

Type with a Swipe: Gesture Keyboard

Submitted in partial fulfilment of the requirements for the degree of

Bachelor of Technology in Computer Science and Engineering

by
SHAURYA CHOUDHARY
18BCE2113

Under the guidance of

Prof. Krishnamoorthy A
SCOPE
VIT, Vellore.



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

October, 2020

DECLARATION

I hereby declare that the report entitled “**Type with a Swipe: Gesture Keyboard**” submitted by me, for the award of the degree of *Bachelor of Technology in CSE* to VIT is a record of bonafide work carried out by me under the supervision of **Prof. Krishnamoorthy A.**

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: VIT, Vellore

Date: 18 / 10 / 2020

shaurya choudhary

Signature of the Candidate

CERTIFICATE

This is to certify that the thesis entitled “**Type with a Swipe: Gesture Keyboard**” submitted by **Shaurya Choudhary (18BCE2113)**, **SCOPE**, VIT, for the award of the degree of *Bachelor of Technology in CSE*, is a record of bonafide work carried out by him under my supervision during the period, 15. 07. 2020 to 15.10.2020, as per the VIT code of academic and research ethics. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfils the requirements and regulations of the University and in my opinion meet the necessary standards for submission.

Place: VIT, Vellore
Date: 18 / 10 / 2020

Signature of the Guide

Internal Examiner

External Examiner

Head of the Department
SCOPE

ACKNOWLEDGEMENTS

I would like to express my special thanks of gratitude to my teacher Prof. Krishnamoorthy A who gave me the golden opportunity to do this wonderful project on Human Computer Interaction, which also helped me in doing a lot of Research and I came up with the implementation of Swipe Keyboard for continuous gesture typing on virtual keyboards. The project “Type with a Swipe: Gesture Keyboard” has helped me realize the potential and real-life applications of Human Computer Interaction and I came to know about so many new things, I am really thankful to them.

Secondly, I would also like to thank my friends who helped me in making this project possible and a lot in finalizing this project within the limited time frame.

shaurya choudhary
Shaurya Choudhary

TABLE OF CONTENTS

CONTENTS	Page No.
Acknowledgement	i
Table of Contents	ii
Abstract	1
Introduction	1
Related Works	2
Proposed System	9
System Design	11
Algorithm and Implementation	12
Results	13
Conclusion	13
References	14

Abstract - Swipe typing is a stroke-based text input technique for cell phones. Rather than tapping, the client enters a word by gesturing through the entirety of its letters on a virtual keyboard with a single persistent stroke. Despite the fact that swipe typing is a broadly accessible input method for touch devices, there's very limited known about the concepts behind it. This work fills this hole by setting up a physiological development model for swipe typing. Several existing approaches are assessed and using that a simple program is implemented to see the project in action. The literature is evaluated to find further possible improvements in the domain.

1 Introduction

This project is made for the J-Component of Human Computer Interaction course, the main objective of doing this project is to implement a program for real-life problems with HCI and understand it's working. The aim of our project is to understand through research and implementation, the workings of gesture typing. Swipe typing is a stroke-based text input technique for cell phones. Rather than tapping, the client enters a word by gesturing through the entirety of its letters on a virtual keyboard with a single persistent stroke. Despite the fact that swipe typing is a broadly accessible input method for touch devices, there's very limited known about the concepts behind it. This work fills this hole by setting up a physiological development model for swipe typing. Several existing approaches are assessed and using that a simple program is implemented to see the project in action

1.1 Scope

Throughout human civilization, text has been an indispensable channel of communication. Modern computers equipped with desktop keyboards have dramatically increased the ease and volume of text-based communication in the form of email, text chat, and Web posting. As computing technologies expanded beyond the confines of the desktop, the need for effective text entry on mobile devices has been increasingly felt over the last two decades. Such a need has inspired both academic researchers and the information technology industry in pursuit of effective text entry methods alternative to the ubiquitous desktop keyboards.

