RingBoard 2.0 – A Dynamic Virtual Keyboard Using Smart Vision

Taylor Ripke, Eric O'Sullivan, and Tony Morelli(✉)

Central Michigan University, Mount Pleasant, MI 48859, USA
{ripkeltj, osullle, morella}@cmich.edu.com

Abstract. Computers have evolved throughout the digital era becoming more powerful, smaller, and cheaper. However, they are still lacking basic accessibility features that appeal to all users. They can be controlled with your voice and eye movement, but there is still much work to be done. This paper presents RingBoard 2.0, a dynamic virtual keyboard that uses computer vision to recognize and track hand movements and gestures. It allows for basic input to a computer using a web camera. This application was built to provide additional accessibility features for those who experience tremors or limited motor capability in their hands, which make it difficult to interact with a standard keyboard and mouse. At the core, it is built to recognize any form of a hand and can accurately track it, regardless of sporadic movement. This paper is an extension of previous work describing touch input for a computer using the HP Sprout [2].

Keywords: Accessibility · Vision · Touch · Keyboard · Tracking

1 Introduction

The QWERTY keyboard patented by Sholes in 1878 is the defining keyboard still used by many electronic devices, including smart phones and computers [1]. It provides a convenient way to quickly transcribe manuscripts into print. The advent of the digital era brought significant change to how information is conveyed and produced; however, it was rapidly leaving many individuals behind. Although revolutionary for its time, there is little information regarding additional accessibility features for those with cognitive or motor impairments. Those individuals face similar challenges today when interacting with modern technology, especially a keyboard and mouse which require precise motor control and concentration.

This paper introduces additional enhancements to RingBoard, an application designed by Wojcik et al. [2] as shown in Fig. 1. RingBoard was designed for use "with a personal computer such that a person with a mobility disability who cannot utilize a standard physical keyboard would be able to better interact with a standard computer" [2]. The updated version integrates advanced computer vision algorithms to track the user's hand while interacting with the system. These additions facilitate those who may have difficultly pressing on the mat with fingers due to limited motor control or loss of limb. The system can adapt to virtually any input, compensating for a limited range of motion or tremors. The paper will detail the algorithms and provide a comparison between different forms of input for the system.

© Springer Nature Switzerland AG 2019
M. Antona and C. Stephanidis (Eds.): HCII 2019, LNCS 11573, pp. 323–333, 2019.
https://doi.org/10.1007/978-3-030-23563-5_26

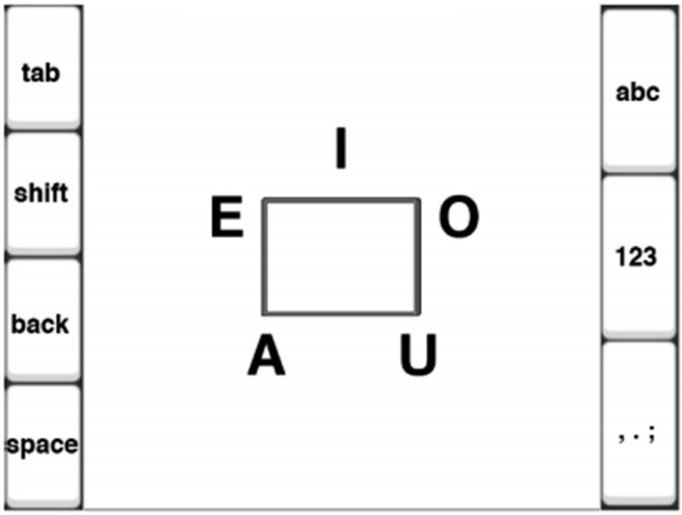

Fig. 1. RingBoard.

