ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 9(33), DOI: 10.17485/ijst/2016/v9i33/89777, September 2016* ISSN (Online) : 0974-5645

Gesture Recognition Algorithm: Braille-Coded Gesture Patterns for Touch Screens: Eyedroid

M. Shabnam* and S. Govindarajan

Department of Computer Applications, SRM University, Kattankulathur, Kancheepuram – 603203, Tamil Nadu, India; shabnam.aslam@gmail.com, sgrsgr@yahoo.com

Abstract

Objectives: The primary objective of this paper is to evaluate the correctness of the algorithm Gesture Recognition that is one of the parts of our novel method 'Braille coded gesture pattern interaction method"; and to study the performance of Eyedroid system using the statistical methodologies. **Methods/Statistical Analysis:** The algorithm for gesture recognition was developed and implemented in C program to check the correctness of each step of algorithm. The gesture input parameters Swipe-Minimum-Distance, Swipe-Threshold-Velocity are defined with range of values. Varying gesture input range of values within domain and out of domain results the expected solution space. Our gesture recognition algorithm is developed for 2D touch surface. The Real time experiments were conducted for 12 blind fold subjects to observe the different versions Eyedroid-A(E-A), Eyedroid-B(E-B) of gesture recognition algorithm perceptions, being implemented in smart phones. The mean WPM for 10 trials was recorded for 12 subjects with E-A and E-B. The two tailed paired t test is performed for the mean dependent sample data of wpm. **Findings:** The existing 3D gesture recognition algorithm Derivative Dynamic Time Warping (DDTW) analyzed by Katarzyna Barczewska, Aleksandra Drozd, 2013 was proven to be efficient method for air fly environment. Our gesture recognition algorithm is developed for 2D touch surface found to be effective by the results of algorithm test. Gesture Recognition algorithm finds the mal gestures and the results proved that there is a significant difference in respect of blind subjects for two different Eyedroid versions. The observed data reveals E-B (mean WPM=14.95) is easy to use than E-A (mean WPM=12) as the number of words entered per minute by E-B is higher than E-A. Our Eyedroid with WPM is 14.95 is obtained to be better interactive system than the existing "Adaptive interaction Technique" system developed by Georgios Y fantidis Grigori Evreinov, 2005 confirms WPM is 12. **Applications/Improvements:** The implication is our Braille coded gesture pattern that enables visually challenged ones to enter more number of words than by the existing system. Our interaction method is deployable at all kinds of touchable devices.

Keywords: Eyedroid-Braille Coded Gesture Pattern, Gesture Recognition Algorithm, t Test for Mobile Interaction

1. Introduction

The literature survey of this research work was done and presented in our previous paper "Survey on the Text entry methods"[1]. This paper is basically based on our previous research work published in "Braille coded Gesture patterns"[2] which describes about the architectural design, and design implications of Eyedroid system. The paper "Framework of Gesture for blind people usable at touch mobile phones "[3] demonstrate Eyedroid architecture design; and from bottom to top of architecture stack of Eyedroid, the forth layer specifies the "User interaction module". paper extracted the forth layer user interaction module that compose the gesture recognition algorithm[3]. Eyedroid can be extended for multi platform and not restricted to Android platform only as given in[4].Braille coded gesture methods require no other hardware components[5]other than a mobile device. From the feedback of subjects participated in the previous experiments suggest the major impact of spatial problem in interacting with the touch mobile device. That is the user input the text in touch screen by tracing the space over to select a particular character; that lengthen the character input time. Adaptive blind interaction method requires the user to use both hands to interleave the layers of alphabet design. Our Braille coded gesture method is

* *Author for correspondence*

different from other methods such as Adaptive[6–8]. Blind interaction, the soft keyboard[10] placed on the screen and utilizes swipe gesture input method which requires two hands to gesture. The spatial[11] problem is addressed by providing a layout less interface incorporated with the braille coded gesture method in our Eyedroid system. But still the character input time has to be improved and the error rate has to be reduced in text entry method. Having these criteria with mind we reduced the number of pixels covered through a gesture on the touch screen. Hence, the Eyedroid system is modified by changing the parameter threshold values which leads to the redesigned system E-B. As the overall interaction space is reduced subjects are able to enter text more quickly on E-B implemented mobile phone than with E-A devices. The average wps (words per second) is obtained in experiment with E-A is 12, the average wps rate obtained in experiment with E-B is 14.95. The section one and two of part of this paper describes the algorithm steps and testing; and section three describes the Eyedroid implementation aspects.