Gesture keyboards allow users to enter text using continuous input strokes. They are also known as gesture typing or shape writing or swipe typing. The user enters words by sliding a finger or stylus from the first letter of a word to its last letter, lifting only between words. It generally uses error-correction algorithms and a language model to guess the intended word. It mainly has two major components that contribute to its accuracy and speed: an input path analyser, and word search engine with corresponding database.

1.2 Objective

With the increasing popularity of smartphones and other touchscreen devices, millions of people face the challenge of entering text on virtual keyboards. Although most of these keyboards resemble the main characteristics of their physical ancestors, their small size and lack of any tactile feedback contributes to high typing error rates. To reduce these rates, most virtual keyboards include error correction algorithms that run automatically while users are typing. In looking for ways to improve the efficiency of virtual keyboards, researchers have established a novel input paradigm, which is called “shape writing” or “gesture typing.” This technique is based on word gestures that subsequently trace each letter of a word on a soft keyboard. In contrast with tapping, the finger is only lifted between words. The traces that are generated during the input of a word gesture are analysed by a statistical model to guess the intended word. Nonetheless, gesture keyboards do not abandon tapping. In fact, they are conventional touch keyboards that afford gesture typing as an additional text entry paradigm. This property imposes a low adoption entry threshold and has stimulated the implementation of numerous commercial gesture keyboards such as ShapeWriter, Swype, SlideIT, Flex T9, SwiftKey Flow, and the Android 4.2+ stock keyboard.

As gesture keyboards operate on large lexicons, which may contain more than 10 000 words, replay-based evaluation is typically performed on rather small subsets of the lexicon. Testing the whole lexicon requires a large database of word gesture data. As it is impractical to collect these data by a laboratory experiment, in this study, we propose a novel technology that enables us to synthesize word gesture input.

2 Related Works

Human Computer interaction has become an inseparable important aspect in all technological segments, and input methods are one of the important components. In the market dominated by touch-enabled devices, it's matter of high priority that people are enable to give their input in an easy and fast manner. A gesture keyboard can be viewed as a conventional touch keyboard that also affords gestures. The user does not have to have learned any gestures before using a word-gesture keyboard. As a beginner, the user simply slides the finger from one letter to another, driven by visual guidance to the next letter key on the keyboard.

Different research journals are surveyed to understand this domain with its inner workings and trade-offs. The details about the same are discussed below.

Research Journal	Title (Study)	Year of Publication	Problem Addressed
Paper - 1	Synthetic Word Gesture Generation for Stroke-Based Virtual Keyboards	2017	Introduce a framework capable of synthesizing the trajectories of word gestures that share many features with human-generated gestures. We use these synthetic gestures to automatically evaluate gesture recognition algorithms with thousands of gestures.
Paper - 2	Inferring Text Entered Through Gesture Typing on Android Keyboards	2016	A new way in which systemwide resources can be a threat to user privacy. We investigate the effect of rate limiting as a countermeasure but find that determining a proper rate is error-prone and fails in subtle cases. We conclude that real-time interrupt information should be made inaccessible, perhaps via a tighter SELinux policy in the next Android version.
Paper - 3	The Word-Gesture Keyboard: Reimagining Keyboard Interaction	2012	This article summarizes a decade-long academic research that led to the establishment of this input paradigm. We developed the basic concepts and initial prototype of a word-gesture keyboard
Paper - 4	SHARK2: A Large Vocabulary Shorthand Writing System for Pen-based Computers	2014	Based on the use characteristics and human performance observations, we designed and implemented the architecture, algorithms and interfaces of a high-capacity multi-channel pen-gesture recognition system.
Paper - 5	Tap and Gesture Typing on a Smartwatch Miniature Keyboard with Statistical Decoding	2016	A finger operated keyboard that supports both touch and gesture typing with statistical decoding on a smartwatch. Just like on modern smartphones, users type one letter per tap or one word per gesture stroke on WatchWriter but in a much smaller spatial scale.
Paper - 6	Optimizing Touchscreen Keyboards for Gesture Typing	2015	This paper systematically investigates the optimization space related to the accuracy, speed, and Qwerty similarity of a gesture typing keyboard. Our investigation shows that optimizing the layout for gesture clarity (a metric measuring how unique word gestures are on a keyboard) drastically improves the accuracy of gesture typing.
Paper - 7	Performance and User Experience of Touchscreen and Gesture Keyboards in a Lab Setting and in the Wild	2015	study the performance and user experience of two popular mainstream mobile text entry methods: the Smart Touch Keyboard (STK) and the Smart Gesture Keyboard (SGK).