2 Related Work

Individuals with physical disabilities such as Cerebral Palsy, Spina Bifida, and tremors may have difficulty interacting with a traditional computer keyboard and mouse. These users may take longer to complete typing tasks and may experience various errors such as holding a key too long, pressing adjacent keys, missing keys, failing to hold two keys simultaneously, pressing the wrong key, or pressing more than one key at a time on accident (Trewin 1999). It is possible to provide accommodations for some of the errors by adjusting settings on the computer. Common solutions include "adjusting the repeat time of a key, looking for sticky keys, and using CAPS Lock [2].

As stated previously, users with limited mobile capabilities may find it difficult to interact with a standard keyboard. People who have limited capabilities in their hands may be able to use a computer keyboard keyguard, which is specifically designed for users with limited motor control. The design allows the user to increase typing accuracy and help stabilize fingers on the right keys [3]. Additional technologies include a no hands mouse controlled by foot movement and keyboard control through foot pedals, which allow the user to press important buttons such as shift, ctrl and alt [3].

AbleNet provides a variety of tools, such as switches, that can be used with a computer to provide users with the capability to interact with the system. Some popular switch devices include the Big Red, which is a large, red button users can press, and the Blue2 Bluetooth Switch, which gives users the capability to press two large buttons [4]. The devices can be used to work with existing applications but are also well-suited to be custom programmed to fit the user's needs.

Another possible solution to allow the user to interact with the system if they have limited control in their hands and arms is to use technology that tracks the position of the head relative to the screen. For example, the SmartNav 4: AT provides a hands-free

mouse solution that allows the user to control their computer using only head movements. This solution provides a way to provide mouse input, as well as keyboard input through the use of a virtual keyboard. It even supports switch or foot pedal input to provide additional input [5]. Earlier versions of similar systems exist, which include printing a copy of a keyboard onto a piece of paper and attaching a laser to the user's head. A computer vision algorithm interfaced with a web camera was used to determine where the laser was on the keyboard and translate it into corresponding text input [8]. The second version of RingBoard presented in this paper was motivated from this implementation.

However, if the user has very limited motor capabilities, another possible solution is Eyegaze Edge, which is a eye-operated communication and control system that uses advanced image processing techniques to track the position of the eye relative to the screen at 60 frames per second, providing accuracy up to a 1/4 inch or less [6]. Users are able to type on a keyboard and generate speech. To click a button, the user waits a specified amount of time, such as half a second [6].

Similar systems also use a graphical keyboard that "allows the user to change the size and location of virtual onscreen keyboard buttons" [9]. These onscreen keyboards have also been used with specification applications, not just customizable character input. For example, one solution was to populate an onscreen keyboard with commonly type phrases specific to the active application [10]. The underlying system can determine a user's commonly typed phrases and "prepopulate the keyboard with user specific common words and phrases" [7].

Although this research is directly applicable for computers, it may also be possible to apply these accessibility features to mobile computers with the addition of a Bluetooth touch keyboard. Previous work has shown that users are okay using an additional keyboard when interacting on a mobile device [8]. In the upcoming section discussing future work, additional accessibility features are being considered that take advantage of predictive text. For example, Gkoumas implemented a solution to predictive typing where the keyboard enlarges letters it thinks are likely to be next [9]. After some training to understand an individual's diction, it may be possible to implement a similar approach on RingBoard.

Furthermore, additional research concerning virtual keyboards has been done by considering the position of the hand [10]. Similarly, as described in this paper, RingBoard is actively involved in tracking the user's hand and using it as input to the keyboard. Rashid proposed an additional enhancement by showing the benefits of having the virtual keyboard be relative to where the user starts typing, rather than having the keys remain in the same location on mobile devices [11]. This idea was considered when implementing the first iteration of RingBoard by having the secondary keys appear in the direction of the user's hand movement [2].

3 Motivation

As previously described, the current iteration of RingBoard was motivated by previous work done in [2]. It is developed on an HP Sprout computer running Windows 10 utilizing a touch sensitive mat, touch screen, and overhead camera and projector.

