iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 3 **Issue:** VI **Month of publication:** June 2015

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Remulation-Based Keyboard Evaluation System: A Review Paper

Neha Ramesh Jain
Department of Computer Engineering, Sarvajanik College of Engineering and Technology, Surat, Gujarat (India)

Abstract - Remulation, a complementary method for evaluating touchscreen keyboard correction and recognition algorithms. It replicates prior user study data through real-time, on-device simulation. Octopus is an evaluation tool that enables keyboard developers to efficiently measure and inspect the impact of algorithmic changes without conducting resource-intensive user studies. It can also be used to evaluate third-party keyboards in a "black box" fashion, without access to their algorithms or source code. Octopus can evaluate both touch keyboards and word-gesture keyboards. This contributes to study a novel approach of evaluating touchscreen keyboards, which includes: Remulation, a new approach for evaluating keyboard correction and recognition algorithms by replicating prior user study data with real-time simulation and design of Octopus, a Remulation-based keyboard evaluation tool and system.
Keywords – Touchscreen, Keyboard, Remulation, Gesture, Simulator

I. INTRODUCTION

Mobile phones are becoming powerful computers, sized to fit in one's hand. What before was a device for calling and texting is now a smartphone with embedded technologies, such as cameras, light sensors, vibro tactile actuators, accelerometers, gyroscopes, GPS receivers, and so on. Today's uses for smartphones extend well beyond calling and texting, and include the capability to write and send e-mail and even type out notes and complete documents using a note app or word processor.
However, such technological progress has also caused a change in traditional practices. A common trend among many smartphones is the removal of the physical keypad, a feature that was common to all mobile phones just a few years ago. Many smartphones today employ touchscreen technology. The devices use on-screen "soft" keyboards. Many of these soft keyboards resemble the classic and renowned Qwerty keyboard layout.We now discuss comparison-collecting input by various keyboards in the following section.

A. Types of Touch Screen Keyboards

The grand mean for entry speed over all 432-phrase iterations was 45.7 wpm [3]. The Octopus keyboard averaged 54.7 wpm, which was 1.4% faster than the standard Qwerty keyboard at 54 wpm, which is also depicted in Fig. 1.1. The T+ keyboard averaged 38.7 wpm, about 9.6% faster than Curve at 35.3 wpm. The Octopus and standard Qwerty are both approximately 40% faster than both T+ and Curve. The large difference between the standard Qwerty and Octopus keyboards, and the Curve and T+ keyboards we attribute mainly to the participant's lack of experience with Shape Writing as well as to their lack of experience with a T9-like system in a touchscreen context. Not surprisingly, the differences in entry speed by keyboard variant were statistically significant. A Bonferroni-Dunn [3] post hoc test revealed that all pairwise comparisons were statistically significant except standard Qwerty vs. Octopus and Curve vs. T+ as shown in the Figure 1.

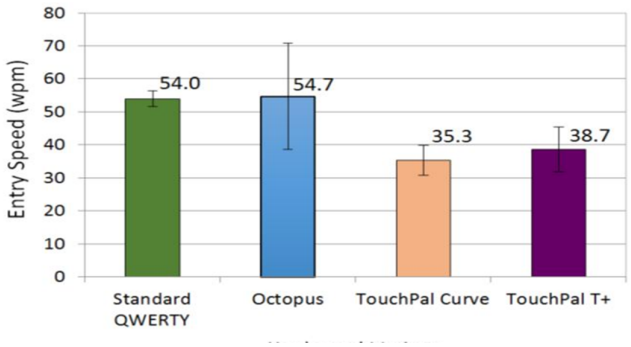


Fig. 1Entry speed (wpm) by keyboard variant[3]

Although the standard Qwerty and Octopus keyboards performed similarly overall, an examination of the learning progression over the 9 phrase iterations reveals a different outcome. As seen in Fig. 2, on the first iteration, Octopus had the slowest entry speed. This was likely due to visual demand, since participants did not have prior experience with an Octopus or similar keyboard. The Octopus does not suggest the desired words at first. In [3], participants were observed to first look around the keyboard and then continue to type regularly if the word did not show up as a suggestion. Nonetheless, a remarkable 187% increase in entry speed for the Octopus keyboard from the first iteration to the last was observed. Overall, the effect of phrase iteration on entry speed was statistically significant, observed in [3].