2.1 Synthetic Gesture Generation

Gesture typing is a stroke-based text input method for mobile devices. Instead of tapping, the user enters a word by gesturing through all its letters on a virtual keyboard with a single continuous stroke. It shows that many features of word gestures can be described by the mathematical framework of a handwriting model. It extends the model to respect the particularities of gesture typing and show that the exact trajectories as well as the velocity profiles of most word gestures can be represented with a high accuracy. It presents an algorithm to extract the model parameters from real.

Finally, the paper introduces a framework capable of synthesizing the trajectories of word gestures that share many features with human-generated gestures. We use these synthetic gestures to automatically evaluate gesture recognition algorithms with thousands of gestures.

There are different approaches to gesture synthesis. One possibility is to build artificial gestures from existing data. For example, if we have a $\Sigma\Lambda$ model of a word, we can modify its parameters to get new synthetic gestures for the same word. We can also construct new gestures by interpolating between two models. Other methods involve averaging existing gestures. This can be achieved with the $\Sigma\Lambda$ model or directly on the raw data using a dynamic time warping-based averaging algorithm such as barycentric averaging. All these approaches require baseline data for the gesture they intend to generate. Gestures of previously unknown words cannot be synthesized, which is a huge restriction for many applications. For this reason, we take a different approach that enables us to generate word gestures for arbitrary words and for any given keyboard layout. The key idea is to build unknown gestures from scratch by exploiting regularities of actual gestures.

Generally, our procedure can be divided into two parts: action plan and velocity profile generation. In the former part, an action plan is constructed for a given word and the geometry of the keyboard layout. In the latter part, a velocity profile is generated from the action plan. The two steps are eventually put together in terms of the $\Sigma\Lambda$ equation.

$$\mathbf{r}(t) = \sum_j \int_0^t \mathbf{v}_j(\tau) d\tau.$$

The features that we used for comparison can be divided into two groups. The first group contains features that relate to the velocity profile of a gesture. They characterize the dynamics of the movement. These features are “global” in the sense that they do not depend on the exact location or shape of a gesture. In contrast, the second group contains spatial features such as start, end, or bounding box area. These features characterize the spatial deviations of the trajectory from the corresponding template. Most of these features strongly depend on the corresponding word and therefore cannot be directly compared globally. Hence, we transform the absolute positions into relative positions with respect to the corresponding value of the gesture template. For instance, we replace the start point of the gesture with the offset to the start point of the template. In total, we use 17 different features.

To devise a suitable generation algorithm, we have analysed the correlation of many properties of genuine word gestures. The strongest and most significant relationships between the normative gesture template and the genuine gestures were incorporated into our algorithms. We assessed the generation method by comparing generated gestures with gestures from a database containing more than 30 000 gestures. The results indicate that the generated gestures share many properties with genuine gestures. Still, human generated gestures are subject to large individual differences and noise. Therefore, the generation algorithm does not cover the whole scope of human behaviour. We extended the proposed algorithm to make the sloppiness and other spatial features such as the smoothness at the corners of the normative gesture template accessible for the researcher. As an application example, we used this improved algorithm to evaluate the sensitivity of three gesture keyboards to the spatial precision of word gestures. The results indicate that the precision at the first and last letters is crucial for correct predictions. In contrast, sloppiness at the corners is tolerated to a higher degree.

