.

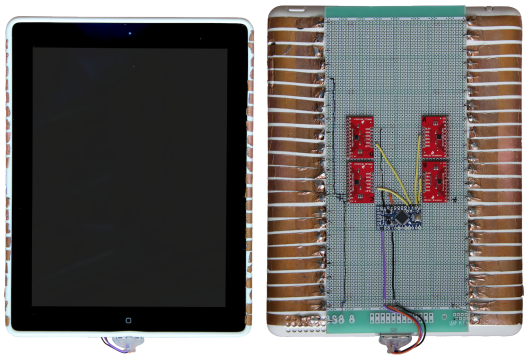


Figure 4. Photos of iGrasp prototype, showing an iPad 2, an Arduino board, four capacitive touch sensor controllers connected to 46 copper foils, all glued to an iPad 2 case.

Sensing Grasp Condition

Based on how users held the iPads in our observational study, we divided the 23 sensors on each side into top-half and bottom-half groups, for a total of four sensor groups. We merge all sensors in each group into a binary ON/OFF reading. Because all the sensor readings were already in binary form, merging sensor readings is simply an OR operation. We used the following heuristics: 1) if all of the sensor groups are OFF, then the user is not grasping the device. 2) for one-handed grasps, there are two grasps that we observed in our observational study. The first one is grasping a tablet by its edge using only one hand. This results in one or both sensor groups on one side showing ON, but both sensor groups on the other side would be OFF. We also observed a small number of users who would rest a tablet on their forearms and grasp the opposite top corner. In this case, a bottom sensor group on one side and the top sensor group on the other side would be ON, but the remaining two sensor groups would be OFF. 3) all other sensor group readings are considered to be two-handed grasps.

Sensing Grasp Position

The current prototype has 23 sensors on each of iPad's long sides (24.2 cm), so it has about 1 cm resolution in sensing grasp position. On each side, we label each sensor from the bottom-most one as $S_i, i = 0 \cdots 23$, and the hand position H is calculated using

$$H = \frac{\sum_{S_i \in S} i}{|S|}$$

where $S = \{S_i |$ sensor i is touched $\}$. To determine where to display the keyboard based on the sensed grasp position, the user could optionally complete a calibration process. Our calibration process has each user grasp the top, middle, and bottom third of the iPad and also manually move the keyboard to a comfortable position K for each of the grasps. We then use a linear regression function generated by a set of (H, K) pairs to calculate the keyboard position given a sensed hand position.

EVALUATION

We evaluated iGrasp's automatic layout and position adaptations by conducting two typing studies under several grasp conditions and postures:

1. comparing *iGraspSwitch*, which automatically switches to the memorized layout and position, to the manually adjustable keyboard.

2. investigateing whether *iGraspPosition*, which additionally positions the keyboard to users' grasp location, improves upon *iGraspSwitch*.

For both user studies, we measured the initial placement time, which is the time that elapsed between the keyboards appearing on screen until the time that users begin to type. The more closely the keyboard layout and position matches the users' preferences, the more quickly the users would be able to begin typing and the lower the initial placement time would be. Conversely, if the keyboard layout and position are not ideal, users would either need to move their hands to accommodate the keyboard layout and position, or need to manually adjust the keyboard to suit their preferences before they begin to type. We also recorded the total task completion time, and typing speed which additionally captures how well the layout matches the users' preferences. We used one way ANOVA and paired t-test to see if the result was significantly different. Furthermore, users reported subjective preferences using five-point Likert scales.

iGraspSwitch vs Manual

Our research question in this study is how our automatic keyboard layout and position switching affect task performance and user preferences.

Study Design

In order to compare our automatic keyboard adaptation and the manual adjustment, we are interested in understanding how each approach performs for the following three grasp conditions: no grasp ($0G$), one-handed grasp ($1G$), and two-handed grasp ($2G$). Specifically, there are 6 possible grasp transitions that users would normally make as they use their tablets in different settings: $0G \rightarrow 1G$, $0G \rightarrow 2G$, $1G \rightarrow 0G$, $1G \rightarrow 2G$, $2G \rightarrow 0G$ and $2G \rightarrow 1G$.

At the beginning of the user study, participants went through a 5-minute training session to practice how to adjust the keyboard layout and position. They also practiced typing tasks to identify their preferred keyboard layout and position for each of the grasp conditions. During the user study, participants could freely adjust the keyboard layout and position.