2. User Interaction Module

The following are the parts of user interaction module of Eyedroid architecture.

- Gesture pattern recognition module
- Alphabet code creation list module
- Mapping alphabet code and gesture pattern
- Miscellaneous gesture module

As described in Braille coded gesture patterns[2]four gestures are required to recognize one particular character, and the first module is detecting the gesture pattern, second module is creating the different fling assign to alphabets, third module coordinates gesture pattern and enters gesture pattern, miscellaneous module creates gestures for special inputs such as clear the screen and back space character. The algorithm for gesture recognition was developed based on DFS[5]with the solution space of gestures {Right, Left, Top, Bottom}. This is Backtracking heuristic search of respective gesture for given inputs; is the solution is not obtained it returns Mal Gesture'.

2.1 Gesture Recognition Algorithm

The input variables X1, X2, Y1, Y2 are the X, Y screen coordinates, velocity-X, velocity-Y are the fling speed. The Minimum swipe distance SWIPEMINDISTANCE is set as 100 pixels. The minimum fling speed SWIPETHRESHOLDVELOCITY is set as 2000 pixels. The variable X1 is the starting point of x coordinate, X2 is ending point of x coordinate. Similarly, the variable Y1 is the starting point of Y coordinate, Y2 is the ending point of the Y coordinate. The constant variables are considered with the constant values of 100 pixels and 2000 pixels for the mobile screen with resolution 540*960. The logic of gesture pattern identification is that *Step 1* reads the input values. Step 2 checks the value of X2 is greater than X1; X1 is the starting point of x coordinate, X2 is ending point of x coordinate. The decision is Right fling if ending point X2 is greater. At the same time Y1, Y2 will be zero. Step 2 checks another constraint such as total number of pixels covered with default pixel rate is greater than the Swipe minimum distance (SWIPEMINDISTANCE) constant; and speed of gesture is greater than swipe speed threshold value (SWIPETHRESHOLDVELOCITY). Similarly steps 3 to 4 checks the minimum swipe distance, starting and ending X, Y coordinate values and correspondingly interpret Left to right fling/Top to Bottom fling/Bottom to Top fling. *Step-2* or *step-3* will be executed when Y1, Y2 is zero and write either left or right gesture; *Step-3* or *step-4* will be executed when X1, X2 is zero and write either top or bottom gesture.

2.1.1 Explanation of Gestures

Based on the usable gestures[12] for touch screen devices we have considered the following gesture patterns to form the combinations of Braille patterns

- LEFT - fling from left to right
- RIGHT – fling from right to left
- TOP - fling from top to bottom
- BOTTOM – fling from bottom to right

This algorithm is the base for E-A, E-B text input Braille based gesture interaction method. The parameters N_p(SWIPEMINDISTANCE), S_g(SWIPETHRESHOLDVELOCITY), T_g(PIXEL-RATE) values are changed in between the two systems with the aim of obtaining. The algorithm is implemented in C programming language. The outputs for the different given inputs of above algorithm is described in tabulated form Table 1. The coordinates of gesture lie in XY plane

of the mobile screen and the X axis varies from X1 to X2 in values -1 to +1. (Table 2). The Y axis varies from Y1 to Y2 in values -1 to +1. The input values are random values taken from mobile screen XY plane. Figure 1 shows the model of mobile screen plane with X, Y coordinate position range values. The algorithm produces expected outputs for the different combination of inputs. As said in the algorithm X1, X2, Y1, Y2, velocity-X, velocity are the inputs and output is any one from the gesture group (Top, Bottom, Right, Left). When analyzing the output values for given input; It is observed in the algorithm results such as the values of Y coordinates variables Y1, Y2 are equal then there is no movement across Y axis so the gesture will be in horizontal way only. Similarly, when the values of X coordinate variables X1, X2 are equal it implies that there is no gesture across X coordinate, hence the gesture will be either Top or Bottom. The red colored output value in the table is no output for invalid outputs.