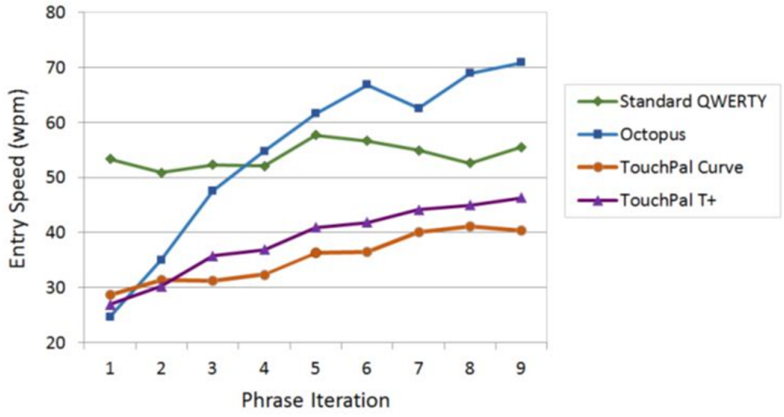


Fig. 2 Entry speed (wpm) by phrase iteration and keyboard variant [3]

II. THEORATICAL BACKGROUND AND LITERATURE

A. *Introduction*

Today, touchscreen keyboards are used by hundreds of millions of people around the world as their default text entry method. To reduce typing errors, most of these “Smart Touch Keyboards” (STKs) correct errors automatically as users type. However, occasionally the error correction system itself makes a mistake, with undesirable and sometimes humorous consequences.

An increasingly popular alternative to the STK is the smart gesture keyboard (SGK). An SGK recognizes words based on the user’s finger gestures (Fig. 3). SGK’s are also known as Shape Writing keyboards, gesture keyboards or word-gesture keyboards in the literature. Different forms of SGKs have been commercially distributed in many products such as ShapeWriter, SlideIT, Swype, Flex T9, TouchPal, and the Android 4.2 stock keyboard (Gesture Typing). SGKs face the same correction challenges as STKs because they must map ambiguous finger gestures to words.

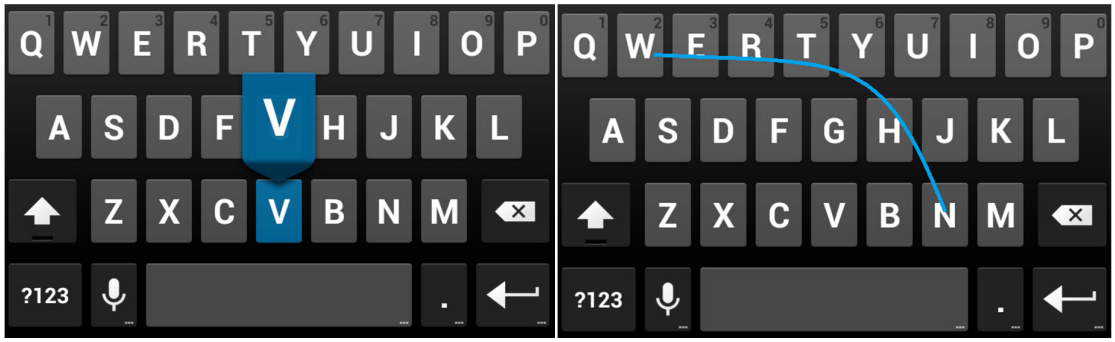


Fig. 3 Smart Touch Keyboard (a) and Smart Gesture Keyboard (b)[1]

HCI(Human Computer Interaction) research has explored various techniques for error prevention, including adapting decoding algorithms to hand posture, and personalizing ten-finger typing for large touchscreens.

As with other advanced UI technologies such as speech recognition, effective and efficient evaluation is critical to the improvement of smart keyboards. An evaluation generally consists of (1) data collection and (2) data analysis. The goal is to facilitate both stages of the process. Collecting keyboard output data typically involves a laboratory experiment with a dozen or moreparticipants. The time and labor required by these experiments make frequent evaluation of small algorithmic changes

infeasible. Moreover, current data analysis techniques do not provide important information about keyboard algorithms. For example, they do not explicitly and quantitatively measure an STK's ability to correct user errors, and the typical accuracy metrics (e.g., the MSD error rate) examine output at the character-level, and do not reflect today's keyboards' word-level behaviors.