In order to have participants start each typing task while grasping the devices naturally, each participant first performed one of the six grasp transitions before starting the task. For each typing task, participants touched an input text field to activate the keyboard, then typed a phrase that was randomly selected from the Makenzie and Soukoreff phrase set [18]. If the input field was occluded by the keyboard, the test app would scroll automatically to make it visible. Each users went through all six possible transitions for the iGrasp condition and for the manual condition, for a total of 12 typing tasks. The order of iGrasp and the manual keyboard, as well as the starting grasp conditions were counter-balanced. Participants filled out a questionnaire on demographics and preferences after they finished all the tasks.

Analysis

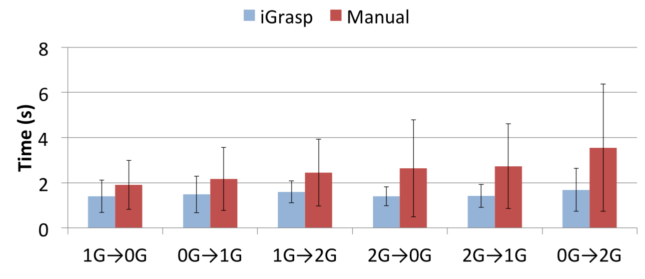
We recruited 18 participants (6 female) from our university population. The age of participants ranged from 21 to 25. All were regular computer users and had experience with touchscreen devices. Three of the participants had their own tablets for more than three months.

We quantified participants' grasp movement by summing up differences in a sequence of sensed hand positions H. That is, the grasp distance moved D_{total} is:

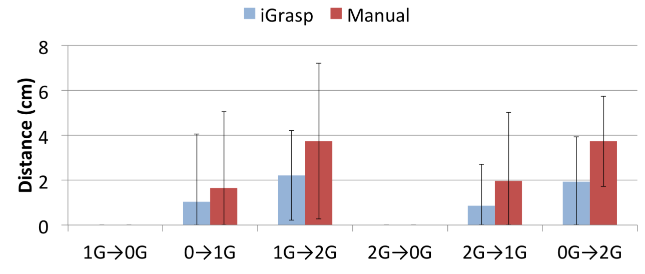
$$D_{total} = \Sigma_{t=2}^{T} |H_t - H_{t-1}|$$

where H_T is the last hand position sensed before task completion. The initial grasp distance moved was calculated in the same manner but summed up to the time the first character was typed.

Figure 5a) shows that average initial placement time, which was the time between the keyboard being activated and the time that the first character was typed. iGrasp improved the initial placement time by 42% compared to the current manual approach (1.49s vs 2.57s). This difference is statistically significant: ($F_{1,118} = 12.263, p < 0.001$). We also ran a two-way ANOVA of techniques (iGrasp vs manual) and conditions (6 transitions). The main effect of conditions is not significant ($p = 0.5$) and the interaction effect of techniques and conditions is not either ($p = 0.95$). Figure 5b) shows the average initial grasp movement distance, which was the total distance the users' hands moved until the first character was typed. The two posture transitions to the no-grasp condition ($1G \rightarrow 0G$ and $2G \rightarrow 0G$) have no data because the participants were not grasping the device. For all other conditions, the average distance reduced by 53% from cm to cm.



(a) Initial Placement Time



(b) Initial Grasp Movement

Figure 5. The initial placement time and initial grasp movement distance for iGrasp and manual layout switching. The error bars represent one standard deviation.

We also found that iGrasp had some improvement in terms of words per minute (WPM) and error rate; however, the difference was not statistically significant in our 18-person user study. The typing speed for iGrasp vs. manual mode was 26.6 vs. 24.8 WPM ($p = 0.61$), and the error rate was 4.3% vs 4.4% ($p = 0.95$). A much larger user study may help draw statistically significant conclusions.

In addition, we asked participants to rate the ease of use of both iGrasp and the manually adjustable keyboard on a fve-point Likert scale, and the participants rated iGrasp significantly higher at 4.2 vs 2.9 for manual adjustment. The difference in rating was also statistically significant ($F_{1,18} = 23.4, p < 0.001$).

iGraspSwitch vs iGraspPosition

Our research question in this study is how continuous positioning, which tracks the exact grasp position and moves the keyboard to that position, affects users' performances and subjective preferences.

