Journal of Interaction Science
a SpringerOpen Journal

RESEARCH Open Access

Predicting upper-bound text entry speeds for discrete-tilt-based input on smartphones

Sandi Ljubic[1*], Vlado Glavinic[2] and Mihael Kukec[3]

Abstract

Motion sensors integrated into contemporary smartphones allow the introduction of new mobile interaction paradigms, here including tilt-based input control in the mobile context. Namely, as opposed to existing implementations that typically apply continuous feedback on tilting, we define Pitch and Roll movement sequences that change the orientation of the mobile device as discrete-tilt input primitives. The respective commands are then used to manage text entry within three discrete-tilt-based methods thus introduced: keyboard bisection, single cursor, and quad cursor. Each method is based on the use of a particular QWERTY-based keyboard layout with related strategy for character input. We model upper-bound text entry speeds for the input methods, taking into account both movement aspects and language context. The movement model corresponds to both the tilt-based shortest path between two consequent characters, which is theoretically defined, and the time of discrete-tilt execution, which is obtained from user testing experiment we conducted. The linguistic model, comprising digraph statistics, is constructed basing on available English corpora. This modeling approach provides discrete-tilt-based text entry speed predictions representing efficiency rates for expert behavior, i.e. for optimal performance. The results obtained enable the evaluation of the proposed designs without need to test with real users, and can furthermore serve as a baseline for efficiency of text entry implementations that rely on discrete tilt.

Keywords: Text entry; Tilt interaction; Mobile devices; Predictive models

Introduction

"Texting on the move" has become a dominant phenomenon in the current mobile computing environment. The concept of quick message exchange has evolved from using feature mobile phones and SMS to taking advantage of touchscreen smartphones and messaging and social networks services. Furthermore, text entry tasks once projected for desktop-based usage only, such as writing e-mails or Web searches, are now also regularly executed in the mobile domain. However, one problem seems to persist: typing on a mobile touchscreen device is slow, uncomfortable, inaccurate, and generally less efficient in comparison to typing on physical keyboards [1]. On the other hand, touchscreen keyboards are software based, so they can support different types of customizations and built-in algorithms for automatic adaptations. They can be easily programmed to accommodate different layouts, screen sizes, device orientations, and languages, as well as be provided with dictionary support and auto-correction features. Many HCI-based research efforts now focus on developing novel soft keyboard models and related input methods in order to provide trouble-free and more efficient text entry.

In general, users are unwilling to waste their time in order to learn new text-input techniques [1], which could be the reason why QWERTY still stands as the default keyboard layout in contemporary mobile touchscreen devices. When it comes to interaction modalities, the prevailing technique used for text entry across the most of the available soft keyboards consists of direct touch (*Tap*), although gestural text input has also attained some acceptance with *Swipe* [2,3]. As contemporary smartphones are in addition provided with built-in motion sensors (e.g. accelerometer, gyroscope, magnetometer) which allow monitoring device movements, such as tilt, shake, rotation or swing, tilt-based interaction in the mobile context is made available, as well as its utilization for text entry. Nevertheless, tilt

* Correspondence: sandi.ljubic@riteh.hr
[1]Faculty of Engineering, University of Rijeka, Vukovarska 58, HR-51000 Rijeka, Croatia
Full list of author information is available at the end of the article

interaction modality is hardly ever considered to be a part of the final text entry solution, even when multimodal techniques were being addressed. In this paper we provide three text entry methods that (i) rely on the discrete-tilt concept, (ii) support typing by wrist motions only, and (iii) work with a QWERTY-based layout. We are fully aware that tilt-only methods cannot provide high text entry rates, but believe that they could support short messaging scenarios where typing on small screens appears to be particularly inconvenient. The proposed methods are analyzed a priori with the use of predictive modeling, focusing on discrete-tilt-based text entry efficiency.

The paper is structured as follows. In Related work section we describe existing tilt-based text entry solutions that usually do not use the QWERTY layout, and/or rely on continuous application feedback. Pitch and Roll as input primitives for mobile devices section introduces the tilt-based concept which supports an interaction less relying on visual feedback; here we define Pitch and Roll movement sequences as discrete-tilt input primitives for mobile devices. In Discrete-tilt Pitch and Roll based text entry methods section we apply the proposed input commands on text entry in the mobile, by introducing three different text entry methods that utilize both QWERTY-based keyboard layouts and discrete-tilt-based strategies for character selection. Modeling upper-bound text entry speeds section deals with the proposed methods, targeting error-free interaction and expert-level input efficiency. We describe the used modeling methodology which consists of: (i) user experiment which is conducted to determine the time of discrete-tilt execution, (ii) movement models able to predict the total time required to enter a particular character, (iii) a constructed linguistic model for the English language that provides a matrix of digraph probabilities, and (iv) calculated predictions for discrete-tilt-based text entry speeds. In Discussion section we evaluate the proposed designs with respect to the predictions hence obtained, and discuss the model limitations. The last section offers a brief recapitulation, including the outline of our future research plan.

Related work

The concept of interpreting data from a mobile device's motion sensors as user's intentional control action has already been examined as a prospective mobile interaction technique. Initial research efforts have resulted with implementations that provide possibilities for scrolling, changing screen orientation, as well as navigation through lists and menus. Mobile gaming has also benefited from both speed and direction control based on motion (a typical example would be a simple game wherein moving a virtual ball through a maze is achieved with device tilting).

Tilt-based text entry techniques, which are of particular importance in this work, were firstly addressed by *Unigesture* [4], *TiltType* [5], and *TiltText* [6] prototypes. The *Unigesture* system relies on specially developed hardware, and uses a zone-based keyboard layout with characters grouped into eight input zones, hence abandoning the QWERTY outline. Each zone, containing 4 characters at most, corresponds to a distinct tilt direction. When using the *Unigesture* approach, individual letters are not inputted directly, but a word is predicted by an inference engine, basing on zones been "visited" in the tilting sequence. *TiltType* was designed for small watch-like devices and requires a two-hand interaction. Its interface is multimodal, since character selection requires both button pressing (for letter group selection) and device tilting (for the desired letter pick). Just like in the *Unigesture* prototype, zone-based keyboard layout and tilts with eight compass directions are used. The *TiltText* system is a tilt-to-write solution with multimodal interaction that augments text entry on the standard 12-button keypads with multiple letters mapped to a particular key. The *MultiTap* technique, regularly used for character disambiguation in such designs, is replaced with tilt support: selection of a character is achieved by pressing a key followed by tilting the phone in one of the four directions.

More recent research on this topic includes more sophisticated accelerometer-based text input. *GesText* [7] represents a mid-air text entry design that relies solely on tilt input, with the respective prototype being based on the Wiimote device and a remote 32″ screen. Two keyboard layouts were proposed in the respective work, and test results confirmed that the zone-based layout supported by a small set of simple tilt gestures is both more efficient and subjectively preferred in comparison to the alphabetically ordered layout. Here, the QWERTY layout was not included in the study. *WalkType* [8] represents an adaptive text entry system which does not use tilt-based interaction, but is worth mentioning in this place because it applies accelerometer data to improve typing on a smartphone with a standard QWERTY keyboard. Automatic compensation of the extraneous movement while walking appeared to be a valuable support, since typing speed and accuracy increased in experiments with walking participants. The *Dasher* project [9] provides a novel text entry method which discards the traditional keyboard concept, and introduces zoom-and-point interaction for selecting "flying letters" in the 2D space. Its main advantage is a well-designed word-completion feature supported by pointing gestures. While pointing can be achieved with various techniques/devices in different contexts of use (e.g. joystick, touchscreen, trackball, mouse, head-tracking, eye-tracking), in the smartphone domain the *Dasher* assumes tilt movements. Fitton et al. [10] provided the

experimental interface for investigating the performance of tilt-based target selection intended for text input on mobile devices. The respective keyboard layout was reduced to a 5×3 grid with 15 alphabetically ordered characters. Selection was achieved by tilt-based positioning of a rolling ball within the appropriate square and maintaining its position for a particular amount of time. This solution was tested by a set of teenage users, on a 7″ tablet device, but related tasks were based on target selection exclusively, rather than on a "real world" text entry scenario. However, qualitative results demonstrated enthusiasm for tilt-based text entry as an idea.

Finally, in our previous work we introduced a fully functional text entry prototype for Android devices in which the standard touch modality could be used in combination with *Pitch* and *Roll* movements [11]. We used the concept of discrete tilts for input primitives, as well as a special QWERTY-based keyboard layout and an original input scheme for character selection. The prototype was successfully tested on four devices from both smartphone and tablet class. This proposed text entry method inspired us to develop two additional tilt-based solutions that are described in detail in the following, together with the original one, and comparatively analyzed using predictive modeling.