The projector displays a virtual keyboard on the touch mat that the user can interact with using their fingers. In the original version, the best results were achieved using fingers for touch input rather than using a fist. Due to the inherit design of the HP Sprout, it is designed to ignore the fist when users are interacting with the touch mat. This is inconvenient for people interacting with the system who have limited precision of movement in their arms and hands.

The current iteration of RingBoard is designed to be more accessible for individuals with limited motor capabilities. To overcome the challenges of fist-based touch input, computer vision algorithms were implemented to track the user's hand. As discussed in the upcoming sections, this type of input provided a convenient solution applicable to all users, compensating for sporadic movement or limited hand functionality.

4 Using the Keyboard

Interacting with the visual keyboard is quite easy. When the application is launched, a black screen will appear for a few seconds while the system calibrates. During this time, the camera is being prepared and the background image is captured. It is essential that only the mat is present during these few seconds, as other objects, such as hands or supplies, can interfere with subsequent processing. Next, the application will wait for the user to put their hand into the frame. It will pause for five seconds, allowing the user to position their hand. The keyboard will then become active and the user can interact with the different keys and buttons.

As described in [2], the keyboard displays a set of primary keys as shown in Fig. 1. These primary keys compose the default layout of the system. To access additional letters, the user needs to move their hand between two letters to display a menu of secondary keys as shown in Fig. 2. To return to the previous menu, the user must either keep their hand stationary for five seconds or increase the surface area of their hand by opening it, the details of which are discussed in the following section.

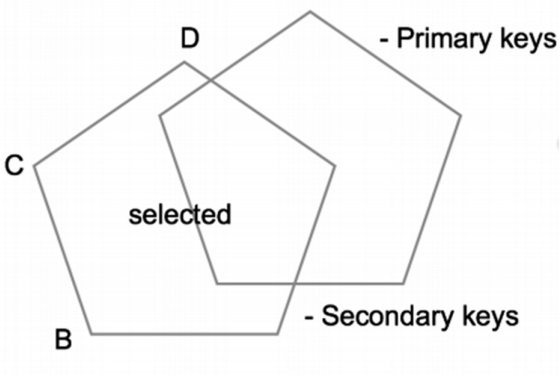

Fig. 2. Secondary keys.

Between each button press, a three-second delay is implemented to prevent the user from accidently pressing a button while moving their hand to a new location. For example, moving from the ring of letters to a button for punctuation may accidently trigger a letter, which is an inconvenience to the user. To reset the keyboard, the user can either keep their hand stationary for seven seconds or open their hand. The details of the implementations of both techniques are detailed in the following section.

5 Implementation

As stated previously, RingBoard provides two forms of input: touch and vision. This section will discuss the intricacies of how the application detects and tracks a user's hand or limb, while compensating for spurious muscle movement. The underlying vision algorithms are designed to be accessible to all individuals, as long as they can move their arm within view of the camera.

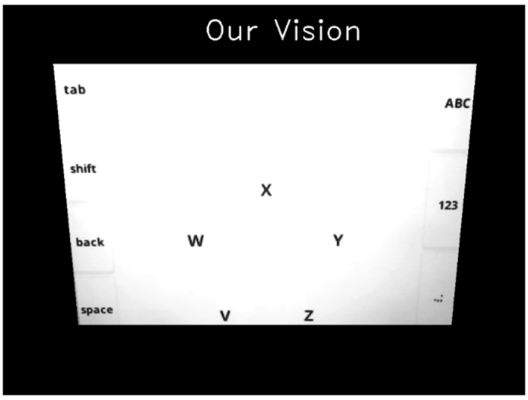

Fig. 3. Background image

The application was developed in Unity and programmed in C#. It is designed to run on the HP Sprout, although it can be modified to work on a system with multiple monitors and a downward facing camera. The Sprout has an interactive touch pad, which also serves as an additional screen, however the touch pad is not utilized and only acts as a background. Any relatively clear surface will work. The current version projects the display on the bottom screen, directly below the overhead camera. The Sprout's overhead camera can be accessed and controlled directly in Unity with the assistance of OpenCV. The OpenCV module provides an easy way to perform fundamental image processing tasks in Unity without having to run a separate program that interfaces with Unity through TCP.