To expand the repertoire of tools and methods for evaluating STKs and SGKs,*Remulation,* a novel keyboard evaluation approach that measures correction and recognition algorithms of keyboards by *re*playing previously collected user data through real-time on-device si*mulation*. It takes touch screen events recorded.

B. Related Work

Collecting natural use data and applying them to train recognition algorithms has been widely adopted as a research methodology in AI (e.g., speech and handwriting recognition). Recently, it has also been employed to design keyboard algorithms. Gunawardana et al. used pre-recorded data on a keyboard to train and evaluate their own error correction algorithm. Remulation uses some of the same techniques for the different problem of comparing two or more third-party keyboards when the source code or algorithm is unavailable. This work shows that it is possible to send "fake" touch events to today's mainstream devices (Android in particular but other OSs in principle) and evaluate keyboards in a "black box" fashion.

In what follows we discuss relevant prior work on (1) simulating human text entry for evaluating keyboard performance without traditional laboratory studies, and (2) current analysis techniques used for assessing a text entry method's accuracy.

1) *Simulating Text Entry:* In 1982, Rumelhart and Norman described a model for simulating skilled typists on physical typewriters, with the goal of understanding human typing behavior. Their model focused on predicting keystroke timing and simulated key transposition and doubling errors. In this work, itrely on direct data replication rather than complex modeling of human performance, and seek to evaluate keyboard algorithms rather than theorize user behavior. Since the early 1990's, research interest has shifted from studying physical typewriters to soft keyboards. The Fittsdigraph model, first proposed by Lewis, was used to estimate average text entry speeds based on movement time between pairs of keys and digraph frequencies. This model has been a popular performance prediction tool for keyboard evaluation with different layouts using a single finger or a stylus, and has been used as an objective function for keyboard optimization. A two-thumb physical keyboard predictive model has also been proposed. Unlike these models, which predict an upper bound for average text entry speed assuming a certain error rate implied by Fitts' law (4% per target), Remulation and analysis method assesses keyboard error rates and accuracies using the same speed that had naturally occurred in the data collection experiment. Also, the focus is on evaluating keyboards with similar appearances but different algorithms, instead of different layouts.

2) *Measuring Accuracy In Text Entry:*

Standard text entry accuracy metrics compare the transcribed and presented strings. The Minimum String Distance (MSD) is often used to measure the "distance" between the two strings, based on the number of character insertions, deletions, and replacements needed to turn one string into another. While this has been effective in measuring traditional physical keyboards that literally output every letter typed, it is insufficient for measuring STKs. STKs embed a dictionary or language model and do not necessarily map touch points to text on an individual letter basis. Similarly, SGKs usually work at the word level: they recognize a continuous finger gesture to output a single word. Since both SGKs and STKs operate at word-level, it is more meaningful to adopt the word-level metrics. Furthermore, although word-level metrics (e.g., WER) are common in other fields such as speech recognition, they are rarely used to measure keyboard performance.

Wobbrock and Myers analyzed the character input stream in addition to the presented and transcribed strings. They assume that text flows serially character-by-character.This assumption does not hold for STKs, in which whole words may be algorithmically modified after they are entered. The character-level metric also does not suit SGKs well.

III. DESIGN AND ANALYSIS

A. OCTOPUS: A Remulation-Based Keyboard Evaluation System

Based on the Remulation concepts described earlier, Octopus (named after the conference room "Dr. Octopus" where the concept was first discussed) is designed and implemented. Octopus is a Remulation-based touchscreen keyboard evaluationtool. Fig. 4 shows its architecture, which consists of (1) the Simulator, (2) Dataset, and (3) the Keyboard Output Receiver.