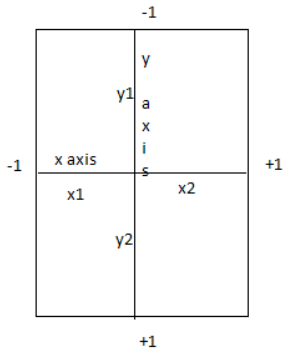

Figure 1. Mobile screen plane model.

Table 1. Algorithm for gesture recognition

/* Gesture Recognition Algorithm */

Comment: gesture/fling by user recognition (Gesture recognition)

Comment: The screen resolution of the mobile device is considered as 540*960

Comment: write the type of gesture from the list(Right, Left, Top, Bottom)

SWIPEMINDISTANCEß100px;

SWIPETHRESHOLDVELOCITYß2000px;

Pixel-rateß200px/Sec.

Step1. Read X1,X2,Y1,Y2,velocity-X,velocity-Y

Step 2. If (**abs**(X1)>**abs**(X2)) **&&if**((**abs**(X1-X2)*pixel rate)>SWIPEMINDISTANCE)

&&**If** (velocity-X> SWIPETHRESHOLDVELOCITY) **then**

return "Right gesture"

endif

Step 3. If (**abs**(X2)>**abs**(X1)) **&& if** ((**abs**(X2-X1)*pixel-rate)>SWIPEMINDISTACE)

&& if (velocity-X>SWIPETHRESHOLDVELOCITY) **then**

return "Left gesture"

endif

Step 4. If (**abs**(Y1)>**abs**(Y2)) **&& if**(**abs**(Y1-Y2)*pixel-rate)>SWIPEMINDISTANCE)

&& if (velocity-Y>SWIPETHRESHOLDVELOCITY) **then**

return "Bottom gesture"

endif

Step 5. If (**abs**(Y2)>**abs**(Y1)) **&& if**(**abs**(Y2-Y1)*pixel-rate)>SWIPEMINDISTANCE)

&& if (velocity-Y>SWIPETHRESHOLDVELOCITY) **then**

return "Top gesture"

endif

Table 2. Algorithm execution display table

S. No	Input values						Output values
	X1	X2	Y1	Y2	Velocity-X	Velocity-Y	Output gesture
1	-1	0.6	0.5	0.5	2500px	3000px	No output
2	-0.8	0.9	0.1	0.1	3090px	4000px	No output
3	-0.2	1	0.3	0.3	2500px	3200px	Right
4	-0.9	0.2	0.4	0.4	2100	2400	Left
5	-0.2	0.1	0.5	0.5	2200px	2600px	Right
6	0.8	0.8	0.4	0.4	3000	2500	No output
7	0.9	0.9	-0.6	0.2	2800px	3200px	No output
8	-0.4	-0.4	-0.9	0.1	3200	2500	Top
9	0.7	0.7	-.8	0.2	3000	3100	Top
10	0.7	0.7	-0.2	0.8	2500	2400	Bottom
11	0.6	0.6	-0.3	0.5	3000	2300	No output
12	-1	0.4	0.1	0.1	1200	1900	No Output
13	0.5	0.5	-.2	1	1600	1800	No Output

2.1.2 Algorithm Test Result Analysis

From the Execution-Display Table 2 of algorithm,

S.no 3: Y1, Y2 values are same that is 0.3(implies gesture must not be across Y) and X2> X1 and swipe distance (X2-X1*pixel-rate) is 160px which is greater than 100px; and the fling speed measured by velocity-X, velocity are greater than 2000px; hence the output gesture "Right" is obtained according to algorithm1 step 3. That is the Y axis range values are same so the gesture should be either from left to right or from right to left. The X axis range values decides the gesture such that X2 is greater hence the gesture started from Right and end with Left; other constraints such as number of pixels covered and Speed of gesture is above the threshold values. Hence the expected solution obtained for given inputs X1, X2, Y1, Y2, velocity-X, velocity-Y .