Pitch and Roll as input primitives for mobile devices

In order to simplify the tilt-only interaction design within our text-entry methods, we decided to use only four basic input primitives: *Pitch* and *Roll* movements along the device's lateral and longitudinal axes are selected as the only valid tilt actions (Figure 1), i.e. pitch up and down, and roll left and right.

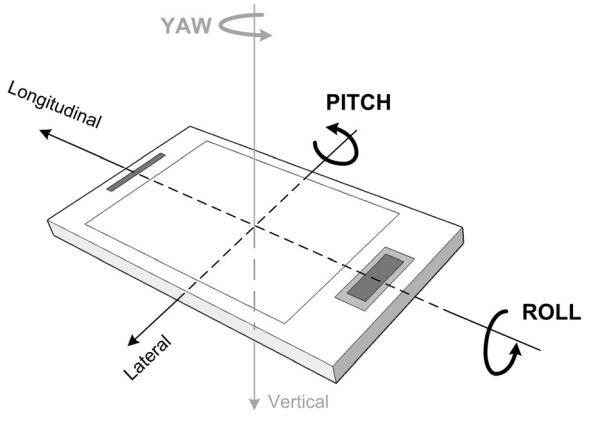

Figure 1 *Roll*, *Pitch* and *Yaw* movements within orientation-based smartphone interaction. In our solutions, tilting around the vertical axis is not used for text entry support.

The four basic tilt movements are mapped into the corresponding commands: twisting the smartphone up/down along the lateral axis generates *Pitch Down* and *Pitch Up*, while rotating left/right along the longitudinal axis results with *Roll Left* and *Roll Right* (Figure 2).

A neutral position is assumed when the device is parallel to the horizontal plane. Once the motion sensors detect deflection from the zero pitch/roll angles, we can say that the device enters in its tilt state. Here we define *continuous tilt* as an interaction modality in which sensor-based data are constantly translated into an application feedback as long as the device dwells in the tilt state. For example, we use continuous tilt in a mobile game for controlling both the direction and speed of a virtual ball in the interface. In such a game, the position of the virtual ball is constantly recalculated and the interface is redrawn, resulting with a smooth ball rolling effect according to tilt directions and angles. On the other hand, for our text entry methods we propose a *discrete-tilt* concept, which supports interaction less relying on visual feedback. The main idea behind this concept is that a well defined sequence of tilt movements corresponds to a distinct input primitive that can be used for application control. A discrete-tilt example for the *Roll Left* input primitive is illustrated in Figure 3.

Since it is very cumbersome to keep a mobile device in its neutral position with zero pitch/roll angles, especially when walking, we define a *neutral position zone* – an angle-based offset wherein the device is assumed to be motionless. Basically, tilting within the neutral position zone will never invoke any particular action, hence involuntary movements will be filtered out. When the device leaves the neutral position zone, it means that the user is starting to execute the tilt sequence intentionally. To make a valid discrete tilt, the pitch/roll threshold angle should be exceeded by tilting in the wanted direction. The threshold angle is introduced in order to augment the disambiguation between spontaneous and deliberate movements, as well as to provide required wrist strokes in line with natural interaction. It is reasonable to define separate threshold angles for *Pitch* and *Roll* because constraints of different wrist movements (flexion, pronation, supination, ulnar and radial deviation) are not equal [12]. Once the threshold angle has been surpassed, the device is assumed to be positioned in the *valid tilt zone*. Finally, an immediate backward movement from the valid tilt zone to the neutral position zone results with the input primitive invocation.

We furthermore enriched the discrete-tilt concept with the possibility of dwelling in the valid tilt zone. Specifically, if the smartphone is retained in the valid tilt zone for a well-defined amount of time before returning to the neutral position zone, a different input primitive could be invoked. We call this type of movement sequence a *long tilt*. Since the previously described

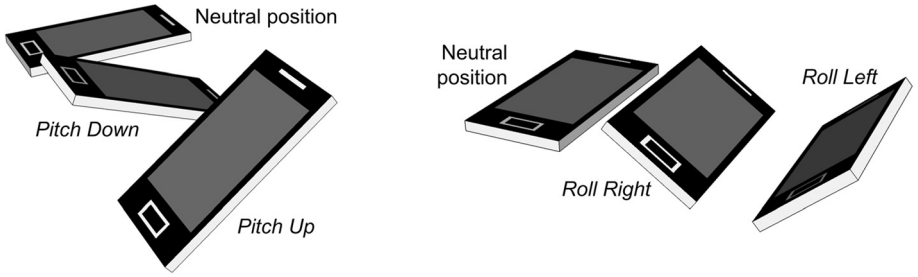

Figure 2 Smartphone basic tilt movements used in our text entry methods: *Pitch Down* (PD), *Pitch Up* (PU), *Roll Right* (RR), and *Roll Left* (RL).

interaction can result either with the regular or the long tilt, we can state that altogether eight input primitives can be generated using our discrete-tilt concept.

Pilot testing during the discrete-tilt implementation phase helped us in defining both the angle and the time values that appear to be the most convenient for our text entry solution. Consequently, we use: ±10° offset for the neutral position zone, 45° threshold angle for *Roll* movements, 30° threshold angle for *Pitch* movements, and 2 sec dwell time to differentiate between regular and long tilts.

The respective state diagram for discrete-tilt-based input primitive recognition process is depicted in Figure 4.

Discrete-tilt Pitch and Roll based text entry methods

In this section we introduce three different text entry methods that rely on the discrete-tilt concept. The design and interaction are described with special emphasis being on keyboard layouts and corresponding input schemes description. For every interaction method we calculate the number of discrete tilts (input primitives) required to select a given character.

Keyboard bisection method

The *keyboard bisection* (KB) method was already described in [11], as a part of our previous efforts to augment touchscreen text entry with multimodal interaction by supporting both tapping and tilting. The main idea was to provide discrete-tilt-based control over visual enlargement of a particular part of the current keyboard layout, and thus to enable easier character selection. Figure 5 shows the proposed keyboard layout design in its initial state, along with possible layout partition according to available basic tilt-based input primitives.

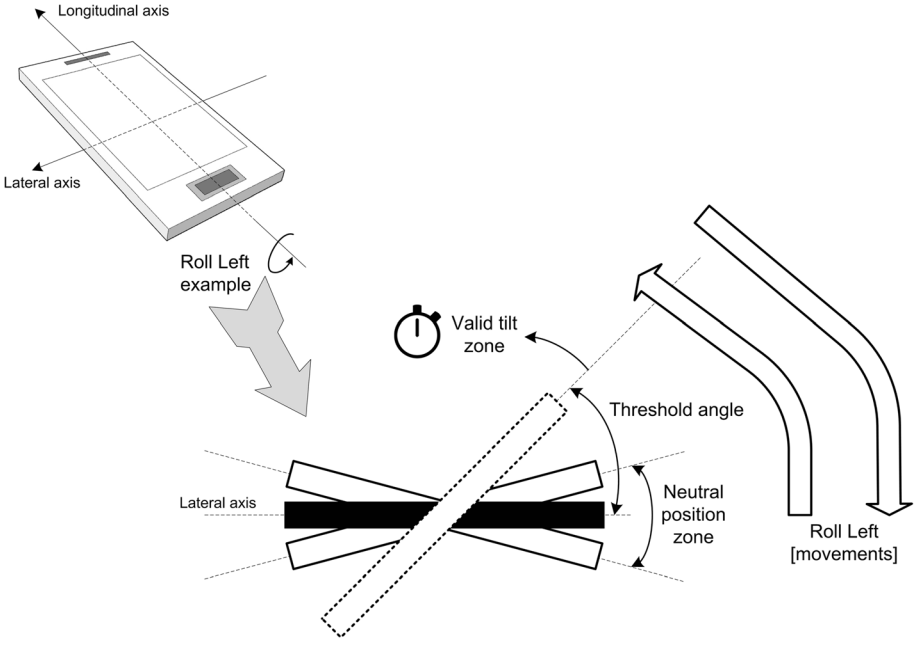

Figure 3 Anatomy of discrete tilt for *Roll Left* input primitive.