©IJRASET 2015: All Rights are Reserved

Volume 3 Issue VI, June 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

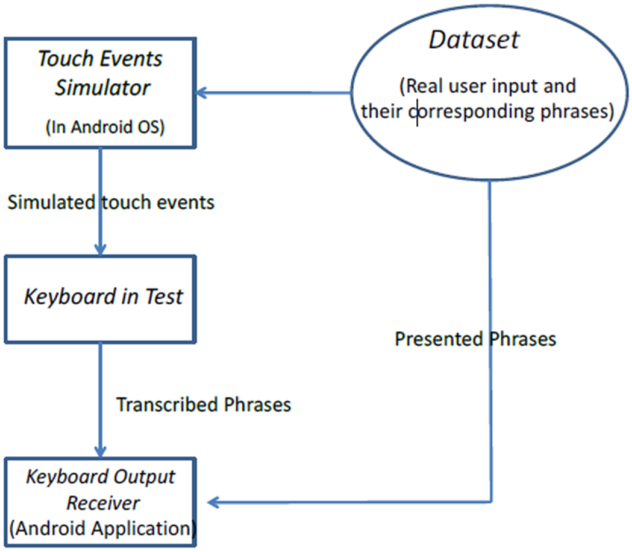


Fig. 4 The architecture of the Octopus System[1]

1) *Touch Events Simulator:* This component simulates touch actions on a mobile device in real time according to the dataset. It can simulate TOUCH_DOWN, TOUCH_MOVE, and TOUCH_UP events, which are the three basic touch operations on a mobile device. Using these events as building blocks, Octopus can simulate all the typical touch interactions. For example, a quick tap usually consists of a TOUCH_DOWN event immediately followed by a TOUCH_UP event, and a gesture usually consists of a TOUCH_DOWN event followed by several TOUCH_MOVE events and a TOUCH_UP event.The Simulator accurately specifies the interval between every two touch events. The precision of such intervals between two events is less than ±10 ms. It can also simulate multi-finger interaction by specifying the finger ID of a touch event. These features allow Octopus to keep fidelity high in simulation, which is critical for evaluating modern soft keyboards. For example, a keyboard might adjust its algorithm according to the typing speed of a user. Simulating touch actions in real time is critical to measure such algorithms. When a user quickly types with two thumbs, she might land the second thumb before lifting the first one, generating multi-touch events. Octopus enables us to investigate how a keyboard handles these situations.

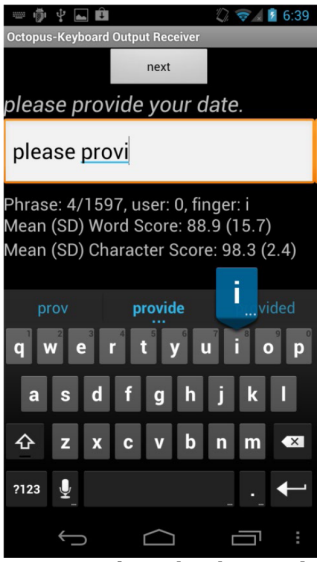


Fig. 5 The UI of Keyboard Output Receiver as Octopus *remulates* the phrase “please provide your date.” The red dot marks a simulated touch point[1]

2) *Dataset:* The dataset is fed into Octopus to simulate real users' text entry actions. The datasets consist of touch events and their corresponding phrases in *presented phrases*. Each touch event includes the event type (i.e., TOUCH_DOWN, TOUCH_MOVE, or TOUCH_UP), the (x,y) screen coordinates, the timestamp, and the finger identifier. The collected data aims to reflect fundamental human performance, independent of particular visual design elements, keyboard features, or algorithms. Also, to challenge keyboards algorithms and better discriminate different keyboards, the dataset strives to capture users' relaxed, natural and uncorrected typing behaviors. In the current implementation of Octopus, the dataset is collected through lab studies in which participants type or gesture the presented phrases on a mobile device as naturally and as quickly as possible, using a collector keyboard. The collector keyboard provides users with only asterisks as feedback when they enter text, to prevent them from adjusting their input behaviors to take advantage of certain keyboard algorithms and features (such as deleting a whole word at a time). Ideally, the layout and dimensions of the collector keyboard are identical to the test keyboard used during Remulation. If the test keyboard has slightly different dimensions, touch points can be scaled and translated according to the target keyboard's height, width, and top-left corner location. The validity of such transformations should be further empirically verified in future research, particularly when the transformation is large.

3) *Keyboard Output Receiver:* This is an application that runs on the touch screen device at the same time as the Simulator that generates the touch events. The Keyboard Output Receiver includes a standard text view widget that receives the *transcribed phrases* from the keyboard. The communication between the keyboard and the Keyboard Receiver is through the mobile OS input method framework, so Remulation can be performed on any keyboard that is installed on the device. The Keyboard Output Receiver logs the *transcribed phrases* generated by the keyboard, compare them with the *presented phrases* from original Dataset, and then calculate and display the quality measures of the keyboard being tested.