S.no 4: Y1, Y2 values are same that is 0.4(implies gesture must not be across Y) and X1> X2 and swipe distance (X1-X2*pixel-rate) is 140px which is greater than 100px; and the fling speed measured by velocity-X, velocity-Y are greater than 2000px hence the output gesture "Left" is obtained according to algorithm1 step 2. That is the Y axis range values are same so the gesture should be either from left to right or from right to left. The X axis range values decides the gesture such that X1 is greater hence the gesture started from Left and end with Right; other constraints such as number of pixels covered and Speed of gesture is above the threshold values. Hence the expected solution obtained for given inputs X1, X2, Y1, Y2, velocity-X, velocity-Y .

S.no 5: Y1, Y2 values are same that is 0.5(implies gesture must not be across Y) and X2> X1 and swipe distance (X2-X1*pixel-rate) is 160px which is greater than 100px; and the fling speed measured by velocity-X, velocity-Y are greater than 2000px hence the output gesture "Right" is obtained according to algorithm1 step 3. That is the Y axis range values are same so the gesture should be either from left to right or from right to left. The X axis range values decides the gesture such that X1 is greater hence the gesture started from Right and end with Left; other constraints such as number of pixels covered and Speed of gesture is above the threshold values. Hence the expected solution obtained for given inputs X1, X2, Y1, Y2, velocity-X, velocity-Y

S.no 8: X1, X2 values are same that is 0.4(implies gesture must not be across X) and Y1>Y2 and swipe distance (Y1-Y2*pixel-rate) is 160px which is greater than 100px;and the fling speed measured by velocity-X, velocity-Y are greater than 2000px hence the output gesture "Top" is obtained according to algorithm1 step 4. That is the X axis range values are same so the gesture should be either from Top to Bottom or from Bottom to Top. The Y axis range values decides the gesture such that Y1 is greater hence the gesture started from Top and end with Bottom; other constraints such as number of pixels covered and Speed of gesture is above the threshold values. Hence the expected solution obtained for given inputs X1, X2, Y1, Y2, velocity-X, velocity-Y

S.no 9: X1, X2 values are same that is 0.7(implies gesture must not be across X) and Y1>Y2 and swipe distance (Y1-Y2*pixel-rate) is 120px which is greater than 100px; and the fling speed measured by velocity-X, velocity-Y are greater than 2000px hence the output gesture "Top" is obtained according to algorithm1 step 4. That is the X axis range values are same so the gesture should be either from Top to Bottom or from Bottom to Top. The Y axis range values decides the gesture such that Y1 is greater hence the gesture started from Top and end with Bottom; other constraints such as number of pixels covered and Speed of gesture is above the threshold values. Hence the expected solution obtained for given inputs X1, X2, Y1, Y2, velocity-X, velocity-Y

S.no 10: X1, X2 values are same that is 0.7(implies gesture must not be across X) and Y2>Y1 and swipe distance (Y2-Y1*pixel-rate) is 120x which is greater than 100px; and the fling speed measured by velocity-X, velocity-Y are

greater than 2000px hence the output gesture "Bottom" is obtained according to algorithm1 step 5. That is the X axis range values are same so the gesture should be either from Top to Bottom or from Bottom to Top. The Y axis range values decides the gesture such that Y2 is greater hence the gesture starts from Bottom and ends with Right; other constraints such as number of pixels covered and speed of gesture are above the threshold values. Hence the expected solution obtained for given inputs X1, X2, Y1, Y2, velocity-X, velocity.