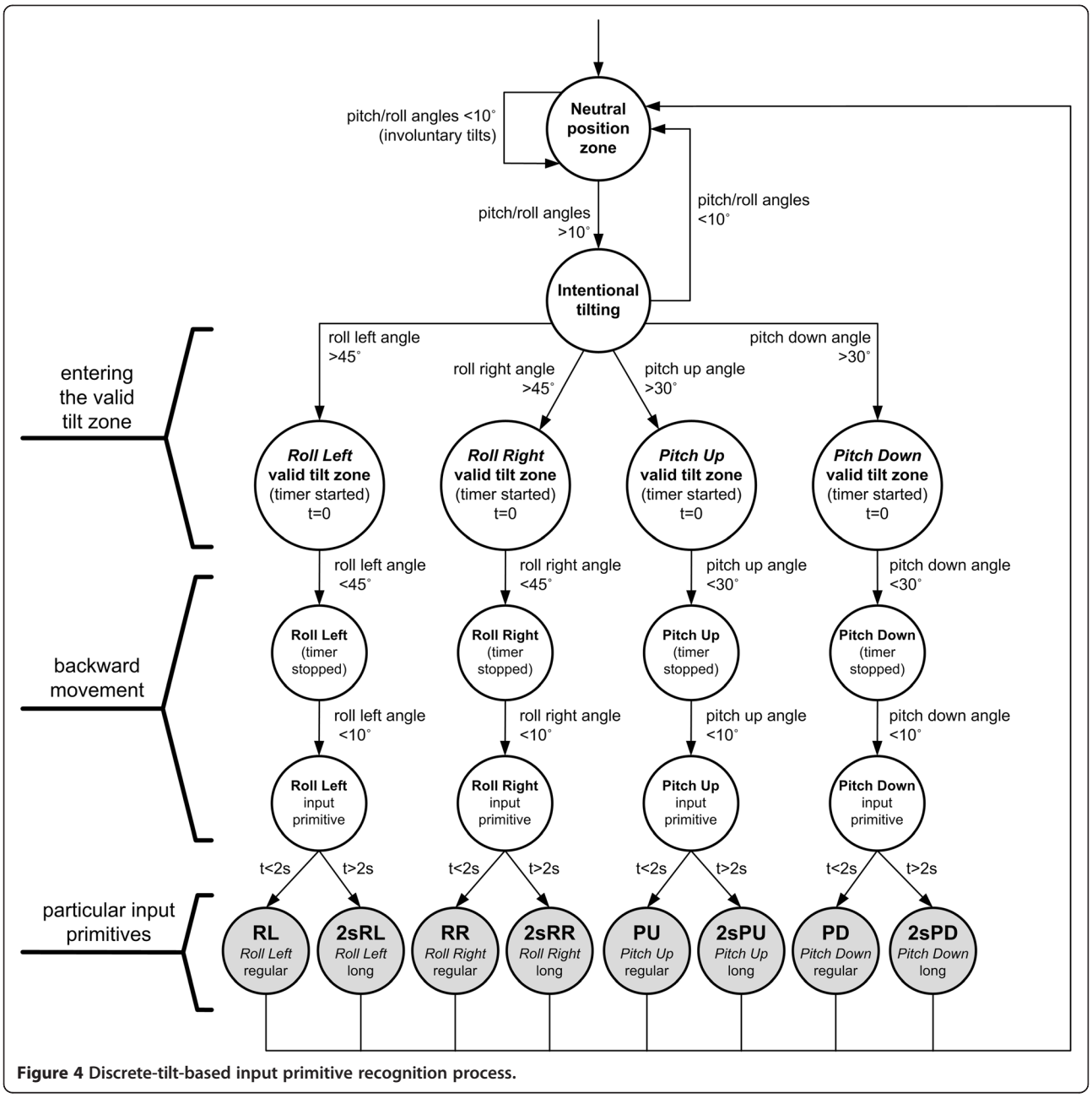

Figure 4 Discrete-tilt-based input primitive recognition process.

This keyboard configuration includes 15 interactive elements, each containing a set of four symbols for related characters and/or actions. It implements the zone-based approach, preserving the QWERTY layout through letter arrangement within eight upper left elements. Along with letters, the initial keyboard configuration additionally includes digits, punctuation marks, and supplementary symbols. Tilting the smartphone in the appropriate direction will cause a "keyboard bisection" resulting in the display of the selected half of the current layout. Gradually reducing the available character set using discrete tilts, from the starting *Level-1* to the lowest *Level-5*, enables text entry with *Pitch* and *Roll* movements exclusively, as shown in Figure 6.

The KB method benefits from multimodal interaction, as character selection is also enabled through direct touch from *Level-2* onwards. Since layout bisection results with larger buttons, each user is enabled to individually decide which level is most suitable for precise touch selection. Hence, altogether three modalities can

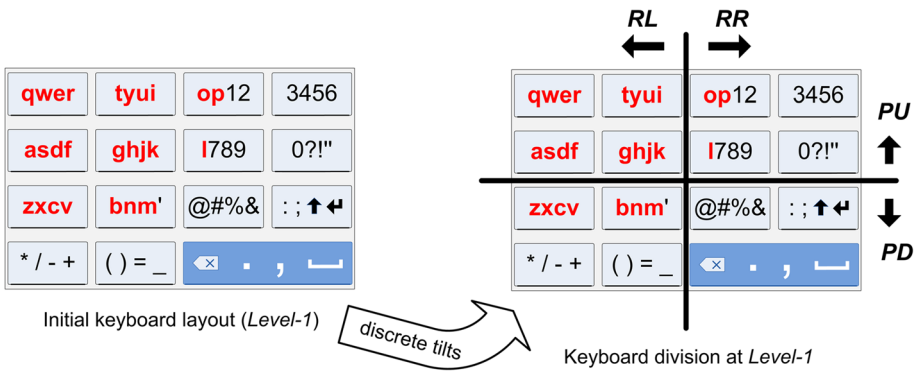

Figure 5 Keyboard bisection principle.

be used for text entry with the KB method: tap-only, tilt-only, and tilt-and-tap. In the continuation of the paper we investigated tilt-only.

The largest element in the initial keyboard layout contains four symbols for the following characters/actions: backspace, period, comma, and space. Corresponding buttons are highlighted using different background color in order to notify the user on the possibility of shortcut activation. Namely, these four common options can be alternatively selected using long tilts, without the need to "navigate" through several layout levels. Shortcut options can be selected regardless of the currently active character layout (*Level-1* – *Level-5*).

Within the KB method, selection of a particular character always involves exactly five discrete tilts, with the exception of shortcuts that can be selected either with one long tilt or four regular tilts. We define $\mathbb{S}$ as the set of four existing shortcuts, $\mathbb{A}$ as the set of all other available characters from the proposed keyboard layout, and N_{KB} as the minimal number of discrete tilts required for the selection of character j that follows a previously entered i:

$$N_{KB}(i,j) = \begin{cases} 5, & j \in \mathbb{A} \quad (\textit{regular tilts}) \\ 4, & j \in \mathbb{S} \quad (\textit{regular tilts}) \\ 1, & j \in \mathbb{S} \quad (\textit{long tilts}) \end{cases}$$

Single cursor method

Although the KB method works with the QWERTY-like configuration, it still requires a considerable learning time in order for the user to become more familiar with changing layouts. Furthermore, a transformable keyboard could be a troublesome concept for users who have a preference towards interface positional consistency. So as to provide a more "steady" design with well-known character layout, we introduce a *single cursor method* (SC). The proposed keyboard comes with a 12×3 character scheme, consisting mainly of alphabet letters in the QWERTY alignment. As opposed to the KB keyboard, there are no

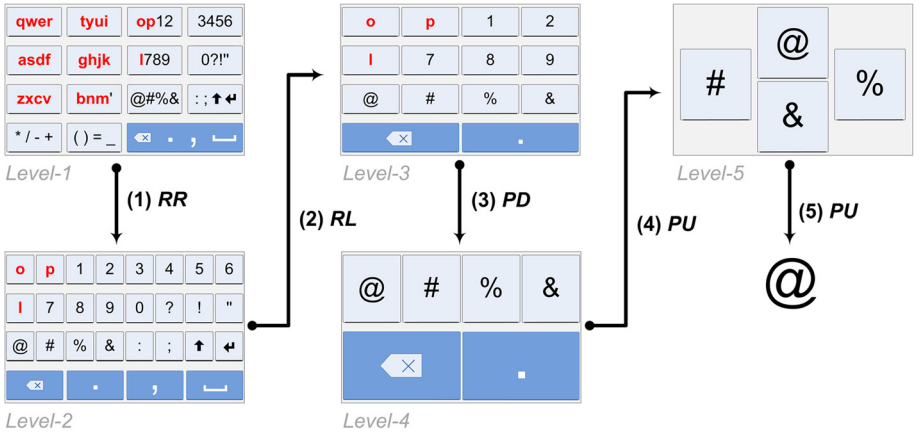

Figure 6 Entering character "@" with keyboard bisection method. In the final level, characters are positioned according to assigned tilt directions.

digits and supplementary symbols such as math operators and brackets. While character selection is performed by tilting the "cursor" within the keyboard layout, input has to be confirmed by dwelling in the neutral position zone upon discrete-tilt execution (see Figure 7).