IV. IMPLEMENTATION AND METHODOLOGY

Octopus is designed to evaluate the effectiveness of keyboard algorithms. This section introduces the measures used: *Character Score*, *Word Score*, and *Ratio of Error Reduction.*

A. Character Score

Character Score is based on Minimum String Distance, (MSD), which is the smallest number of character insertions, deletions, and replacements needed to transform one string into another:

$$\text{MSD error rate}=\frac{MSD(P,T)}{MAX(|P|,|T|)}$$

where$P$ is the presented phrase and T is the transcribed phrase. To more intuitively and directly reflect accuracy, we convert the error rate to *character score*:

$$CharacterScore=(1-\text{MSD}errorrate)\times 100$$

The character score is between 0 and 100, and approximately indicates the percentage of correct characters. The higher the score, the more accurate is the keyboard. A keyboard with a score of 100 is error-free. However, note that this score depends on the test dataset. A dataset could include fundamental errors (e.g. input based on misread words from *presented text*), so a score of 100 may not be achievable.

B. Word Score

Word Score is based on Minimum Word Distance (MWD), which is the smallest number of word deletions, insertions, or replacements needed to transform one string into another.

$$\text{MWD error rate} = \frac{MWD(P,T)}{MAX(|P|,|T|)}$$

A word is defined as a string of characters entered between one or more continuous spaces. $|P|$ and $|T|$ are the lengths of the presented and transcribed phrases, measured in number of words.

Similar to Character Score, Word Score is defined as:

$$WordAccuracyScore=(1-MWDerrorrate)\times 100$$

It approximately represents the percentage of correct words for a given dataset. Like Character Score, it is also dataset-dependent.

C. Ratio of Error Reduction (RER)

©IJRASET 2015: All Rights are Reserved

Character and *Word Scores* measure the overall accuracy of a given dataset, but they do not specifically measure a keyboard's correction and recognition capabilities. Comparing transcribed to presented text does not provide information about how many of a user's errors were corrected by the touch keyboard. Fig. 6 shows an example of user touch input on a soft keyboard. The presented text is "home," but the user touches the keys "h", "o", "m", and "w." A naïve keyboard that does not attempt to correct user imprecision may output "homw" but the keyboard in the figure has successfully corrected the user's input and output the text "home." However, if we compare the presented text to the transcribed text, we do not learn about the successful error correction. If the user's input had been precise (hitting the "h," "o," "m," and "e" keys), the naïve keyboard would have produced the same transcribed text. A Word Score comparing presented text with transcribed text will give the same result for the two scenarios.

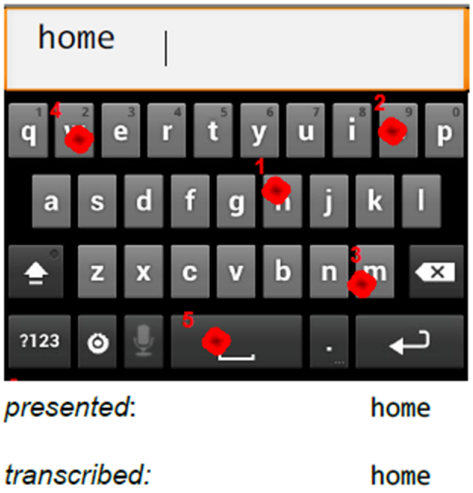


Fig. 6 A user touches the screen when entering the word "home." The touches are labeled in the order entered. The user imprecisely entered the "e," touching above the "w" key instead[1]

The baseline text is generated from the closest key labels on the keyboard to the users' actual touch points. This is a naïve key-detection algorithm that offers no error correction, and literally transcribes the user's touch points. Comparing the *baseline* text to the presented text provides information about a *user's* accuracy; comparing the *baseline* text to the transcribed text provides information about the *keyboard's* ability to correct user errors.