The above Serial number cases are finds the expected outputs for the given inputs. The no output cases are discussed below; they are also expected outputs for the given input. The input values of following cases are given as out of domain input values hence they return no outputs. It is concluded that the algorithm is correct to recognize gestures in touch screen.

2.1.3 Justification of No Output Cases for Out of Range Input Values

From the Execution display Table 2 of algorithm

S.no 1: Y1, Y2 values are same i.e. 0.4(implies gesture must not be across Y) and X1>X2 and swipe distance (X1-X1*pixel-rate) is 80x which is not greater than 100px; hence the algorithm returns no output solution. Here SWIPEDISTANCE value that is the number of pixels covered is less than the minimum threshold value, so the algorithm returns 'No output'. This will happen when the subject lightly flings the screen that is he gives the false gesture.

S.no 2: Y1, Y2 values are same that is 1(implies gesture must not be across Y) and X2, X1 and swipe distance (X2-X1*pixel-rate) is 20x which is not greater than 100px; hence it is returned No output according to algorithm1 to step 3. Here SWIPEDISTANCE value that is the number of pixels covered is less than the minimum threshold value, so the algorithm returns 'No output'. This will happen when the subject lightly fling or touches the screen that is he gives the false gesture

S.no 6: X1, X2 values are same that is 0.8(implies gesture must not be across X) and the values of Y1, Y2 also same; and it doesn't satisfy any conditions from step 2 to step 5 in algorithm1, hence the algorithm returns no output solution. This implies the X, Y values are same hence the subject simply touches the screen rather than swipe the screen.

S.no 7: X1, X2 values are same that is 0.9(implies gesture must not be across X) and Y1> Y2 and swipe distance (Y2-Y1*pixel-rate) is 80px which is not greater than 100px; Hence it is returned as no output according to algorithm1 step 4. This is the similar case like S. No1,2.

S.no 11: X1, X2 values are same that is 0.6(implies gesture must not be across X) and Y2> Y1 and swipe distance (Y2-Y1*pixel-rate) is 40px which is not greater than 100px; Hence it is no output according to algorithm1 step 5. This is the similar case like S. No1,2.

S.no 12: Y1, Y2 values are same that is 0.1(implies gesture must not be across Y) and X2>X1 and swipe distance (X2-X1*pixel-rate) is 120px which is greater than 100px; But the velocities are less than 2000px.Hence it is returned no output according to the algorithm1 step 3. Here the speed is below the velocity threshold value; hence this implies the user swipe very slowly on the screen.

S.no 13: X1, X2 values are same that is 0.5(implies gesture must not be across X) and Y1> Y2 and swipe distance (Y1-Y2*pixel-rate) is 160px which is greater than 100px; But the velocities are less than 2000; hence it is returned no output. According to the algorithm 1 step 4 here the speed is below the velocity threshold value; hence this implies the user swipe very slowly on the screen.

The invalid (out of bound) input values return no output; so the algorithm is slightly modified by handling the exception cases. The exception can be handled in following two ways.

- If not output; then return repeat fling gesture.
- If no output; then rise voice message "wrong gesture" and repeat fling gesture. Here is

Table 3 is modified algorithm1 after adding exception handling method[13].

2.1.4 Algorithm Test Conclusion Study

The Gesture recognition algorithm concludes the aspects according to the test results such as the fling gesture in the mobile touch screen is recognized through the constrains

- The X (difference between X1, X2), and Y (difference betweenY1, Y2) values should be greater the absolute of 0.5; because in order to get the number of pixels covered to be more than 100px. ((0.5... to 1 * 200px)>100px); this case suggest that the user has to do a fling gesture of nominal distance; It should not be too short or too high on the screen for effective gesture recognition.
- The speed of gesture also at nominal speed; it should

Table 3. Algorithm with exception handling

/* Exceptions included Gesture Recognition Algorithm */

Comment: gesture/fling by user recognition (Gesture recognition)

Comment: The screen resolution of the mobile device is considered as 540*960

Comment: write the type of gesture from the list(Right, Left, Top, Bottom)

SWIPEMINDISTANCEß100px;

SWIPETHRESHOLDVELOCITYß2000px;

Pixel-rateß200px/Sec.