When the SC method is initially invoked, a cursor starting position is assigned to the character "e" considering the most frequently used letter in the English alphabet. The highly efficient text-entry process with the SC method assumes constantly finding the optimal tilt-based "path" between two subsequent characters. It must be noted that circular navigation is allowed, meaning that cursor switch between the first and the third character row, as well as between the first and the twelfth character column can be achieved by single discrete tilt. In Figure 8 we consider the tilt-based distance between two characters according to different possible navigation strategies.

Obviously, the time required to input a particular character (or to activate some control option such as *shift* or *enter*) will depend on the chosen tilt-based path between the cursor initial position and the location of the target symbol. Just like the KB method, the SC method also includes the same four shortcuts which can be alternatively selected using long tilts. There are scenarios in which long tilts will herein provide faster access to shortcut options in comparison with the basic navigational input scheme.

In order to identify the minimal number of discrete tilts required for a particular character selection with the SC method (N_{SC}), we first define the keyboard coordinate system (*column*, *row*) wherein the character "q" has the origin position (1, 1), while the *backspace* character the position (12, 3). In general, if the current cursor marks the character i at position (x_1, y_1), and character j needs to be entered at position (x_2, y_2), then the optimal distance between i and j consists of $D_{OPT(SC)}$ tilts:

$$D_{OPT(SC)}\left(i_{(x1,y1)}, j_{(x2,y2)}\right) = min\left\{\begin{array}{c} |x_2-x_1| + |y_2-y_1|, \\ |x_2-x_1| + (3-|y_2-y_1|), \\ (12-|x_2-x_1|) + |y_2-y_1|, \\ (12-|x_2-x_1|) + (3-|y_2-y_1|) \end{array}\right\}$$

In the mathematical expressions above, the difference $(x_2\text{-}x_1)$ corresponds to *Roll Right* and *Roll Left*, while $(y_2\text{-}y_1)$ applies for *Pitch Up* and *Pitch Down*. Finding $D_{OPT(SC)}$ actually resolves the question if circular navigation should be used for accessing the given character j. In addition, the case with $D_{OPT(SC)} = 0$ has to be tackled in a specific way. Zero distance appears when the same character needs to be entered twice in a row, i.e. when $(i = j)$ holds. Repeating the previously entered letter requires exactly two discrete tilts before dwelling – the first one for moving the cursor to the random adjacent character, and the second one to bring the cursor back at the primal position. Hence, when zero distance is obtained, we have to make a $0 \rightarrow 2$ substitution for $D_{OPT(SC)}$. According to the aforementioned, it is now possible to define N_{SC} in line with the related cursor path between characters i and j:

$$N_{SC}(i,j) = \begin{cases} D_{OPT(SC)}(i,\ j), & (i \neq j) \wedge (j \in \mathbb{A} \cup \mathbb{S})(\textit{regular tilts}) \\ 2, & (i = j) \wedge (j \in \mathbb{A} \cup \mathbb{S})(\textit{regular tilts}) \\ 1, & j \in \mathbb{S}(\textit{long tilts}) \end{cases}$$

While $\mathbb{S}$ refers to the shortcut set which contains four characters/actions accessible both with regular and long tilts, $\mathbb{A}$ represents the set of all other available characters from the proposed keyboard layout that can be selected with regular tilts only. The N_{SC} values are in the range from 1 to 7.

Quad cursor method

With the introduction of the *quad cursor method* (QC), we try to provide faster access to the characters in the keyboard layout proposed within the SC method. While keyboard configuration keeps the same 12×3 scheme, the cursor principle is modified so as to enable marking a group of four adjacent characters in the same row (Figure 9).

Moving the quad cursor by means of discrete tilting will result in highlighting the new character quadruple according to the respective input primitive. In essence, the keyboard layout is virtually divided into nine quad-based zones which are accessible by corresponding cursor "movements". Circular navigation is allowed, as well as is shortcut activation using long tilts, just as within the SC method. QC-based character input always requires a sequence of three actions: (i) moving the cursor to the quadruple that contains a given letter/symbol, (ii) "activating"

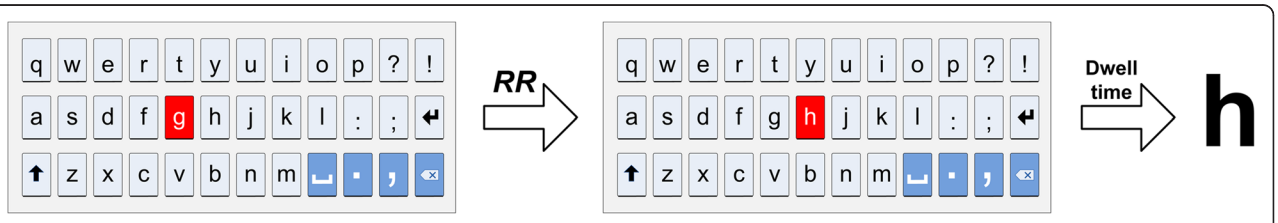

Figure 7 Entering character "h" using discrete tilt with single cursor method, assuming that cursor (specially visualized character) initially marks letter "g".

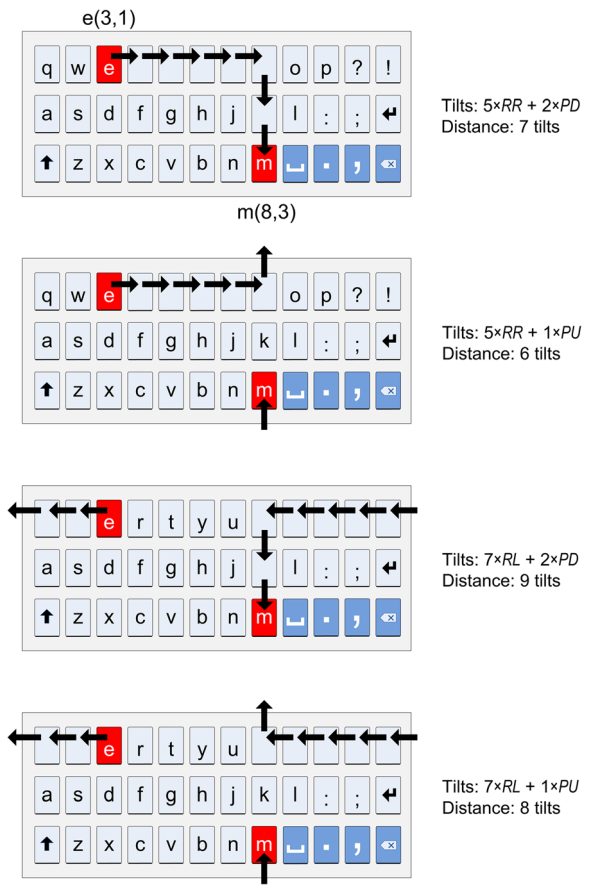

Figure 8 Distance between letters "e" and "m" from the discrete-tilt-based standpoint. In the presented case, four basic navigation strategies (right/down, right/up, left/down, and left/up) involve four tilt-based paths with different distances.

the respective quadruple by dwelling (in the neutral position zone), and (iii) making a final choice among four characters by single discrete tilts. The proposed text entry method is visualized in Figure 10.

In comparison with the SC method, quad cursor navigation allows faster positioning within the keyboard layout, but in addition requires one extra tilt for the final 4-letter disambiguation. This letter resolving is performed exactly the same way as in the final level within the KB method.

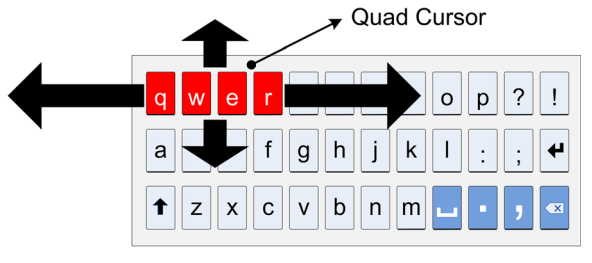

Figure 9 As opposed to the single cursor which always marks exactly one character, the quad cursor includes four horizontally aligned adjacent characters. The cursor initial position is assigned to the "q-w-e-r" quadruple.