Study Design

This study focuses on comparing different approaches to position the keyboard. In order to control for the different layouts (split vs merged) and layout transitions, we focus on the two-handed grasp that supports more usage scenarios, and only use the corresponding split layout that was preferred by the most participants from our first study (70%). The three positioning techniques are the following: 1) docked at the bottom, which is the default position on iOS and the only position supported by Android 4.0, 2) iGraspSwitch that shows the keyboard at its last-used position, and 3) iGraspPosition that continuously tracks the exact grasp position and shows the keyboard at that position the moment the keyboard is activated.

At the beginning of the user study, participants went through a 5-minute training session to practice how to adjust the keyboard layout and position. Participants were then asked to do calibration in each posture for us to build a personalized model on positioning the keyboard based on each users' particular grasp positions. Our testing application instructed participants to grasp the tablet at three different positions and asked the participants to adjust the keyboard to their most comfortable position. Participants then typed a short phrase while the participant's sensed hand position H and the keyboard position K was recorded to build a regression model.

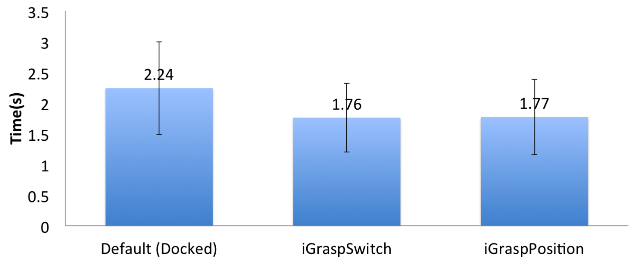
To thoroughly evaluate how well each positioning approach works, we needed the participants to perform typing tasks with a large number of postures that people would normally take when using tablets. Also, because we were primarily interested in the initial placement time between the different keyboard positions, shorter words that enable more trials would be preferable.

We asked the participants to go through the following sequence of five postures: lying down, lying on one side, sitting and leaning back, sitting, and standing. For each posture, the participants first picked up the tablet, activated the keyboard, completed a typing task, then placed the tablet down before transitioning to the next posture. Similar to our second user study, picking up the tablet at the beginning of each posture is important to ensure that the participant would be grasping it naturally, and not affected by the previous task or posture. The short typing tasks we used was based on the 100 most-frequently-used words that were three characters long and contained characters on both sides of the split keyboard.

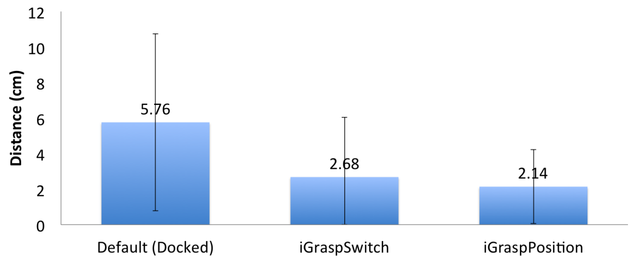
In order to prevent the participants from anticipating where the keyboard would appear, we did not reveal which positioning approach was used and we randomly shuffled the positioning order for each posture. Each participant needed to go through the posture sequence three times to experienced all possible keyboard positions and posture combinations. We asked each participant to perform two trials, so each user went through the sequence of postures six times for a total of 30 typing tasks (3 keyboard positions $\times$ 5 postures $\times$ 2 trials). After the participants finished the two trials, they completed an additional trial in which the keyboard positioning approach was revealed to them. We then asksed the participants to fill out a questionnaire and the NASA perceived workload index (TLX) [12]. We also interviewed them to get their feedback.

Analysis

We recruited another 18 participants (8 female) from our university population that were completely different from our second study. The age of the participants ranged from 19 to 24. All were regular computer users and had experience with touchscreen devices. Six of the participants owned tablets.



(a) Initial Placement Time



(b) Initial Grasp Movement

Figure 6. Inital placement time and initial grasp distance moved for the three keyboard positioning approaches: docked, position switching, and continuous positioning. The error bars represent one standard deviation.