Step1. Read X1,X2,Y1,Y2,velocity-X,velocity-Y

Step 2. If (**abs**(X1)>**abs**(X2)) **&&if**((**abs**(X1-X2)*pixel rate)>SWIPEMINDISTANCE)

&&**If** (velocity-X> SWIPETHRESHOLDVELOCITY) **then**

return "Right gesture"

else return "False"

endif

Step 3. If (**abs**(X2)>**abs**(X1)) **&& if** ((**abs**(X2-X1)*pixel-rate)>SWIPEMINDISTACE)

&& if (velocity-X>SWIPETHRESHOLDVELOCITY) **then**

return "Left gesture"

else return "False"

endif

Step 4. If (**abs**(Y1)>**abs**(Y2)) **&& if**(**abs**(Y1-Y2)*pixel-rate)>SWIPEMINDISTANCE)

&& if (velocity-Y>SWIPETHRESHOLDVELOCITY) **then**

return "Bottom gesture"

else return "False"

endif

Step 5. If (**abs**(Y2)>**abs**(Y1)) **&& if**(**abs**(Y2-Y1)*pixel-rate)>SWIPEMINDISTANCE)

&& if (velocity-Y>SWIPETHRESHOLDVELOCITY) **then**

return "Top gesture"

else return "False"

endif

not be very slow. Of course the mobile accepts high velocity speed of fling gesture. That why there is no higher limit set for velocity in the algorithm.

- Mal gestures identified: 1. Slow swipe the screen, 2. Simple touch in the screen.

2.1.5 Reliability of Algorithm

The algorithm was tested with unit testing method to check the correctness of the algorithm steps and the resultant tabulated input output values are valid for all 13 trials of the algorithm. The structure of algorithm is search the solution based on the condition; return output if the solution is found; otherwise it returns exception handling message; Hence the algorithm is derived as Gesture recognition Backtracking algorithm with 100% reliability is achieved for the known touch panel configuration. The other interaction modules such as Alphabet code creation list module, Mapping alphabet code and gesture pattern,

miscellaneous gesture module will be mentioned in our next paper.

3. Eyedroid-B: Threshold Values are Reduced

Design specification of E-A is elaborated in [14] this paper. Here is E-B specification. The difference between E-A and E-B are given by Figures 2 and 3 respectively. The figures reference the pixel coverage is more in E-A and less in E-B.

- Threshold value for the number of pixels covered (N_p) is 100 pixels (px) per gesture and is now reduced to 40 px per gesture.
- Threshold value of speed (S_g) is 2000 PX per gesture and is now reduced to 800 PX per gesture.
- Threshold value of the gesture time (T_g) is 1000 ms per gesture and is now reduced to 300 ms per gesture

The E-B system is very accurate and is faster than the E-A system. The user utilizes only the small area covering 40 to 60 pixels for the entire screen pixel resolution. Additionally, the E-B system becomes accurate and error free gesture system. The E-B system virtually creates a joystick system[15].

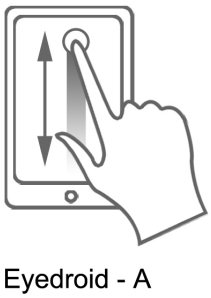

Figure 2. Fling bottom to top gesture on Eyedroid-A.

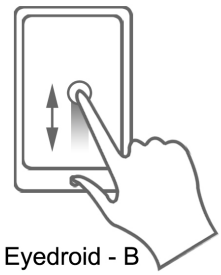

Figure 3. Fling bottom to top on Eyedroid-B.