$$I_n[x, y] = |I_c[x,]y - I_b[x, y]| > T \tag{1}$$

Eq. 1 Background Subtraction

The application begins by capturing an image of the touch pad without the user's hand present initially. This is done to calibrate the application for background subtraction. The image captured is taken at an angle due to the Sprout's design. A comparison can be seen by comparing Fig. 1 to Fig. 2. Furthermore, the image captures the sides of the touch mat, including the desk that it is sitting on. This interferes with image processing and is removed by isolating the interest area with black polygons as shown in Fig. 2.

During runtime, the application captures an image and compares it against the static background image to determine if an object is present. To reduce additional unwanted noise the images were converted to grey. Equation 1 depicts the equation used for background subtraction. The result of the background subtraction image is a binary image indicating areas where a change in pixel intensity was greater than the threshold T as shown in Fig. 3. I_n is the new image produced by taking the absolute value of the current image subtracted from the background image within a given threshold to produce a binary image.

As mentioned in the previous section, an important aspect of the keyboard is the ability to reset it after the user has made a selection. The user can either keep their hand stationary for five seconds or rapidly increase the surface area of their hand by opening it, assuming their hand is in the shape of a fist. For the former method, the application in constantly tracking and monitor the area of your hand. If after five seconds the area remains relatively constant within a predefined threshold, it will reset the keyboard bringing it back to the default screen. Similarly, if the user opens their hand suddenly, the application is also looking for a sudden change in area, which is also a predefined threshold. According to each user's individual needs, these thresholds can be modified in the settings menu of the keyboard as sensitivity levels may vary from person to person.

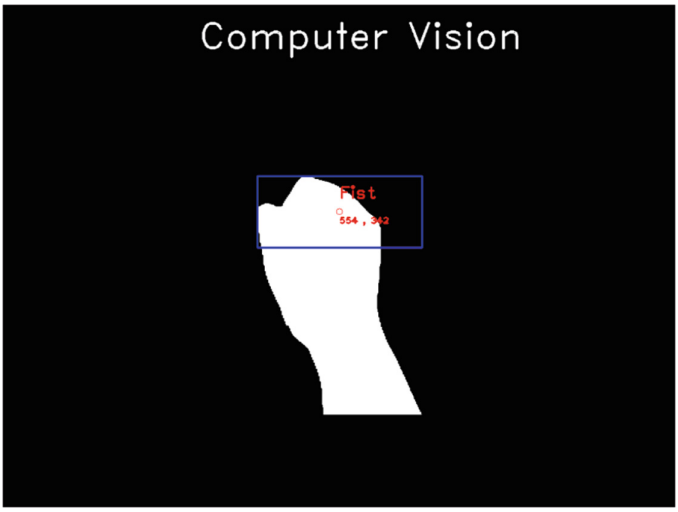

Fig. 4. Binary image with transform

OpenCV implements a findContours module that "retrieves contours from the binary image," which is useful for "shape analysis and object detection and recognition" [7]. For each of the detected contours, its area was computed and used to find the largest contour, which we assume to be the hand. To speed up the algorithm, only contours with an area >T (we used 5000), were considered. The process of finding the area of a contour is computed using image moments. Image moments are useful for describing properties of the contour, including the calculate center and area. Equations 2 and 3 are derived from OpenCV's documentation found in [7]. The [x, y] coordinates computed from the largest contour were assumed to be the hand.