To quantify a touch keyboard's ability to correct user error into correct words, we propose another metric, the *Ratio of Error Reduction* (RER). This metric is defined as the proportion of errors in the baseline string that were fixed by the keyboard. The RER is calculated as follows:

$$\text{RER} = \frac{E_{baseline} - E_{transcribed}}{E_{baseline}} \times 100\%$$

where $Ebaseline$ is the error rate of the baseline text and $Etranscribed$ is the error rate of the transcribed text. *RER* is applicable only when $Ebaseline > 0$. We can use either the MSD or the MWD to express the error rates.

V. ANALYSIS

The Octopus *remulation* method is one method of keyboard evaluation. Text entry is a complex process and no single method, such as a laboratory experiment, instrumented field deployment, or modeling and prediction, can provide a complete understanding of a keyboard. The Remulation based approach offers another method that has both strengths and limitations.

A. *Strengths*

1) *Efficiency:*Empirical data can be used and reused multiple times to evaluate different keyboards. Efficiency is important for rapid algorithm iteration, since a developer can determine the impact of algorithm changes without new human participants.
2) *Fidelity:*This approach faithfully replicates participants' behavior in real time during the user study. It can precisely specify the time intervals between touch events, and also can support multi-touch input.
3) *Sensitivity*: Running Octopus is like employing the same group of participants typing over and over on multiple keyboards.

There are no confounding variables like time of day, fatigue, or learning effects, so the impact of small algorithm tweaks can be measured reliably.

4) *No source code required*: This approach offers a "black box" evaluation: one must only install a keyboard on a mobile device to evaluate it. This enables evaluation of third-party keyboards without accessing their proprietary algorithms or source code.

B. Limitations

Octopus focuses on correction and recognition algorithms of keyboards. It does not evaluate the entire user experience of a touchscreen keyboard. In particular, it does not evaluate UI-related interaction behaviors, such as selecting the target via the suggestion bar or using the backspace key. Also, it is limited to keyboards with the same layout as the one used in data collection (e.g., Qwerty).

Because the user data is collected a priori and the keyboard for data collection does not provide the user with feedback for correct or erroneous input, Octopus does not account for changes in user behavior in response to keyboard output. Octopus focuses on "open loop" typing aspects in which the user types ahead and trusts the keyboard to correct their imprecise input (be it touch or gesture). This type of data reflects the most basic and a natural input behavior, unencumbered by the UI and algorithms, but does not capture feedback-driven behavior adjustments.

VI. PROOF READING AND FUTURE WORK

A. Proof Reading

Remulation,a novel approach to evaluating keyboard correction and recognition algorithms by replicating prior user study data via real-time simulation. It contributes to the wide spectrum of user interface evaluation methods, ranging from A/B testing in laboratory experiments to model-based prediction. Two new metrics, *Word Score* and *Ratio of Error Reduction* (RER), to measure keyboard accuracy at the word level, and to quantify STK error-correction capability has been discussed.

Based on the *Remulation* approach and new data analysis methods, we have studied the design of Octopus, a keyboard evaluation tool.For example, Octopus showed that today's STKs can correct over 70% of the word errors that a naïve touchscreen keyboard would produce on the Salt dataset. Octopus also exposed different types of errors made by the tested STKs, even though standard metrics showed that the keyboards performed identically. Octopus remulation can also be applied to continuous word gesture-based keyboards, and found that one SGK is much stronger than another, which in turn suggests that this novel input paradigm may progress even further with future research.

B. Future Work

We note that the performance measures from Octopus Remulation depend on the test dataset. A dataset can be either too easy (nearly perfect input) so all keyboards can give high word scores (a ceiling effect), or too difficult (all input hopelessly sloppy and erroneous) so no keyboard can do well on it (a floor effect Octopus can be used and enhanced in several ways. We saw only two applications of Octopus for evaluating touch and gesture keyboards. With a new set of data, Octopus can be used to evaluate keyboards with different layouts on various devices, including tablets. With appropriate operating system support, the Remulation approach can also be implemented on platforms other than Android, such as Apple's iPhone and Microsoft's Windows Phone, enabling comparison across platforms.