2.2 Keyboard Interaction revolution

At some level, it is relatively easy to invent a new text entry method. After all, a text input method is a coding system for text communication. There can be potentially an infinite number of possible ways to code text by spatiotemporal means, including Morse code and the great many diverse writing systems of the world. However, to develop a mobile text entry method truly acceptable to the mass consumer market is exceptionally difficult for many reasons. First, since text input is one of the most intensive and frequent human-computer interaction (HCI) tasks, speed is a very important consideration. Users are accustomed to fast keyboard typing on their own desktop or laptop keyboard.²⁰ A mobile text input method is ideally as fast as a desktop keyboard, or at least fast enough so the users do not have to defer text writing to a non-mobile setting. Second, in order to gain wide adoption, a text entry method must impose minimal cognitive load on new users. This means that little or no learning should be required for users to start using a new text entry method. Most computer users have already invested time and effort in learning typing on Qwerty keyboard. A new method that requires even a fraction of that investment upfront is difficult for mass adoption. Third, a successful new text

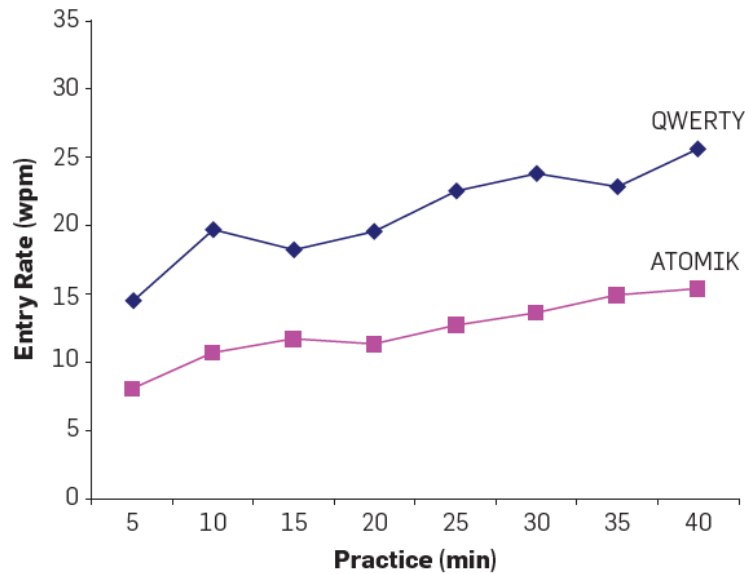
input method should support development of proficiency – the ability to have performance improvement toward higher efficiency through practice in use. Unfortunately, ease of adoption and efficiency in user interface design are often at odds with each other due to having different cognitive foundations. The alternatives too often reduce to determining where the load resides: easy to start but inefficient ever-after, or hard to learn but highly efficient as (hard-won) skill is acquired.

One continuous movement: In comparison to tapping-based touchscreen keyboards, gesture keyboards do not require up and down movements for each letter. Instead an entire word involves only one continuous movement. Anecdotal evidence from centuries of stenography research has pointed out the impeding effect on speed performance of repeated lifts. From everyday writing, we also know that when we write fast, we write cursive—meaning multiple letters are linked as one continuous stroke. To a degree, the word gestures on a gesture keyboard in effect become a modern form of shorthand for words, akin to European shorthand systems. Note that minimizing the number of separate actions was the main motivation in Montgomery’s wipe-activated keyboard and single-stroke shorthand for characters, such as Unistrokes, Graffiti, and their Roman antecedent, Notae Tironianae, developed by a slave of Cicero, Marcus Tullius, in 63 BC. The speed advantage of a single-stroke word gesture input, as opposed to single-finger (or stylus) tapping of individual letters of the same word, can also be understood in motor control modelling terms. Tapping individual letters in a word can be viewed as a sequence of discrete target pointing tasks, each can be modelled by Fitts’ law

$$t_{k,k+1} = a + b ID$$

$$ID = \log_2 \left(\frac{D_{k,k+1}}{S} + 1 \right)$$

Typing competition was a common method of demonstrating typewriter quality in the mechanical typewriter days. Typing competition’s results are often affected by the rules and context of the competition, but nonetheless the record setting method reveals the top range of performance possible with a given text input method. Similarly, the peak error free one sentence speed that can be achieved with a given input method reveals one aspect of the method’s potential.