For the QC method, we define the keyboard coordinate system as a 3×3 grid (in line with the quad-based zones), such that the quadruple "q-w-e-r" is located at position (1, 1), whereas position (3, 3) holds the shortcuts quadruple (*space-period-comma-backspace*). In other words, every four characters are assigned to the same position coordinates (e.g. letters "v", "b", "n", and "m" are all located at the position (2, 3)). The optimal distance between the character i at position (x_1, y_1), and character j at position (x_2, y_2), is therefore:

$$D_{OPT(QC)}\left(i_{(x1,y1)}, j_{(x2,y2)}\right) = min\left\{\begin{array}{c} |x_2-x_1| + |y_2-y_1|, \\ |x_2-x_1| + (3-|y_2-y_1|), \\ (3-|x_2-x_1|) + |y_2-y_1|, \\ (3-|x_2-x_1|) + (3-|y_2-y_1|) \end{array}\right\}$$

All considerations for the case where $D_{OPT(QC)}$ represents zero distance are identical as those within the SC method analysis (assuming $0 \rightarrow 2$ substitution). The minimal number of discrete tilts required for a particular character selection with the QC method is then:

$$N_{QC}(i,j) = \begin{cases} D_{OPT(QC)}(i,\, j) + 1, & (i \neq j) \wedge (j \in \mathbb{A} \cup \mathbb{S})(\textit{regular tilts}) \\ 2 + 1, & (i = j) \wedge (j \in \mathbb{A} \cup \mathbb{S})(\textit{regular tilts}) \\ 1, & j \in \mathbb{S}(\textit{long tilts}) \end{cases}$$

In the expression above, the +1 term refers to the extra tilt required for final 4-letter disambiguation. Definitions for both $\mathbb{S}$ and $\mathbb{A}$ sets remain the same. Since $D_{OPT(QC)}$ corresponds to the cursor navigation path whose tilt number can range between 1 and 2, the N_{QC} values are in the range from 1 to 3. Obviously, $N_{QC} = 1$ is valid for the long tilts usage only.

Modeling upper-bound text entry speeds

The theoretical efficiency of a particular text entry method can be predicted on the grounds of behavioral description of the simple text entry task. Soukoreff and MacKenzie [13] provided an exemplary quantitative prediction technique which encompasses two components: a movement model, and a linguistic model. While the movement model aims to predict the total time CT_{ij} required to enter character j preceded by previously entered character i, the linguistic model uses letter-pair (digraph) frequencies in a given language, resulting with calculated probabilities of occurrence P_{ij} for each digraph. Two respective models have to be combined in order to develop the prediction for an average character entry time CT_L in a given language, with the available

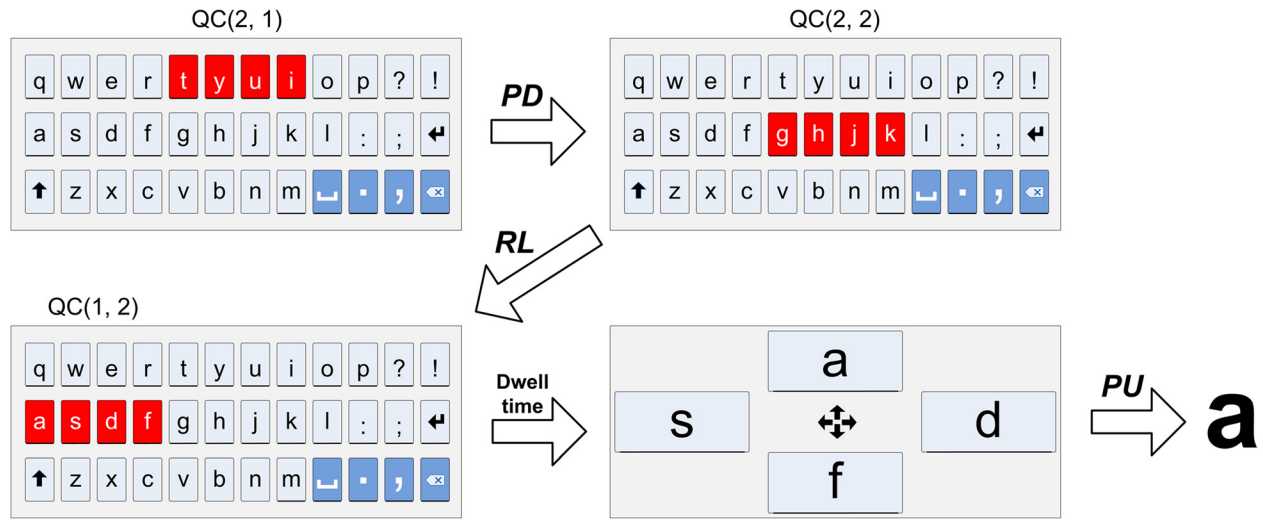

Figure 10 Entering character "a" using discrete tilts with the quad cursor method, assuming that cursor (specially visualized quadruple) initially marks the letter group "t-y-u-i".

character set $\mathbb{C}$ used in the respective text entry method:

$$CT_L = \sum_{i \in \mathbb{C}} \sum_{j \in \mathbb{C}} (P_{ij} \times CT_{ij})$$

The reciprocal of CT_L yields the average number of characters per second, which can easily be converted into the well-known text entry metric *WPM* (words per minute):

$$WPM_{max} = \frac{1}{CT_L} \times \frac{60}{5}$$

The transformation above is based on the usual definition from the typing measurement domain which determinates a word to be five characters long, including spaces and punctuation. The final metric represents the theoretical upper-bound text entry rate (therefore the suffix *max*), because the predictive model relies exclusively on the time to input characters, and assumes no additional time cost for mental activities such as thinking or visual searching.

In their initial work with text entry modeling [13], Soukoreff and MacKenzie used Fitts' law in order to predict the movement time between keys on a stylus-operated soft keyboard. The same approach has later been successfully applied for predicting text entry rates on a 12-key mobile phone physical keypad [14], as well as on several variations of soft keyboard layouts [15]. A model of two-thumb text entry on miniature QWERTY keyboards was also proposed using the same principle [16]. Beside Fitts' law for the movement model, the same linguistic model was used in the whole aforementioned research, basing on the digraph-frequency list with 27×27 = 729 digraphs, which is constructed for the English language with a limited character set consisting of 26 letters (a-z) and the *space* character. Hence, text entry methods are modeled without punctuation marks. Adapting the linguistic model to another language requires changing the digraph probabilities according to that particular language. Consequently, the existing prediction technique is not culture-specific, as it can even be used with non-Latin-based alphabets. Proof of concepts can be found in works related to predictive evaluation of Korean text entry [17] and Chinese text entry [18] on a multitap-based mobile phone keypad with 12 buttons.

In our proposed text entry solutions, Fitts' law cannot be used for the movement model because it does not apply for a discrete-tilt-based interaction. Instead, we have to model the total time required to enter a character by taking into account: (i) the minimal number of discrete tilts required for character selection, (ii) the average time for a single discrete-tilt execution, and (iii) the dwelling time, which applies to cursor-based text entry methods only (SC, QC). In order to find the discrete-tilt execution time, we carried out a user testing experiment. Furthermore, we calculated new digraph probabilities for the English language, according to the available corpora, so as to include two additional punctuation marks (*period* and *comma*).

Experimental evaluation of discrete-tilt execution time

In order to analyze the effects of both tilt type (*Roll*, *Pitch*) and interaction style (single-handed tilting in portrait orientation, and two-handed tilting in landscape orientation) on discrete-tilt execution time, a user testing experiment has been carried on. For testing purposes, a simple Android application for discrete-tilt execution data

gathering was developed. The application internally stores measurement results – time of discrete-tilt execution and achieved tilt angle, along with the information about user ID and utilized interaction style. Discrete-tilt parameters – angle offset for the neutral position zone, and threshold angles for both *Roll* and *Pitch* movements, can be assigned within the configuration activity. On the other hand, the user interface of the testing activity contains nothing more than a picture of an arrow, indicating the given direction of the tilt within the particular task instance (Figure 11).

In addition, if the smartphone device is not held in the neutral position zone at the beginning of the task, a special warning note ("*Please align*") becomes visible on the screen.

Participants

Twenty eight users, mostly students, were involved in the experiment (24 males, 4 females), their age ranging from 21 to 34 with an average of 23.5 years. Every participant had previous experience in operating touch-screen smartphones equipped with motion sensors, with the average experience of all users being 20.5 months. Sixteen users declared their preference for single-handed usage, in comparison with twelve that opted for a two-handed smartphone handling. Only two participants were left-handed.

Apparatus

A *Samsung Galaxy S II* (model I9100) was used as a smartphone platform for the testing application. This device comes with a 4.3″ display screen, weights 116 grams, and runs under Android OS version 2.3.4 (*Gingerbread*). Data from both the accelerometer and the magnetometer sensor were used for software-based determination of the device orientation. The sensor data sampling rate was set to the option *SENSOR_DELAY_FASTEST* in order to enable the most precise available measurement. For the same reason, *SystemClock.elapsedRealtime()* was used for time measurement in the application, as the respective clock is guaranteed to be monotonic, tolerant to power saving modes, and the recommend basis for general purpose interval timing [19]. Finally, all network-based services on the smartphone were turned off during the experiment.