However, this virtual joystick can be created anywhere on the touch screen not necessary from very Top or Very Bottom from or very Left from or from Very Right. The real time experiments conducted with visually challenged people. The objective is to find WPM by individual subject with E-A and E-B and to compare the ease of use in versions of Eyedroid system.

3.1 Experiment

This experiment is performed followed by the experiments done in the previous paper with the same set of subjects. The first level of the Kirkpatrick[3]evaluation reveals that participants make errors when performing the figure gesture. Some participants hold the mobile phone in the palm and use the thumb to do the gesture[16]. Some participants used another hand's index finger to do the gesture. Those who use the thumb or even the index finger did partial gestures, that is, to swipe from top-to-bottom, they swipe a little from top-to-bottom; to swipe from bottom-to-top; they start at the bottom and move up towards the right, which causes confusion for gesture recognition by the screen system. To avoid these errors, E-B was developed. The touch recognition accuracy parameters' threshold setting is reduced as mentioned in the design of E-B. This live experiment compares the performance of E-A and E-B with respect to visually challenged participants. E-B is not very different from E-A. Hence, a small introduction was sufficient for the participants regarding E-B, and they started to use it flexibly. The snapshot of subjects working at experiment in shown in Figure 4.

Figure 4. Blind subjects at experiment with Eyedroid.

3.2 Procedure

This experiment includes two sessions. The character input time was observed with high threshold values for the E-A system in session 1 and tabulated in Table 4. The character

input time was observed with low threshold values for the E-B system. Both sessions [13][14][15] consists of 12 sub sessions for individual participant activity.

Table 4. Real time experiment results

Blind person	Words per minute (WPM)	
	High Threshold-Eyedroid-A	Low Threshold-Eyedroid-B
1	12	12
2	13	17
3	11	15
4	10	17
5	12	17
6	13	14
7	10	15
8	11	13
9	15	14
10	16	15
11	11	16
12	10	14

The average wpm E-A 12.
The average wpm E-B 14.95

3.3 Technical Setting

E-A was installed in one Samsung Galaxy mobile device, and output feedback was received by the head phone. E-B was installed in another Samsung Galaxy mobile device, and output feedback was received from the head phone.

3.4 Data Analysis

The sample data "Words Per Minute" is calculated for ten times per subject members of 12 with E-A deployed device and E-B deployed device. The time of successful word entry was measure with a timer. The number of words entered by the subject was observed and data recorded and tabulated. The post-hoc paired t test was performed for the empirical sample data in order to analyze the performance of subject's usage with two gesture input methods (E-A and E-B).The aim of the experiment is to find the ease of use of Eyedroid two versions by the blindfolds.

3.4.1 Review of Data

The result of this experiment is pretty straightforward. The value of t is -3.80 with the significant level 0.05 and the value of p is 0.002929. The probability $p<0.05$ which implies there is a significant difference in between the E-A and E-B methods in respect of subjects.

3.4.2 Discussion

The null hypothesis μ_0 is zero because the statement "no effect on the changing threshold parameter values of gesturing" is false. Hypothesis μ_1 is <0.05, that is, there is a great effect on the change of threshold values for the touch screen accuracy factors. Visually challenged participants find that the method (E-B) Braille-coded gesture input method with low threshold values for the number of pixels covered in the gesture, the low-speed of the gesture, and the minimum duration of the swing time as more comfortable, using the full, effective interaction method. The E-B system has minimum factor values. The E-B system creates a virtual joystick, that is, instead of using the entire screen size. For gesturing, the user can use a small area of 1.5 inches for the entire screen size. Hypothesis is μ_2 is true which is greater than p value, which proves the statement true that the E-A system is less effective than the E-B system for the visually challenged participants.