The word-shorthand gesture keyboard project has produced a wide range of results from which we attempt to piece together a coherent but simplified account in this article. Throughout the project, we tried to bridge invention with science, practical product design and development with theory-driven research, and application of modern computing techniques with human performance insights and modelling. We drew inspirations from theoretical HCI thoughts in, for example, Buxton's work on user learning. We frequently applied methods, models or at least the spirit of a school of thought in HCI spearheaded by the classic monograph of Card, Moran and Newell. This school of thought bases human-computer interaction design on psychological insights embodied in approximate human behaviour and performance regularities, rules, equations and models. We also exploited to a degree we could the power of statistical approaches to information processing rooted in classic information theory,³⁶ but enabled and modernized as computational power increases to a level on mobile devices impossible only a few years ago.

2.3 Gesture Keyboard for Smartwatch

WatchWriter, a finger operated keyboard that supports both touch and gesture typing with statistical decoding on a smartwatch. Just like on modern smartphones, users type one letter per tap or one word per gesture stroke on WatchWriter but in a much smaller spatial scale. Watch-Writer demonstrates that human motor control adaptability, coupled with modern statistical decoding and error correction technologies developed for smartphones, can enable a surprisingly effective typing performance despite the small watch size. In a user performance experiment entirely run on a smartwatch, 36 participants reached a speed of 22–24 WPM with near zero error rate.

The primary challenge to effective manual text input on a smartwatch is the relatively small display size. A normal watch face's width measures two to three human fingers. For a regular Qwerty keyboard whose top row has 10 letter keys, the landing finger may cover three or more key spaces.

Like on many smartphone keyboards, WatchWriter features a suggestion bar that presents the user with two possible recognitions for the word that they are currently composing. The entries in the suggestion bar automatically update upon each key press, and can show either corrections (e.g., from “tje” to “the”) or prefix-completions (e.g., from “tomo” to “tomorrow”). In order to commit the current word and move on to the next one, the user taps on one of these suggestions (in gesture typing, the user may also simply perform their next word’s gesture and the best suggestion for the current word is automatically committed). These suggestion taps also automatically insert a space after the committed word, eliminating the need for a dedicated spacebar key and saving valuable Fat Fingers, Small Watches #chi4good, CHI 2016, San Jose, CA, USA 3818 screen space. Another advantage of requiring deliberate suggestion taps to commit words is that users are much more aware of what gets inserted into the text view. It removes the well-known danger of having a spacebar tap auto-correct to an unintended word without the user noticing, leading to the types of humorous autocorrect “fails” posted to websites like www.damnyouautocorrect.com. In gesture typing, the choices on the suggestion bar are the keyboard’s two best predictions given input and context. In tap typing, the bold suggestion on the left side of the suggestion bar contains the most likely prediction as decided by the decoder, and the right suggestion contains the literal string. If the most likely prediction matches the literal string, then the left side of the suggestion bar displays the second most likely prediction instead. Upon tapping an item in the suggestion bar, that text is then added to the running output string above, and the suggestion bar is cleared in preparation for a new word.

The user study consisted of two separate tests, one for gesture typing and one for tap typing. In Test 1, 18 participants gesture typed a fixed set of 21 phrases on four keyboard layouts: Qwerty and three other non-Qwerty layouts. In Test 2, another 18 participants tap typed the same set of 21 phrases on three layouts: Qwerty and two other non-Qwerty layouts. In both tests the order of layouts were balanced by a Latin Square design. As stated earlier, the alternative layouts are of no interest to the current paper. Each participant completed the study in around 45 minutes in total.

The 21 phrases were selected at random from MacKenzie and Soukoreff’s list of phrases for evaluating text entry techniques due to their memorability. These phrases consisted of common English-language words. Participants were instructed to correct their entry errors as best they could.