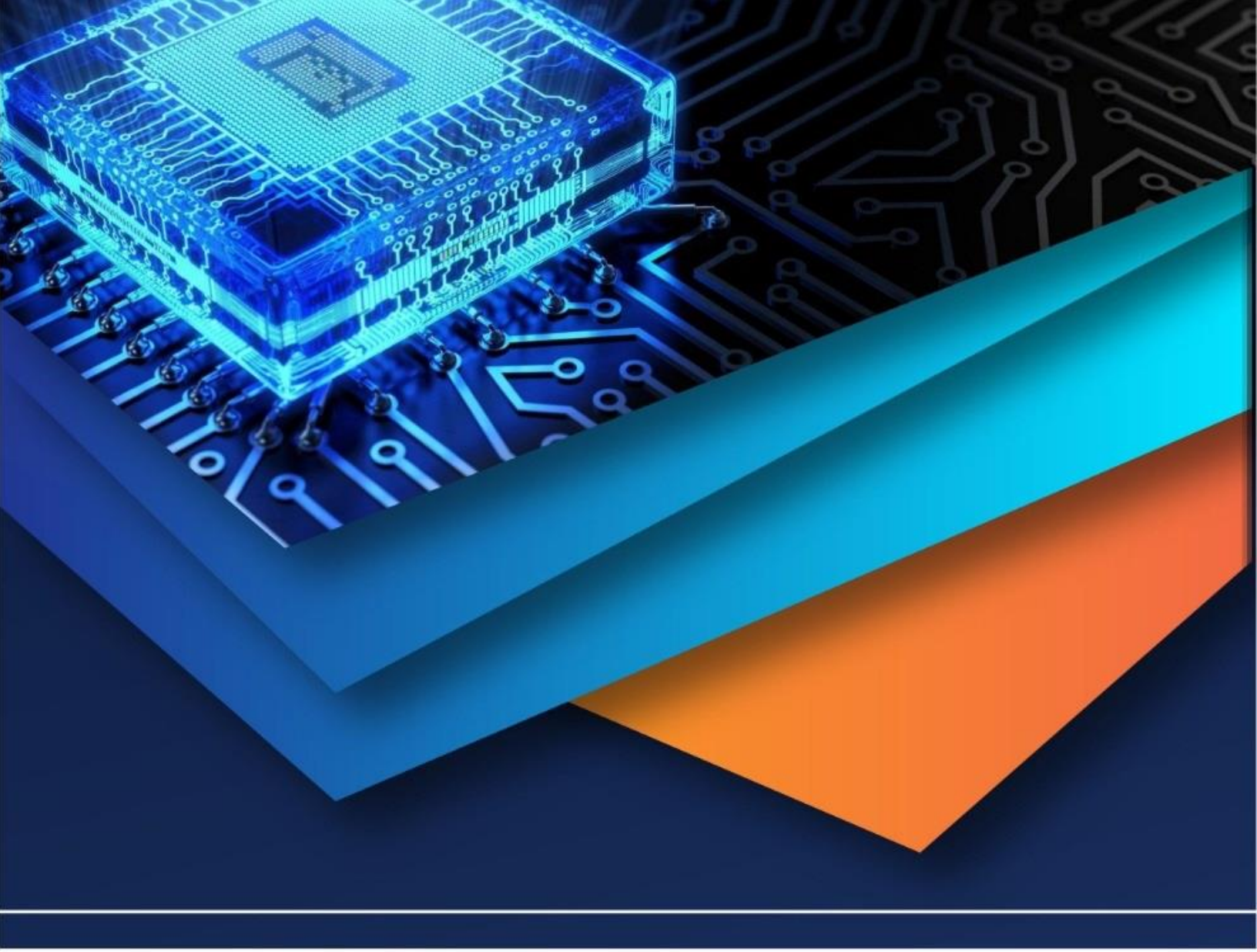
REFERENCES

[1]Xiaojun Bi1, ShiriAzenkot 2, Kurt Partridge1, ShuminZhai, "Octopus: Evaluating Touchscreen Keyboard Correction and Recognition Algorithms via "Remulation",CHI 2013, , Paris, France, April 27–May 2, 2013.

[2]Xiaojun Bi, Tom Ouyang and ShuminZhai, "Both Complete and Correct? Multi-Objective Optimization of Touchscreen Keyboard", ACM CHI 2014, Toronto, Canada, April, 2014.

[3] Cuaresmsa, J., &MacKenzie I. S.,"A study of variations of Qwerty soft keyboards for mobile phones", MHCI 2013,Ottawa, Canada,2013.

©IJRASET 2015: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)