$$\bar{x} = \frac{M_{10}}{M_{00}},\ \bar{y} = \frac{M_{01}}{M_{00}} \tag{2}$$

Eq. 2 Centroid [7]

$$Area(M_{00}) = \sum\sum x^0 y^0 I[x, y] \tag{3}$$

Eq. 3 Contour Area (Adapted from [7])

This solution works assuming that only part of the hand is present at a given time. In cases where the user extends their hand to span the entire vertical space of the mat, using the x,y coordinates provides through image moments fails. Equation 2 computes the centroid, which is the center coordinates of the contour. Thus, when the hand extends the entirety of the hand, the coordinates will be centered around the wrist. To compensate, a simple transformation function was applied. The result of the image tracking can be seen in Fig. 4.

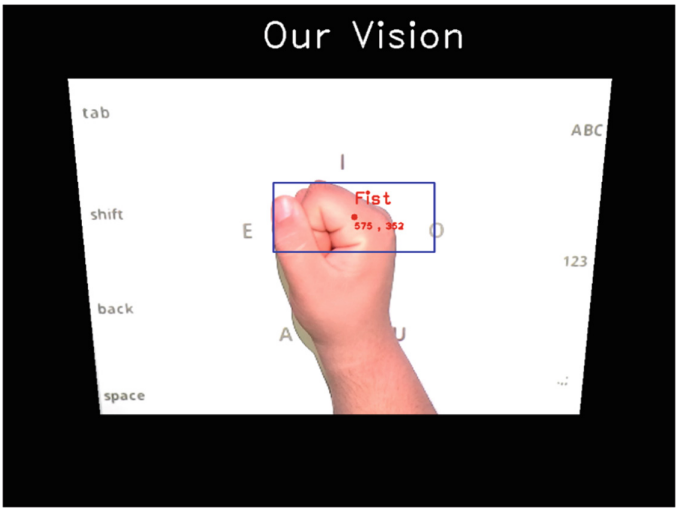

Fig. 5. User's perspective

In certain situations, or behavioral input, the centroid coordinates may be inconsistent, generating sporadic movement that in turn triggers incorrect input to the keyboard. To counteract this behavior and provide stability for those with muscle tremors, a trivial averaging function is applied to the input. The number of subsequent images used to calculate the average can be changed to determine how much averaging should be provided to minimize sporadic behavior. This was also useful to eliminate inconsistencies during image processing, which can be affected by lighting conditions or new stimuli.

The actual centroid position is proportional to the amount of area present. More area will affect how far the centroid is translated in the vertical direction. For example, when the user has their hand near the top of the mat, the corresponding centroid will also be at the top. We assume the approximate area for a hand at the top of the mat, which came about to be about 80,000 pixels. Using this information, we compute the area once the hand is visible as a proportion of the total pixels and apply it to the y-coordinate of the centroid. A quadratic expression can also be substituted for increased accuracy. As the area of the hand gradually increases, we want it to affect the y-coordinate greater once it has surpassed the center of the image. The current implementation uses a quadratic transform.

As discussed previously, the keyboard pauses for three seconds between keypresses to assure that the user does not accidently hit another button if they are moving their hand across the screen. At any screen in the application, the user has the ability to reset the keyboard back to the default screen by either keeping their hand stationary for seven seconds or by opening their hand. The first method is computed by looking at the area over a period of time. If the area did not change within a given threshold, the keyboard will automatically reset. Similarly, if the area of the hand suddenly changes by a decent amount, such as when opening their hand, it will also reset the keyboard. Both of these settings can be directly configured in the keyboards settings.

6 Unity Interface

The coordinates generated by the centroid of the largest contour directly control the cursor in the unity environment, which is used to interact with the keyboard. Each of the buttons and letters projected on the mat have colliders that detect when the coordinates of the cursor have overlapped onto the respective object, thus triggering the event as shown in Fig. 5. The vision implementation added additional challenges not present in the traditional touch version. For example, assuring that a button was pressed only once and resetting the keyboard after ten seconds of inactivity were essential features.