Figure 6 shows the average initial placement time and also the initial grasp movement, which is the distance the participants' hands had moved before typing the first character. As shown in Figure 6a), both *iGraspPosition* and *iGraspSwitch*'s had nearly identical initial placement time. The initial placement time and task completion time significantly improved by 21% and 22% ($F_{2,537} = 32.13, p < 0.001$) over the default, docked position, respectively.

The results show that *iGraspPosition* was the most accurate in placing the keyboard at users' comfortable typing location, causing the users' hands to move the least distance. The average total grasp movement of *iGraspSwitch* was 2.58cm. The 66% improvement over the default docked position is significant ($F_{2,537} = 46.37, p < 0.001$), but is not statistically significant ($p = 0.09$) from *iGraspPosition*. The average initial grasp movement distance of *iGraspPosition* was also the lowest at 2.14cm. The 63% improvement over the default position is statistically significant but is not statistically significant ($p = 0.25$) compared to *iGraspPosition*.

The results of subjective workload with the NASA TLX procedure are shown in Figure 7. The result indicates that on average the workload of *iGraspPosition* and *iGraspSwitch* are lower and the difference is statistically significant ($p < 0.05$) compared to the default keyboard. Howerver, there is no significant difference in each subscale between *iGraspPosition* and *iGraspSwitch* ($p > 0.05$).

In addition, participants rated the ease of use for each keyboard positioning approach using a 5-point Likert scale. The

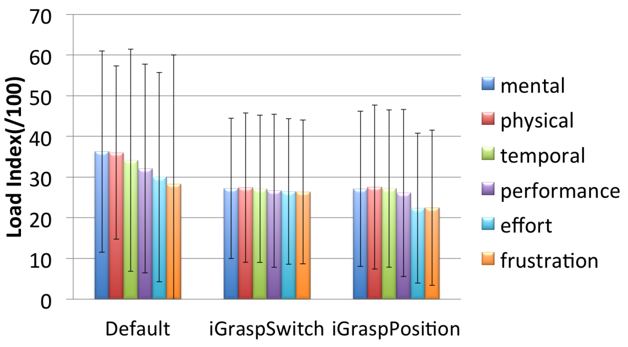


Figure 7. NASA TLX task load scores for the three keyboard positioning approachs.

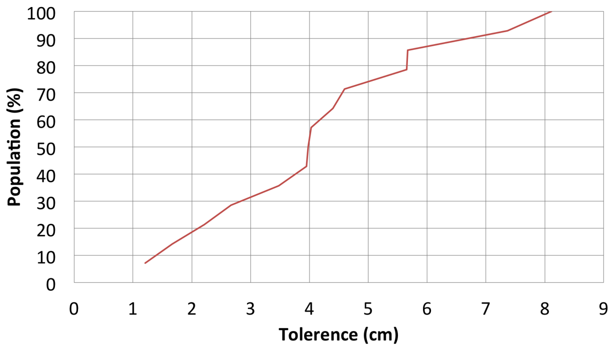


Figure 8. Cumulative distribution function of users' positioning error tolerance.

rating of the default keyboard is 1.33. *iGraspPosition* and *iGraspSwitch* were rated 3.94 and 3.77, respectively. The difference is significant ($p < 0.01$) between the default keyboard and iGrasp's approaches, but the difference is not significant ($p = 0.76$) between the two iGrasp modes.

To summarize our findings from the third user study, both *iGraspPosition* and *iGraspSwitch* approaches are significantly better than the docked keyboard position in terms of task performance, perceived task load, and participants' preferences. However, we did not observe statistically significant difference between the two iGrasp approaches in any of the metrics we analyzed.

DISCUSSION

Users' Tolerance of Positioning Error

Results from our third user study show that even though continuous positioning had shorter grasp movement distance, participants' typing performance was comparable. It suggests that users are able to tolerate some amount of positioning error. To better understand this tolerance, we analyzed each user's grasp position from *iGraspPosition* keyboard from our third user study. The optimal grasp position H_O was recorded when each user chose their preferred keyboard position. We then define *tolerance* as the distance from H_O to users' grasp positions at the end of each typing task (five postures $\times$ two trials). We use the position of the last character typed instead of the first character because users' hands may still be moving during the task, and the ending position is likely to be the more comfortable typing positon.