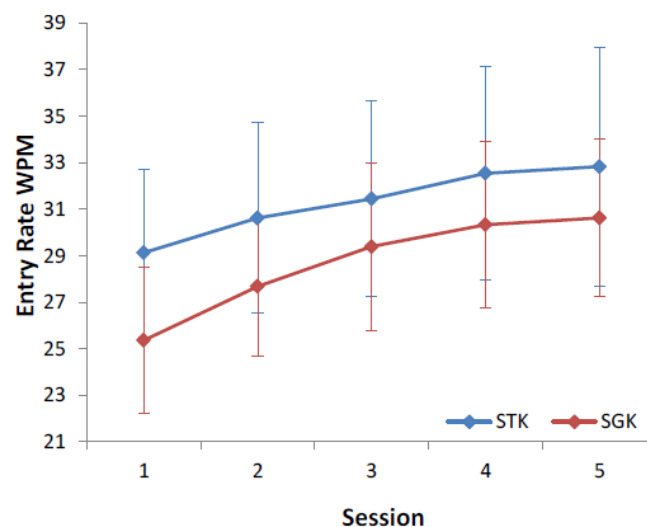
WatchWriter demonstrated that the single-step tap and gesture typing can be quite effective even on a small watch form factor, thanks to the combination of auto-correction and prediction capabilities already developed on smartphones and the human motor control flexibility and adaptability.

2.4 Performance Comparison between STK and SGK

The paper studies the performance and user experience of two popular mainstream mobile text entry methods: The Smart Touch Keyboard (STK) and the Smart Gesture Keyboard (SGK). Our first study is a lab-based ten-session text entry experiment. In our second study we use a new text entry evaluation methodology based on the experience sampling method (ESM). In the ESM study, participants installed an Android app on their own mobile phones that periodically sampled their text entry performance and user experience amid their everyday activities for four weeks. The studies show that text can be entered at an average speed of 28 to 39 WPM, depending on the method and the user’s experience, with 1.0% to 3.6%-character error rates remaining. Error rates of touchscreen input, particularly with SGK, are a major

challenge; and reducing out of vocabulary errors is particularly important. Both SGK and STK have strengths, weaknesses, and different individual awareness and preferences. Two-thumb touch typing in a focused setting is particularly effective on STK, whereas one-handed SGK typing with the thumb is particularly effective in more mobile situations. When exposed to both, users tend to migrate from STK to SGK. We also conclude that studies in the lab and in the wild can both be informative to reveal different aspects of keyboard experience, but used in conjunction is more reliable in comprehensively assessing input technologies of current and future generations.

In total we collected 100 hours of data (50 minutes of writing per session (excluding breaks) \times 120 sessions). Using STK, participants entered an average of 1393 sentences (SD = 275) during each session totalling 13,927 data points. 211 of these were filtered out as outliers since they were determined to be more than three standard deviations away from the mean. Using SGK, participants entered an average of 1,282 sentences per session (SD = 225), which totalled 12,816 data points; out of which 278 points were discarded as outliers. All statistical analyses were done using repeated-measures analysis of variance at significance level $\alpha = 0.05$. Bonferroni corrections were used to adjust the significance levels for post-hoc analyses. We report the majority of the statistical results in tables. In the tables m is the sample mean, S1 is the first session in a condition, S5 is the 5th (last) session in a condition, 95% CI means the 95% confidence interval (Z-scores).



Smart text input on mobile and touchscreen devices has been an active area of innovation both in the research literature and in the commercial world. However, as we noted in the introduction and related work, empirical research has been limited in scope, size, and technology form factor. Most reported text entry research has also been based on research prototypes. This paper reports the results of two systematic studies, in the largest scale in text input that we are aware of, using a widely deployed, publicly available product with both STK and SGK capabilities from the same developer, establishing a set of empirical findings useful for further advancement of the field. First, we found that text can be entered at an average speed of 28 to 39 WPM, depending on the method and the user's experience, with 1.0% to 3.6%-character error rates remaining. Second, error rates of touchscreen input, particularly with SGK, are still quite high; further advancements in the field need to focus on error tolerance and error correction. Reducing OOV errors are particularly important. Third, SGK and How Fast Can

You Type on your Phone? CHI 2015, Crossings, Seoul, Korea STK both have strengths, weaknesses, and different individual awareness and preferences. Two-thumb touch typing in a focused setting seems particularly effective on STK, whereas one handed SGK typing with the thumb seems particularly effective in more mobile situations. This research shows that when exposed to SGK, users tend to migrate from STK to SGK. This constitutes perhaps the strongest empirical evidence of SGK's strength. Fourth, research methodology matters; studies in the lab and in the wild both can be informative to different aspects of keyboard experience, but used in conjunction is more reliable in comprehensively assessing input technologies of current and future generations.