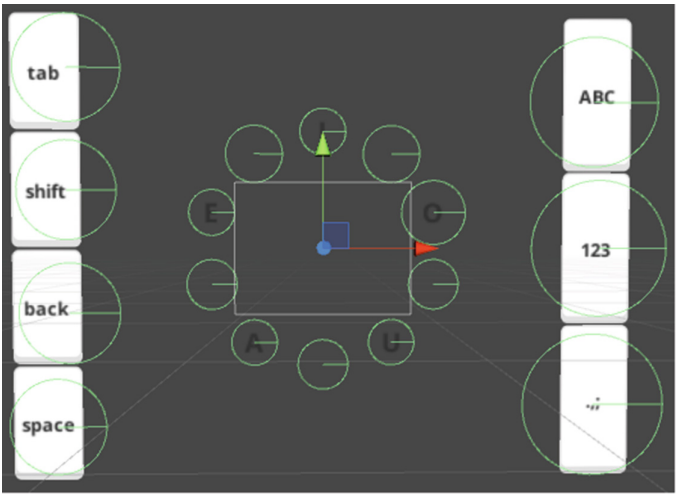

Fig. 6. Colliders

7 Practical Applications

The current version of the keyboard provides the user the capability to type text and control the mouse by tracking the location of the hand or arm. The initial release provides users the functionality of a real keyboard, including ways to type letters, numbers, and special characters. Additional buttons such as tab, shift, back, and space were also included for a more streamlined experience. The keyboard currently only supports these primary buttons as a method of input. Other buttons, such as the function keys or volume control, will be added in the future.

In addition, the current version allows the user to control the mouse. Currently, if the user stops moving the mouse, the program will delay for a specified amount of time, such as half a second, and then perform a double click action. This is useful if the user wants to open an application. The keyboard, which is displaying on the bottom screen, has an application button which provides a list of common applications that can launched by moving the mouse over the text, as shown in figure. The interface, which is running as a Unity application, communicates with a python script over TCP that is responsible for launching and putting the application into focus. Once in focus, the user can interact with the application similar to how they would with a standard mouse. An example of the system running can be seen in Fig. 6 (Fig. 7).

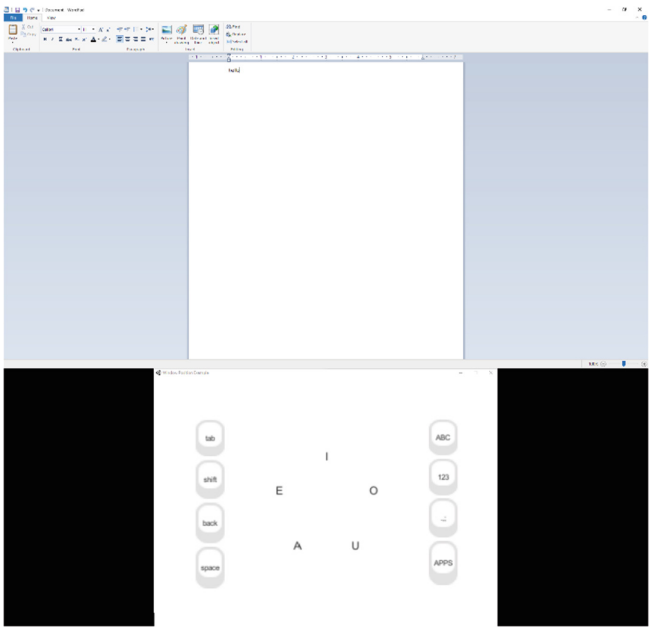

Fig. 7. Controlling external applications

8 Future Work

Many additional accessibility features will be added to RingBoard in the near future. The first will be the integration with the Windows OS. The Unity application will connect to a script responsible for controlling the computer based on the users input. For example, if the user presses the 'mouse' button, the user will be able to control the cursor on the top screen, allowing them to interact with the computer. The coordinates are mapped and sent using TCP to a script responsible for setting the position of the cursor in real time.