Procedure

At the beginning of the test, participants were involved in a short practice session (about 10 minutes) in order to familiarize with both the smartphone device and testing application features. Within the practice session they were able to consider the assigned angle thresholds (45° for *Roll*, 30° for *Pitch*) and the neutral position zone (±10° angle offset), and thus to experience the difference between valid and invalid discrete-tilt movements. In actual testing, they were instructed to perform given tasks by executing discrete tilts "quickly, but in a natural way, without awkward wrist motions".

Since two factors, each with two levels, were considered within the experiment, four test conditions were altogether assigned: (i) *Roll* tilting with the single-handed usage in portrait mode, (ii) *Pitch* tilting with the single-handed usage in portrait mode, (iii) *Roll* tilting with two hands in landscape mode, and (iv) *Pitch* tilting with two hands in landscape mode. Each participant completed 24 discrete tilts in total: 6 tasks per distinct test condition. Both *Roll* and *Pitch* tasks uniformly addressed two possible tilt directions, hence there was an equal number of *RL*, *RR*, *PU*, and *PD* tasks. Obviously, a repeated measures design was utilized, thus a possible learning effect had to be properly compensated. For that purpose we applied counterbalancing, by dividing participants into four groups with different order of test conditions according to the 4×4 balanced Latin square design [20].

All tasks in the laboratory environment could be accomplished while sitting or standing, hence each participant had to choose a respective position regarding her/his own preference. Discrete-tilt execution time was evaluated using data obtained from the CSV file generated by the testing application. Figure 12 shows discrete-tilt task execution using a smartphone with the smaller form factor.

Results

Participants completed 672 discrete-tilt tasks in total, all of which produced valid input primitives. There was no

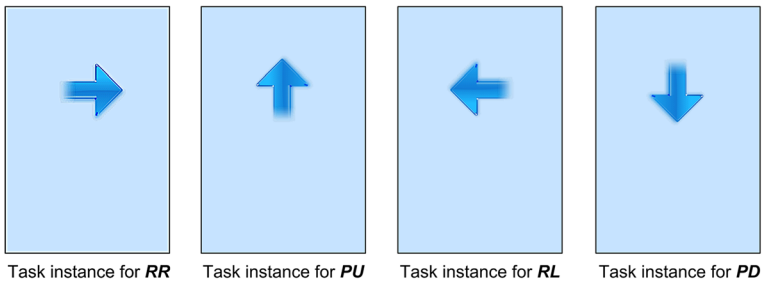

Figure 11 Tasks are in the testing application defined by the given direction of the discrete tilt that has to be executed.

Figure 12 Checking the testing application on *Samsung Galaxy Mini 2* (S6500D), which is the smartphone used as the development platform.

report of difficulties experienced in performing interaction whatsoever. That being said, we can justify the selected values for threshold angles, and assume discrete-tilt to be easy to learn. Mean values and standard deviations for obtained discrete-tilt times are summarized in Table 1. In addition, angles achieved are also presented.

To analyze the obtained data, a 2×2 repeated measures ANOVA was performed, with *Tilt Type* and *Interaction Style* being the within-subjects factors. The mean time difference between *Pitch* tasks completion (425.56 ms) and *Roll* tasks completion (427.07 ms) was not statistically significant ($F_{1,27}$ = 0.024, ns). Discrete tilt executed by using one hand in the portrait orientation lasted 424.70 ms on the average, as opposed to 427.92 ms for two-handed interaction in the landscape orientation. However, the respective difference was also not statistically significant ($F_{1,27}$ = 0.089, ns). Finally, the interaction effect *Tilt Type* × *Interaction Style* showed to be non-significant as well ($F_{1,27}$ = 0.351, ns).

Since there were no statistically significant main effects found on discrete-tilt execution time, claiming different times for observed conditions would not be justified. Therefore, it was decided to assign a single time value for discrete-tilt duration. We used the grand mean for task completion time (426.31 ms) and, in order to simplify further calculations, rounding to the hundredth. In conclusion, for single-valued discrete-tilt execution time we suggest: T_{tilt} = 0.43 s.

Movement models for discrete-tilt-based text entry methods

As already mentioned afore, a movement model for a particular text entry method predicts the total time CT_{ij} required to enter character j preceded by a previously entered character i. Since we already defined the minimal number of discrete tilts required for a particular character selection, as well as the discrete-tilt execution time, we can now provide movement models for the proposed text entry methods.

For the keyboard bisection method, we have:

$$CT_{ij}(KB) = N_{KB}(i,j) \times T_{tilt}$$

Since the KB method relies on tilt-only interaction, with no dwelling whatsoever, the total time to enter a particular character can be calculated as a simple sum of all discrete-tilt durations. Specifically, if N_{KB} values are used, we obtain:

$$CT_{ij}(KB) = \begin{cases} 2.15\,s, & j \in \mathbb{A}(regular\ tilts) \\ 1.72\,s, & j \in \mathbb{S}(regular\ tilts) \\ 2.43\,s, & j \in \mathbb{S}(long\ tilts) \end{cases}$$

The KB is the only method among the proposed ones in which the time to input a particular character does not depend on the previously entered one. Actually, every character, if not from the shortcut set, requires exactly five regular discrete tilts, i.e. 5×0.43 = 2.15 seconds. Of course, this holds for expert behavior, assuming no time is wasted for finding the desired character, or thinking about bisection strategy. Another characteristic of the expertise is performing long discrete tilts of duration of exactly 2.43 seconds (0.43 s for discrete-tilt movements and precisely 2 s for dwelling). However, this is extremely difficult, since a user will usually dwell in a valid tilt zone for a little longer. Shortcut activation times seem strange at first. Namely, the model predicts faster shortcut activation using regular tilts (4×0.43 = 1.72 s) in comparison with the single long tilt (2.43 s), thus the shortcut concept can be questioned. At this point we state that shortcuts do not always provide time gain, but are extremely valuable in lowering the interaction burden. Doing four regular tilts

Table 1 Results: descriptive statistics summary for time and angle of discrete-tilt execution

Tilt type	*Interaction style*	Discrete-tilt time T_{tilt} [ms]	Angle achieved [°]
Pitch	Single-handed, portrait	426.56 ± 105.22	62.20 ± 6.98
Pitch	Two hands, landscape	424.54 ± 106.80	55.53 ± 3.86
Roll	Single-handed, portrait	422.84 ± 115.06	60.94 ± 6.88
Roll	Two hands, landscape	431.29 ± 102.16	53.60 ± 4.29

instead of one long tilt will certainly increase the fatigue level.

We now provide the movement model for the single cursor method:

$$CT_{ij}(SC) = N_{SC}(i,j) \times T_{tilt} + T_{dwell}$$

Along with the total time required for tilt-based navigation to a certain character (cursor shifting along the optimal-distance path), dwelling time T_{dwell} has to be included in the equation as well. It should be noted that dwelling actually results with input of the previously selected character in the SC method. T_{dwell} (always applied at the neutral position zone) should not be mistaken for 2s dwelling threshold that differentiates regular and long tilts. After substituting N_{SC}, we get the following formula:

$$CT_{ij}(SC) = \begin{cases} D_{OPT(SC)}(i,j) \times 0.43 + T_{dwell}, & (i \neq j) \wedge (j \in \mathbb{A} \cup \mathbb{S})(\textit{regular tilts}) \\ 0.86 + T_{dwell}, & (i = j) \wedge (j \in \mathbb{A} \cup \mathbb{S})(\textit{regular tilts}) \\ 2.43s, & j \in \mathbb{S}(\textit{long tilts}) \end{cases}$$

Hence, the total time for character input in this case depends on the tilt-based distance between a previously entered character and the target character. If the same character needs to be entered consecutively, two tilts are required (2×0.43 = 0.86 s) before dwelling in the neutral position zone. Shortcut activations using long tilts do not involve T_{dwell}, since the related input primitives (*2sRL*, *2sRR*, *2sPU*, and *2sPD*) are directly assigned to the one of four available options.

The movement model for the quad cursor method is analogous with the SC movement model:

$$CT_{ij}(QC) = N_{QC}(i,j) \times T_{tilt} + T_{dwell}$$

The only difference is in the role of T_{dwell}: dwelling in the QC method does not trigger character entry like in the SC, but selects the quadruple for final 4-character disambiguation instead. Making a final choice among the four characters is done by regular discrete tilt which is included in N_{QC}. Inserting N_{QC} in the movement model equation gives the following formula:

$$CT_{ij}(QC) = \begin{cases} D_{OPT(QC)}(i,j) \times 0.43 + 0.43 + T_{dwell}, & (i \neq j) \wedge (j \in \mathbb{A} \cup \mathbb{S})(\textit{regular tilts}) \\ 1.29 + T_{dwell}, & (i = j) \wedge (j \in \mathbb{A} \cup \mathbb{S})(\textit{regular tilts}) \\ 2.43s, & j \in \mathbb{S}(\textit{long tilts}) \end{cases}$$

Just like in the SC movement model, when long tilts are used, T_{dwell} is not accounted for. Repeating an already entered character ($i = j$) assumes 3 regular tilts altogether (3×0.43 = 1.29 s): two tilts before dwelling – for selecting the proper quadruple, and one tilt after dwelling – for 4-character disambiguation.