For each user, we calculated the maximum tolerance across the 10 tasks. The average tolerance across all users (N=18) is 4.2 cm. If we further drill down and separate the grasps above H_O and below H_O, we get a tolerance of 4.2 cm when users' grasp above the optimal typing postion, and a tolerance of 0.4 cm when users grasp below. This large difference suggests that it is much easier to reach below to type on a keyboard that is positioned too low, than to reach above to type on a keyboard that is positioned to high. Figure 8 shows the cummulative distribution function of tolerance. 50% of the users have a tolerance of 4.0 cm.

Based on this finding, it is better to implement iGraspSwitch than to implement iGraspPosition because it only needs four capacitive sensors. Each sensor should have a sensing area that covers half of each side of the device and have similar width as our prototype. Four sensors are needed instead of two because we observed some one-handed grasps in our observational study where users rested the bottom corners of the devices on their forearm and grasped the opposite top corner of the devices.

Limitations

In our pilot study, a participant wearing shorts reported that the keyboard position was far from his grasp position. It turned out that he was resting the device on his legs to support it, and the sensors near the bottom were reporting touch events. We adjusted the sensor layout and removed the two sensors at the bottom, leading to our current 46-sensor prototype. We are currently building an iGrasp prototype to support both landscape and portrait usages, and are experimenting with touch point clustering to identify contiguous grasp regions and then applying machine learning techniques to build a more robust grasp classifier.

Capacitive sensors have some inherent limitations. For example, wet hands and users wearing gloves would be difficult to detect correctly. When no grasps are detected, iGrasp would present the keyboard used in the no-grasp condition, which is most likely docked, and would provide the same user experience as the default keyboards. Users in these situations may also have trouble typing on the capacitive touchscreen keyboard.

Real-time Positioning

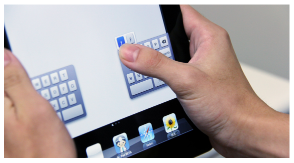
We implemented real-time keyboard positioning and tested it by our study with additional 5 users (age 19 to 24, 2 female). The system would track users' grasps and move the keyboard to match the users' hand position even when users were typing. All users who tested real-time positioning commented that the constantly-moving keyboard was distracting, and they could not type correctly due to keyboard drifting caused by slight changes in grasp while typing. We modified the positioning to stop updating its position after the first key is pressed in our user studies.

Enhancing Continuous Positioning

During our pilot studies, we encountered several challenges in implementing accurate continuous positioning. The first was how to stabilize the sensed grasp position, since users' hands may come in and out of contact with the sensors. When users grasp a tablet, movement such as stretching a thumb to reach a far target may cause the sensed hand position to change. We used moving average approach which buffered the last 30 sensor readings (0.5 seconds at 60 Hz) and returned the average value to minimized such drifting to improve positioning accuracy.

The second challenge was building an accurate personalized model to map a sensed grasp position to users' desired keyboard position. During our pilot studies, we used the second order polynomial regression. By examining users' models built from sets of (hand, keyboard) pair collected in different postures, we found all models were close to linear, so we used the linear regression in our final prototype to avoid overfitting. However, two participants in our pilot study reported that the keyboard positions were not accurate. One was lower than the desired position, and one was higher. We analyzed users' models again and found that the models in different postures were different. Therefore, the predicted result may be accurate in one posture, but exceed the user's tolerance region in another posture. Posture detection may be needed to switch between different models.

The reasons may be that users' eyes relative to the hands and the tablets vary between postures. Also, the comfortable typing range may also change depending on the relative angle between users wrists and the tablets. As shown in Figure 9, with the same grasp position, the user can reach the 'Y' key when holding the device at an 90 degree angle relative to his wrist, but cannot reach the same key when holding the device at a different angle.


(a) User could reach the top row when the angle between wrist and tablet is close to 90 degrees.

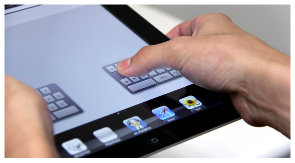

(b) User could only reach the middle row when the angle between wrist and tablet is about 0.

Figure 9. Users' reach, relative to their grasp position, changes when the angle between wrists and tablets changes.