Furthermore, the user will have the capability to interact with common applications such as web-browsers and text-based entry applications. Current work at the time of publishing has implemented the ability for the user to type the name of the application in RingBoard, which is sent to the script responsible for spawning an instance of the application and focusing the window on the top screen. From there, the user can interact with the application using the mouse and keyboard.

Improvements to the tracking process will be continuously updated allowing for smoother control of the application while interacting with the keyboard. We are currently developing methods to provide a seamless interaction with Windows. Common challenges include handling situation requiring double-clicking events and handling multiple instances of the same application. Our goal is to have the application run at boot, which will automatically display the keyboard once the application reaches the login screen.

In addition, another area of improvement may be the addition of predictive text. Overtime the keyboard should learn the user's commonly typed phrases and be able to suggest them as additional options to the letters presented. Thus, the users would only have to navigate to the corresponding text to complete the word or phrase. These features are still being evaluated for usability and accessibility.

RingBoard provides an accessible way for those with muscular or cognitive, and verbal disabilities to interact with technology. Although current systems provide accessibility features such as text-to-speech and visual enhancements, such as color contrast and magnifying, there has not been much research in areas for manual input to a computer. RingBoard seeks to provide an accessible solution that can be easily used by all.

References

1. Sholes, C.S.: Type-Writing Machine. US Patent 207, 559, filed March 8, 1875, issued August 27 (1878)
2. Wojcik, B., Morelli, T., Hoeft, B.: RingBoard – a dynamic virtual keyboard for fist based text entry. The Journal on Technology and Persons with Disabilities (2018)
3. Fentek Industries: Computer Keyboard Keyguard Products. http://www.fentek-ind.com/Keyguard.htm#.Wy_SCqdKiUk. Accessed 23 June 2018
4. Ablenet. https://www.ablenetinc.com/. Accessed 23 June 2018
5. SmartNav by NaturalPoint: SmartNav 4: AT Overview. http://www.naturalpoint.com/smartnav/products/4-at/. Accessed 23 June 2018
6. LC Technologies, Inc.: Communicate with the world using the power of your eyes. http://www.eyegaze.com/eye-tracking-assistive-technology-device/. Accessed 22 June 2018
7. OpenCV. Web. Structural Analysis and Shape Descriptors (2018). https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=findcontours#findcontours
8. Ahsan, H., et al.: Vision based laser controlled keyboard system for the disabled. In: Proceedings of the 7th International Symposium on Visual Information Communication and Interaction. ACM (2014)
9. Missimer, E.S., et al.: Customizable keyboard. In: Proceedings of the 12th International ACM SIGACCESS Conference on Computers and Accessibility. ACM (2010)
10. Norte, S., Fernando, G.L.: A virtual logo keyboard for people with motor disabilities. In: ACM SIGCSE Bulletin, vol. 39, no. 3 (2007)
11. Wandmacher, T., et al.: Sibylle, an assistive communication system adapting to the context and its user. ACM Trans. Access. Comput. TACCESS **1**(1), 6 (2008)
12. Armstrong, P., Wilkinson, B.: Test entry of physical and virtual keyboards on tablets and the user perception. In: Proceedings of the 28th Australian Conference on Computer-Human Interaction. ACM (2016)
13. Gkoumas, A., Komninos, A., Garofalakis, J.: Usability of visibly adaptive smartphone keyboard layouts. In: Proceedings of the 20th Pan-Hellenic Conference on Informatics. ACM (2016)
14. Yin, Y., et al.: Making touchscreen keyboards adaptive to keys, hand postures, and individuals: a hierarchial spatial abckoff model approach. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM (2013)
15. Rashid, D.R., Smith, N.A.: Relative keyboard input system. Proceedings of the 13th International Conference on Intelligent User Interfaces. ACM (2008)