Linguistic model

A new linguistic model for the English language has been constructed since we use a specific character set consisting of exactly 29 elements, including the letters a-z (lowercase), the *space* character, and two punctuation characters (*period*, and *comma*). The model therefore provides a 29×29 matrix of digraph probabilities P_{ij} (probabilities that the character i is followed by the character j).

For the computation of the P_{ij} matrix values, we used the compiled English text corpus containing more than 969 thousands of sentences. The corpus was assembled using three different sources: (i) *SETimes* – a parallel corpus of English and south-east European languages, based on the content published on the SETimes.com news portal, (ii) *hrenWaC* – a Croatian-English parallel web corpus, and (iii) *TED talks parallel corpus* – a collection of parallel sentences extracted from the Croatian-English TED talks transcripts. All of the said corpora are published under the CC-BY-SA or CC-BY-NC-SA-3.0 license, and were obtained from the website of the Natural Language Processing group [21] which operates within the Faculty of Humanities and Social Sciences at the University of Zagreb, Croatia. Only English parts of available corpora were used, all the text was transposed to lowercase, and only the said 29 characters were considered valid for the language statistics computation. In other words, digraphs with characters not defined in our 29-character set were not included in P_{ij} calculation.

In this way we obtained exactly 841 values representing calculated probabilities of occurrence for each digraph. Table 2 shows the statistics for ten most-frequent digraphs in our compiled corpus.

Eight digraphs from the presented Table 2 are listed in the similar top-ten table provided in [13] which was constructed on the grounds of the 27×27 matrix (English character set with *space* and without any punctuation mark) and a much smaller corpus with digraph count of 107,199. Although the respective probability values do not exactly match, we believe that our linguistic model fairly characterizes everyday English.

Table 2 Ten most frequent digraphs from the used English text corpus (total number of digraphs: 119,157,501)

Digraph	Probability
e-*space*	0.027159927
space-t	0.023404754
t-h	0.020670600
s-*space*	0.018853811
space-a	0.017752403
h-e	0.016824732
i-n	0.016397423
n-*space*	0.014970799
a-n	0.014045259
d-*space*	0.013646426

Speed predictions for discrete-tilt-based text entry

At this point, in order to develop upper-bound text entry speed predictions, we combine the tilt-based movement models $CT_{ij}(KB)$, $CT_{ij}(SC)$, and $CT_{ij}(QC)$ with the linguistic model P_{ij}:

$$WPM_{max} = \frac{1}{\sum_{i\in\mathbb{C}}\sum_{j\in\mathbb{C}}\left(P_{ij}\times CT_{ij}\right)}\times\frac{60}{5}$$

All equations and parameters needed were previously defined, with the exception of T_{dwell}. The power of predictive modeling now becomes even more apparent: we could easily obtain several input rate predictions for different T_{dwell} values, without the need to test a text entry method with real users. We could furthermore observe the effect of dwelling time on upper-bound text entry speed predictions. The value T_{dwell} = 1.2 s will be used in further calculations, as initial tests in the implementation phase showed the respective time to be the suitable pause between sequences of active tilting.

When computing time predictions, we assume that two input strategies can be applied for every proposed discrete-tilt-based text entry method: (i) shortcut usage allowed: if the target character resides in the shortcut set, it will be entered with a single long tilt, and (ii) shortcut usage not allowed: all characters have to be entered using regular discrete tilts only. In such a way we are able to provide two predictions for the same text entry method, and to accordingly analyze the effects of long tilts.

A simple computational program was developed for predictions calculation. For every possible *i-j* digraph ($i\in\mathbb{C}, j\in\mathbb{C}, \mathbb{C} = \mathbb{A}\cup\mathbb{S}$), it calculates CT_{ij} according to the given input strategy and formulas presented for the particular method. For SC/QC methods, it first resolves the optimal tilt-based distance between characters *i* and *j*, and performs a 0→2 substitution when necessary. Once CT_{ij} values are obtained, the program combines (by multiplying) them with the respective digraph probabilities from the available 29×29 matrix, thus providing the final WPM_{max} metric. Table 3 shows the results summary.

Discussion

The obtained predictions, totaling approximately 5 WPM on the average, indicate rather low text entry rates for all discrete-tilt-based input methods. Such results are expected and understandable as tilt-based text entry cannot compete with standard touch-based texting. Nevertheless, tilt-to-text concept could provide certain benefits in situations when typing on small touchscreens becomes particularly inconvenient, and especially when users' visual contact with the device display is obstructed. Indeed, the discrete-tilt-based interaction provides the possibility for "blind typing" using the proposed methods, once both keyboard layouts and bisection patterns have been memorized.

Table 3 Upper-bound text entry speed predictions for proposed discrete-tilt-based input methods

Text entry method	WPM_{max}	
	Long tilts not enabled	Long tilts enabled
Keyboard Bisection (KB)	5.79	5.46
Single Cursor (SC)	4.21	4.42
Quad Cursor (QC)	5.16	5.15

The KB method is predicted to be the most efficient. According to the results, when long tilts are not enabled, it can produce 3.15 characters per minute more than QC, and 7.9 characters per minute more than SC. Taking into account the number of discrete tilts required for a single character input, these differences seems important. On the other hand, text entry with the KB method is supposed to be the most burdensome, since both physical and mental activities at a higher level are expected. Unlike with SC and QC, there is no dwelling time which could be used for "wrist resting", while at the same time the layout is constantly changing, hence imposing a considerable cognitive load.

As opposed to the KB method, cursor-based methods are more straightforward and include a consistent full QWERTY layout, hence better learnability is assumed. However, when it comes to efficiency, the predicted input rates show the lowest WPM values for the SC method. The reason for that are sometimes very lengthy tilt-based paths between two characters that have to be covered by cursor switching. In the worst-case scenario altogether seven discrete tilts have to be activated in order to select a target character. Interestingly, the most frequent English digraph e-*space* corresponds to a 7-tilts distance in the proposed keyboard layout. This can be a motive to find a "better place" for the *space* character, and to propose a possibly more efficient design. The QC method involves much shorter tilt-based distances, thus better efficiency predictions in comparison with the SC are no surprise. According to the trade-off between the obtained predictions and the expected ease of use, QC seems to be the most convenient method to use; however, this has to be further investigated.

The effect of long tilt on text entry speed depends on the observed input method. Results show that using long discrete tilts: (i) decreases text entry speed within the KB method, (ii) enhances input efficiency within the SC method, and (iii) has no particular impact if the QC method is applied. Respective differences in predicted text entry speeds are almost negligible, resulting with limited variations in the respective characters per minute metric (0.05-1.65). On the other hand, long tilts are especially worthwhile in lowering the interaction burden,

hence using them in all proposed text entry methods appears to be a good choice.

All predicted values hold for the highest possible level of discrete-tilt-based text entry expertise. Text input without making any errors is assumed. Cognitive activities such as visual search, thinking, and decision making are not included in the described predictive modeling, hence upper-bound entry rates are addressed only. It can be observed that it is very hard, if not impossible, to achieve such proficiency, especially with completely new text entry methods. A considerable learning time is required to become familiarized with the bisection principle and the corresponding layout (KB), as well as to gain experience in finding the optimal tilt-based path between two characters or quadruples (SC, QC). Novice users are expected to use the introduced input methods with noticeably lower text entry speeds with respect to the ones obtained by model predictions. However, predictive modeling is undoubtedly valuable in providing benchmarks and possibilities to both analyze and discuss designs prototypes in advance.