CONCLUSIONS AND FUTURE WORK

Our studies have shown that most users have a different preferred virtual keyboard layout and position depending on how they were holding the tablets. We proposed iGrasp, a novel approach that uses grasp-sensing to automatically adapts the keyboard layout and position. Our evaluations show that participants were able to begin typing 42% earlier using the iGrasp's adaptive keyboard compared to manually adjustable keyboards. In addition, participants also rated iGrasp significantly easier to use (4.2 vs 2.9 on 5-point Likert scale). We also found that continuous position adaptation shows no statistically significant improvement over users' last-used positions.

Although our grasp condition classification heuristics worked well for the participants in our user studies, we are currently exploring machine learning techniques to generalize the types of grasps we can support. We are also exploring other types of grasp-based adaptive user interfaces, such as resizing the interaction area based on the grasp size of the users. Last, we plan to open-source our sensor design and grasp sensing framework to make it easier for the research community to explore grasp-based interface and interaction.

REFERENCES

1. Benko, H., Izadi, S., Wilson, A. D., Cao, X., Rosenfeld, D., and Hinckley, K. Design and Evaluation of Interaction Models for Multi-touch Mice. In *GI '10*, ACM (2010), 253–260.
2. Butler, A., Izadi, S., and Hodges, S. SideSight: multi-touch interaction around small devices. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, ACM (2008), 201–204.
3. Cheng, L.-P., Hsiao, F.-I., Liu, Y.-t., and Chen, M. Y. iRotate : Automatic Screen Rotation based on Face Orientation. In *Proc. CHI '12*, ACM (2012), 2203–2210.
4. Cheng, L.-P., Lee, M.-H., Wu, C.-Y., Hsiao, F.-i., Liu, Y.-t., Liang, H.-S., Chiu, Y.-C., Lee, M.-S., and Chen, M. Y. iRotateGrasp: Automatic Screen Rotation based on Grasp of Mobile Devices. In *Porc. CHI '13*, ACM (2013).
5. Faraj, K. A., Mojahid, M., and Vigouroux, N. BigKey : A Virtual Keyboard for Mobile Devices. In *Proc.eedings of the 13th International Conference on Human-Computer Interaction. Part III: Ubiquitous and Intelligent Interaction*, no. 60, Springer-Verlag (Berlin, Heidelberg, 2009), 3–10.
6. Fitrianie, S., and Rothkrantz, L. J. M. An adaptive circular (text) input for handheld devices. In *Proc. CompSysTech '11*, ACM (2011), 405–410.
7. Goel, M., Findlater, L., and Wobbrock, J. O. WalkType : Using Accelerometer Data to Accommodate Situational Impairments in Mobile Touch Screen Text Entry. In *Proc. CHI '12*, ACM (2012), 2687–2696.
8. Goel, M., Wobbrock, J. O., and Patel, S. N. GripSense : Using Built-In Sensors to Detect Hand Posture and Pressure on Commodity Mobile Phones. In *Proc. UIST '12*, ACM (2012), 545–554.
9. Gu, J., and Lee, G. TouchString: a flexible linear multi-touch sensor for prototyping a freeform multi-touch surface. In *Adjunct Proc. UIST '11*, ACM (2011), 75–76.
10. Gunawardana, A., Paek, T., and Meek, C. Usability Guided Key-Target Resizing for Soft Keyboards. In *Proc IUI '10*, ACM (2010), 111–118.
11. Harrison, B. L., Fishkin, K. P., Gujar, A., Mochon, C., and Want, R. Squeeze me, hold me, tilt me! An exploration of manipulative user interfaces. In *Proc. CHI '98*, CHI '98, ACM (1998), 17–24.

12. Hart, S. G., and Staveland, L. E. Development of Nasa Tlx (Task Load Index): Results of Empirical and Theoretical Research. Human Mental Workload. In *Human Mental Workload* (1988), 139–183.

13. Himberg, J., Häkkilä, J., Kangas, P., and Mäntyjärvi, J. On-line personalization of a touch screen based keyboard. In *Proc. IUI '03*, ACM (2003), 77–84.

14. Hinckley, K., Pierce, J., Sinclair, M., and Horvitz, E. Sensing techniques for mobile interaction. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, vol. 2, ACM (2000), 91–100.

15. Hinckley, K., and Sinclair, M. Touch-sensing input devices. In *Proc. CHI '99*, no. May, ACM (1999), 223–230.

16. Kim, K., Chang, W., Cho, S.-j., Shim, J., Lee, H., Park, J., Lee, Y., and Kim, S. Hand grip pattern recognition for mobile user interfaces. In *Proc. AAAI '06*, vol. 21, MIT (2006), 1789–1794.