3 Proposed System

SHARK stands for SHorthand Aided Rapid Keyboarding. SHARK2 is successor of original SHARK algorithm which enabled gesturing on a stylus keyboard for familiar words but required tapping for others. The algorithm is designed for Stylus-Keyboards (SK) also known as virtual keyboards. SHARK2 algorithm eliminates the need to switch between these 2 modes and allow swipe typing for almost all words. Each pattern of a word is formed by the trajectory through all of the letters of the word on the virtual keyboard, from the first to the last in order. These patterns are called sokgraph (SK + Graph).

The SHARK2 stands out from other algorithms as:

- The number of sokgraph (slide patterns) gestures each individual user may need is much greater than the gesture repertoire of an alphanumeric recognizer such as Graffiti or Jot.
- Unlike natural (longhand) cursive handwriting recognition, a fraction of the sokgraph shorthand gestures, particularly those for short words, can be identical or similar in shape; therefore, shape alone may not provide sufficient information to recognize the user's intent.
- A set of sokgraphs defined on a keyboard layout constitutes a symbolic system novel to the user.
 - It is an advantage because each sokgraph has a unique ideal prototype, in contrast to natural hand writing in which even the same letter can be written in perfectly legitimate but very different styles.
 - It is also a disadvantage because there is not a natural corpus of sokgraphs that can be collected, precluding many of the standard data-driven machine learning approaches to recognition

The algorithm uses different functions to extract the Input gesture trace and match it with the template and return the most probable word after matching from the dataset. The main methods of the proposed models are:

3.1 Template pruning

- An initial pruning component first filters out a large number of the sokgraph templates from entering later stage recognition channels.

- We compute the start-to-start and end-to-end distances between a sokgraph template and the normalized unknown input gesture.
- If either of the two distances is greater than a set threshold, the template will be discarded.

3.2 Shape channel recognition

- The proportional shape matching distance between an unknown pattern u and a template pattern t is used for classification of sokgraph.
- The final result of the shape channel is an approximate scale and translation invariant distance measure of the similarity between the patterns.
- It is based on the average sum of the equidistant sample points' spatial distance.

3.3 Location recognition

- Location information provides an increased recognition resolution of sokgraphs.
- Location is part of the user's memory of a sokgraph and therefore will be reproduced during gesture production.
- The sokgraph shape recognition produces several sokgraph confusion and location plays an important role to distinguish among the lexicon.
- We use a function that gives the lowest weight to the middle point, and the rest of the points' weights increase linearly towards the two ends.

3.4 Dynamic channel weighting

- Adjusting the relative weight of the shape and location channels according to the speed of the input gesture production gives more versatility and accuracy to the model.
 - If user tracks sokgraph from word to word, the location channel will yield high score.
 - If user rapidly swipes in loop between start and end letters, shape channel will yield high score.
- The dynamic weighting is primarily based on the gesturing speed.
- It modifies the location channel probability of each individual word according to its path on the keyboard.

4 System Design

Each channel does not necessarily have enough discriminative power, but the collective information from the multiple channels can separate the sokgraphs sufficiently. The two core channels are a shape recognizer and a location recognizer. The former classifies a pen gesture according to the normalized (scale and translation invariant) shape of the pen gesture. The latter classifies a pen gesture according to the absolute location of the gesture on the keyboard. Both the shape channel and the location channel draw their recognition templates from a lexicon. The SHARK2 paradigm requires the lexicon to include all (but just enough) words a particular user needs in regular writing. This lexicon can be constructed with various methods. It can be

a preloaded standard dictionary, or a list of words extracted from the user's previously written documents, including emails and articles, or words added by the user. In practice it is a combination of all.