3.4.3 Applicability of the Results

We have created a new text input method for touch screens using finger gestures. The results of experiment-1 show that the button-based, touch-and-tell, and gesture methods perform equally with a little practice by the participants on each method. The average value of the time taken by participants to input a character is not very different between the three methods. The results of experiment-2 show that the long-term training on the gesture input method have a great effect on system adaptability. The significant difference shows that the long-term training allows the visually impaired to use the gesture input method more easily. Based on the feedback of the visually impaired participants at the time of training, the E-B system was developed and tested with participants in experiment 3. The E-B system is faster than the E-A system. The E-A system covers more space than the E-B system. Because the number of pixels covered by a gesture is smaller in the E-B system than the E-A system and the E-B system is proven as the best input system with an average character input time of two seconds.

4. Conclusion and Future Enhancement

The Braille coded gesture recognition algorithm was

developed and tested for its reliability and the invalid gestures are identified and listed. Our next studies continue with analyzing the algorithm for other mobile platform dependencies, and testing and analyzing the performance of gesture pattern mapping algorithm to improve the number of characters' enterer in the mobile phones and to extend the system to other wearable touchable devices. The layout less gesture based interaction method, Braille coded gesture pattern interaction method was developed as Eyedroid implementable at Android operating system and versioned as E-A and E-B. Real time tested between human and machine and experiment results are discussed. The discussion ends with the Eyedroid-B system is making a screen joystick which enables the little faster text entry system.

5. References

1. Shabnam M, Govindarajan S. Survey on the text entry methods used in touch screen mobile devices by visually challenged. International Journal of Advanced Intelligence Paradigms. In press.
2. Shabnam M, Govindarajan S. Braille-coded gesture patterns for touch- screens a character input method for differently enabled persons using mobile devices. Proceedings on International Conference on Communication, Computing and Information Technology ICCCMIT. 2014; (1):1–5.
3. Aslam SM, Swaminathan G. Framework of gesture for blind people usable at touch mobile phones. International Journal of Applied Engineering Reasearch. 2015; 10(17);37658–63.
4. Kang H, Cho J, Kim H. Application study on android application prototyping method using app inventor. Indian Journal of Science and Technology. 2015 Aug; 8(18):1–5.
5. Ulusoy M. A touch based finger-motion-adaptive control design for braille reading. Northeastern University; 2015. p. 157.
6. Gallavata G, Ewerth R, Freisleben B. A robust algorithm for text detection in images. 2003 Proceedings of the 3rd International Symposium Image and Signal Processing and Analysis, ISPA. 2003; 2:611–16.
7. Brownlee J. Clever algorithm: Nature inspired programming recepies. ACM Digital Library; 2011.
8. Georgios Yfantidis, Evreinov G. Adaptive blind interaction technique for touchscreens. Universal Access in the Information Society. 2006 May; 4(4):328–37.
9. Southern C, Clawson J, Frey B, Abowd G, Romero M. An evaluation of braille touch: Mobile touch screen text entry for the visually impaired. Proceedings of the 14thACM Conference on Hunan-Computers interactions with mobile devices and services, NY; 2012. p. 317–26.
10. Silfverberg M, MacKenzie IS, Korhonen P. Predicting text entry speed on mobile phones. Proceeding of the ACM SIGCHI Conference on Human Factors in Computing, CHI'00, USA; 2000. p. 9–16.
11. Prakash M, Gowshika U, Ravichandran T. A smart device integrated with an android for Alerting a Person's health condition: Internet of things. Indian Journal of Science and Technology. 2016 Feb; 9(6):1–6.
12. Hotelling S, et al. Gestures for touch sensitive input devices; 2013.
13. Armitage P, Berry G, Mathews JNS. Statistical methods in medical research.
14. Ott R L, Longnecker M. An introduction to statistical methods and data analysis. Nelson Education; 2015.
15. Daniel WW, Wayne WD. Biostatistics: a foundation for analysis in the health sciences; 1995. p. 1–3.
16. Prakash M, Gowshika U, Ravichandran T. A smart device integrated with an android for Alerting a Person's health condition: Internet of things. Indian Journal of Science and Technology. 2016 Feb; 9(6):1–6. DOI: 10.17485/ijst/2016/v9i6/69545.