17. Kristensson, P.-O., and Zhai, S. SHARK2: a large vocabulary shorthand writing system for pen-based computers. In *Proc. UIST '04*, no. 2, ACM (2004), 43–52.

18. MacKenzie, I. S., and Soukoreff, R. W. Phrase sets for evaluating text entry techniques. In *Proc. CHI '03 extended abstracts*, ACM (2003), 754–755.

19. Mayzner, M., and Tresselt, M. *Tables of Single-letter and Diagram Frequency Counts for Various Word-length and Letter-position Combinations*. Psychonomic monograph supplements, v. 1, no. 2, 1965.

20. Nybergh, K., and Himberg, J. Touch detection system for mobile terminals. In *Proc. MobileHCI '05*, ACM (2004), 331–336.

21. Oliver, B. M. Time Domain Reflectometry. *HP Journal* (1964), 15(6).

22. Pai, D. K., VanDerLoo, E. W., and Sadhukhan, S. The tango: A tangible tangoreceptive whole-hand human interface. In *Proc. World Haptics '05*, IEEE Computer Society (2005), 141 – 147.

23. Rudchenko, D., Paek, T., and Badger, E. Text text revolution: a game that improves text entry on mobile touchscreen keyboards. In *Proc. Pervasive '11*, Pervasive'11, Springer-Verlag (2011), 206–213.

24. Rudeck, F., and Baudisch, P. Rock-Paper-Fibers : Bringing Physical Affordance to Mobile Touch Devices. In *Proc. CHI '12*, no. d, ACM (2012), 1929–1932.

25. Sato, M., Poupyrev, I., and Harrison, C. Touché: Enhancing Touch Interaction on Humans, Screens, Liquids, and Everyday Objects. In *Proc. CHI '12*, ACM (May 2012), 483–492.

26. Savage, V., Zhang, X., and Hartmann, B. Midas : Fabricating Custom Capacitive Touch Sensors to Prototype Interactive Objects. In *Proc. UIST '12*, ACM (2012), 579–588.

27. Schmidt, A., Aidoo, K., Takaluoma, A., Tuomela, U., Van Laerhoven, K., and Van de Velde, W. Advanced interaction in context. In *Handheld and ubiquitous computing*, Springer (1999), 89–101.

28. Schmidt, A., Beigl, M., and Gellersen, H. There is more to context than location. *Computers & Graphics 23*, 6 (1999), 893–901.

29. Song, H., Benko, H., Guimbretiere, F., Izadi, S., Cao, X., and Hinckley, K. Grips and gestures on a multi-touch pen. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, vol. 21, ACM (2011), 1323–1332.

30. Taylor, B., and Bove Jr, V. Graspables: grasp-recognition as a user interface. In *Proceedings of the 27th international conference on Human factors in computing systems*, ACM (2009), 917–926.

31. Veldhuis, R., Bazen, A., Kauffman, J., and Hartel, P. Biometric verification based on grip-pattern recognition. In *Security, Steganography, and Watermarking of Multimedia Contents, volume 5306 of Proceedings of SPIE*, vol. 31, Centre for Telematics and Information Technology University of Twente (2004), 634–641.

32. Wagner, J., Huot, S., and Mackay, W. BiTouch and BiPad : Designing Bimanual Interaction for Hand-held Tablets. In *Proc. CHI '12*, ACM (2012), 2317–2326.

33. Wimmer, R. FlyEye: grasp-sensitive surfaces using optical fiber. In *Proc. TEI '10*, ACM (2010), 245–248.

34. Wimmer, R., and Baudisch, P. Modular and deformable touch-sensitive surfaces based on time domain reflectometry. In *Proc. UIST '11*, no. 1, ACM (2011), 517–526.

35. Wimmer, R., and Boring, S. HandSense: discriminating different ways of grasping and holding a tangible user interface. In *Proc. TEI '09*, ACM (2009), 359–362.

36. Yu, N., Tsai, S., Hsiao, I.-c., Tsai, D., Lee, M., Chen, M., Hung, Y., and Others. Clip-on gadgets: expanding multi-touch interaction area with unpowered tactile controls. In *Proc. UIST '11*, ACM (2011), 367–372.