Although we consider both our movement and linguistic models to be good estimates for deriving expert-level text entry rate predictions, they nevertheless show some limitations which are discussed in the following. The movement model relies on the user's motor characteristics, i.e. the time needed to complete the discrete-tilt action. This parameter was empirically determined within the respective experiment in which a single smartphone device was used for testing. Since different smartphone manufacturers do not necessarily embed the same motion sensors in their device models, a further study should investigate if discrete-tilt time is in fact device-independent. The testing application used in this work can be applied for evaluating discrete-tilt handling on various smartphones under the Android OS. When it comes to the linguistic model, it should be noted that it is limited to a 29×29 digraph matrix, and does not include uppercase letters, numbers, or supplementary symbols. However, unlike some previous work in this domain, the punctuation marks *period* and *comma* exist in the model. While numbers and supplementary symbols (such as ampersand, asterisk, backslash, etc.) are less frequent in the language corpus, uppercase letters are more common, hence their involvement could improve the accuracy of the linguistic model. Adding new 26 letters to the available character set would require a new calculation of digraph probabilities in a 55×55 matrix. The movement model would have to be modified then as well, with minimal number of tilts being calculated on the ground of alternating keyboard layouts (between lowercase and uppercase, via the *shift* function).

Conclusion and future work

We have introduced the discrete-tilt concept for text entry, which is a special case of tilt-based interaction wherein the input primitive is triggered after a well-defined sequence of *Pitch* or *Roll* movements of the mobile device. As opposed to solutions that use continuous-tilt approach, discrete-tilt-based interaction in great deal does not rely on visual feedback. Altogether eight input primitives have been proposed and applied for text entry.

Three different methods for discrete-tilt-based text entry have been designed, including respective QWERTY-like keyboard layouts and appropriate strategies for character selection/input. Long discrete tilts, intended for shortcut activation, can be concurrently used with regular tilts in all of the proposed methods. Theoretical predictions for expert text entry rates are provided, according to the combination of movement and linguistic models. The movement model corresponds to both the time of discrete-tilt execution (which has been experimentally determined), and the minimal number of tilts required for target character input (which has been calculated for every particular method). The linguistic model, comprising digraph statistics, has been built basing on available English corpora. Obtained predictions revealed different upper-bound text entry speeds for proposed discrete-tilt-based input methods. The KB method showed to be the fastest one, with a more prominent difference when compared with the low-efficient SC method. According to the results, long tilts generally have no major effect on input efficiency. In conclusion, predictive modeling has been successfully applied to discrete-tilt-based text entry, resulting with upper-bound efficiency estimates that can be used as a baseline for similar designs relying on tilt interaction.

Our future work plan includes several directions. Most importantly, the proposed text entry method prototypes necessitate formal usability testing. We expect to get a detailed insight into the correlation between theoretical predictions and empirically obtained text entry rates. In addition to efficiency, other usability attributes should be carefully observed as well. E.g., learnability, errors, and satisfaction play important role for overall acceptance of newly proposed methods. Required physical and mental efforts have to be tackled in order to get a better understanding of ease-of-use constraints.

Further work needs to be done in order to investigate the discrete-tilt-based text entry in a real-life mobile context, with simulation of walking scenarios, attention shifts, and external distractions. We hope that "blind typing" could be helpful in some of these circumstances. In addition, different form factors of mobile devices should be encompassed in the future studies, here including both tablet and phablet classes.

Finally, we find the idea of combining discrete-tilt and continuous-tilt within cursor-based text entry methods an idea worth to consider. Such an approach would assume continuous input primitive invocations once the

device leaves the neutral position zone, but with distinct audio and/or tactile feedback for every single invocation. Consequently, the cursor could be continuously moved in a particular direction, with the corresponding interaction still less relying on any visual feedback. The implementation of the proposed design is already underway.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
SL modeled the proposed text entry methods, programmed the respective prototypes, calculated digraph probabilities, organized the user testing experiment, and wrote a preliminary version of the paper. VG has formed the main research idea and has provided substantial guidance for the development of the models. MK participated in the design of the user testing experiment and helped with statistical analysis. All authors read and approved the final manuscript.

Acknowledgements
This paper describes the results of research being carried out within the project 036-0361994-1995 Universal Middleware Platform for e-Learning Systems, as well as within the program 036-1994 Intelligent Support to Omnipresence of e-Learning Systems, both funded by the Ministry of Science, Education and Sports of the Republic of Croatia.

Author details
[1]Faculty of Engineering, University of Rijeka, Vukovarska 58, HR-51000 Rijeka, Croatia. [2]Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, HR-10000 Zagreb, Croatia. [3]Medimurje University of Applied Sciences in Cakovec, Bana Josipa Jelacica 22a, HR-40000 Cakovec, Croatia.

Received: 23 July 2014 Accepted: 18 September 2014
Published online: 30 September 2014

References

1. Findlater, L, & Wobbrock, JO. (2012). From plastic to pixels: in search of touch-typing touchscreen keyboards. *ACM Interactions, 19*(3), 44–49.
2. Zhai, S, Kristensson, PO, Gong, PF, Greiner, M, Peng, SA, Liu, LM, & Dunnigan, A. (2009). *ShapeWriter on the iPhone – From the Laboratory to the Real World. In Proc. CHI EA 2009 (pp. 2667–2670)*. New York: ACM.
3. Swype. http://www.swype.com. Accessed 12 Jul 2014.
4. Sazawal, V, Want, R, & Borriello, G. (2002). *The Unigesture Approach. In Proc. Mobile HCI 2002 (pp. 256–270)*. New York: ACM.
5. Partridge, K, Chatterjee, S, Sazawal, V, Borriello, G, & Want, R. (2002). *TiltType: Accelerometer-Supported Text Entry for Very Small Devices. In Proc. UIST 2002 (pp. 201–204)*. New York: ACM.
6. Wigdor, D, & Balakrishnan, R. (2003). *TiltText: Using Tilt for Text Input to Mobile Phones. In Proc. UIST 2003 (pp. 81–90)*. New York: ACM.
7. Jones, E, Alexander, J, Andreou, A, Irani, P, & Subramanian, S. (2010). *GesText: Accelerometer-Based Gestural Text-Entry Systems. In Proc. CHI 2010 (pp. 2173–2182)*. New York: ACM.
8. Goel, M, Findlater, L, & Wobbrock, J. (2012). *WalkType: Using Accelerometer Data to Accommodate Situational Impairments in Mobile Touch Screen Text Entry. In Proc. CHI 2012 (pp. 2687–2696)*. New York: ACM.
9. Inference Group: The Dasher Project. http://www.inference.phy.cam.ac.uk/dasher/. Accessed 12 Jul 2014.
10. Fitton, D, MacKenzie, IS, Read, JC, & Horton, M. (2013). *Exploring Tilt-Based Text Input for Mobile Devices with Teenagers. In: Proc. HCI 2013*. London: British Computer Society.
11. Ljubic, S, Kukec, M, & Glavinic, V. (2013). *Tilt-Based Support for Multimodal Text Entry on Touchscreen Smartphones: Using Pitch and Roll. In Proc. HCI International 2013, UAHCI/HCII 2013, Part III, LNCS 8011 (pp. 651–660)*. Heidelberg: Springer.
12. Rahman, M, Gustafson, S, Irani, P, & Subramanian, S. (2009). *Tilt Techniques: Investigating the Dexterity of Wrist-based Input. In Proc. CHI 2009 (pp. 1943–1952)*. New York: ACM.
13. Soukoreff, RW, & MacKenzie, IS. (1995). Theoretical upper and lower bounds on typing speed using a stylus and soft keyboard. *Behaviour & Information Technology, 14*, 370–379.
14. Silfverberg, M, MacKenzie, IS, & Korhonen, P. (2000). *Predicting Text Entry Speed on Mobile Phones. In Proc. CHI 2000 (pp. 9–16)*. New York: ACM.
15. MacKenzie, IS, Zhang, SX, & Soukoreff, RW. (1999). Text entry using soft keyboards. *Behaviour & Information Technology, 18*(4), 235–244.
16. MacKenzie, IS, & Soukoreff, RW. (2002). *A Model of Two-Thumb Text Entry. In Proc. GI 2002 (pp. 117–124)*. Toronto: Canadian Information Processing Society.
17. Ilinkin, I, & Kim, S. (2008). *Design and Evaluation of Korean Text Entry Methods for Mobile Phones. In Proc. CHI EA 2008 (pp. 2853–2858)*. New York: ACM.
18. Liu, Y, & Raiha, K-J. (2010). *Predicting Chinese Text Entry Speeds on Mobile Phones. In Proc. CHI 2010 (pp. 2183–2192)*. New York: ACM.
19. Android Developers: SystemClock. http://developer.android.com/reference/android/os/SystemClock.html. Accessed 12 Jul 2014.
20. MacKenzie, IS. (2013). *Human-Computer Interaction: An Empirical Research Perspective*. Elsevier.
21. Natural Language Processing group: Corpora. http://nlp.ffzg.hr/resources/corpora/. Accessed 12 Jul 2014.

doi:10.1186/s40166-014-0003-6
Cite this article as: Ljubic *et al.*: **Predicting upper-bound text entry speeds for discrete-tilt-based input on smartphones.** *Journal of Interaction Science* 2014 **2**:3.