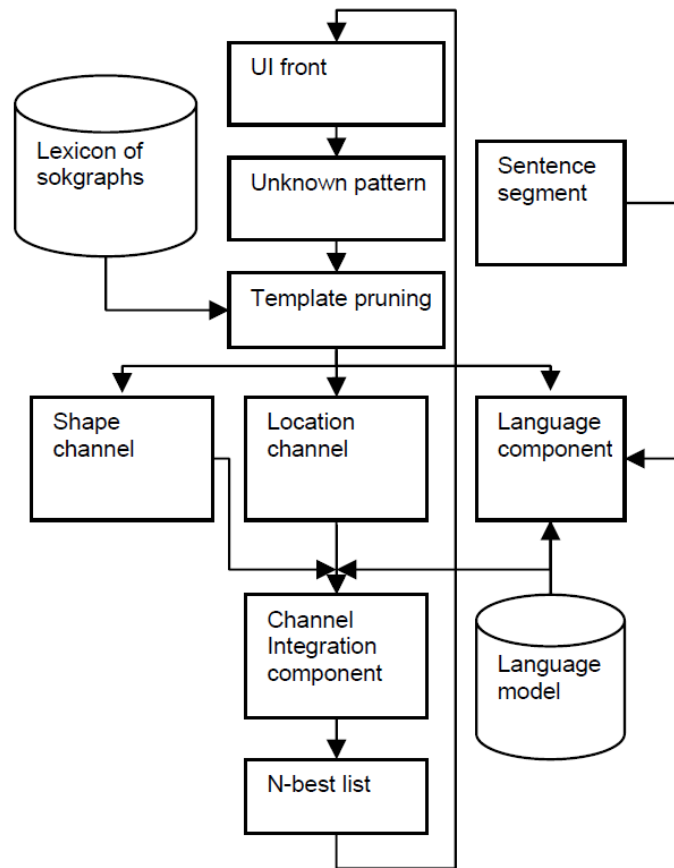


FIG. Generic SHARK2 Algorithm System Architecture

5 Algorithm and Implementation

5.1 Algorithm with modules

1. The dataset is extracted into 4 arrays: Words, probability, X and Y template points respectively. The template points correspond to the alphabets' co-ordinates in the virtual keyboard.
2. The backed processing of sokgraph is done and matched again the derived arrays to ger N-best words.
3. `generate_sample_points()`:
 - Converts every gesture to a set of 100 points for comparison of input sokgraph and template.
 - Returns 2 arrays of sampled points for both X and Y co-ordinates.
 - NumPy functions are used to calculate distance among the consecutive elements of the template.
 - Sample points are generated for every template in the dataset in prior.
 - Interpolation is used to generate middle points using SciPy.
4. `do_pruning()`:

- Compares the length of the corresponding word from the dictionary. If the length is less than or equal to 4, then it only compares the start to start and end to end distances of the template point with a certain threshold value.
 - If the length is greater than 4, it also considers some intermediate points among the sampled ones for comparison so as to exclude unnecessary words after pruning.
5. `get_shape_scores()`:
 - The total number of words in the pruned list is pretty less as compared to original list which helps in speeding up the code.
 - The function generates shape score for every valid words after pruning.
 - The sampled input gesture is compared against valid template and assigned a score for each comparison.
 - The computed scores are normalized and stored in an array.
 6. `get_location_scores()`:
 - The location scores are also computed through comparison against valid sampled inputs.
 - The distance of trace input from template is calculated using hypot function from NumPy.
 - The template trace lines are defined by connecting the centroids of the alphabets constituting the word.
 - Both input and template trace are re-sampled to N-points.
 7. `get_integration_scores()`:
 - The contribution of shape and location scores is managed by defining the score coefficients.
 - The balance between the scores is managed and then the normalized contribution of each score is summed up to give final integration score.
 - The final scores are stored in an array for all the possible valid words.
 8. `get_best_word()`:
 - Using NumPy we find the sorted index of the integration scores.
 - The index is matched with valid words array and the word is returned as the best match for the recorded sokgraph.
 - Further, next 3 matching words from the sorted index are also returned as suggestion.
 9. The app is designed on Flask framework to run on web and the interactive virtual keyboard is hosted to a browser.
 10. The implementation result is attached in further slide.

5.2 System Requirements

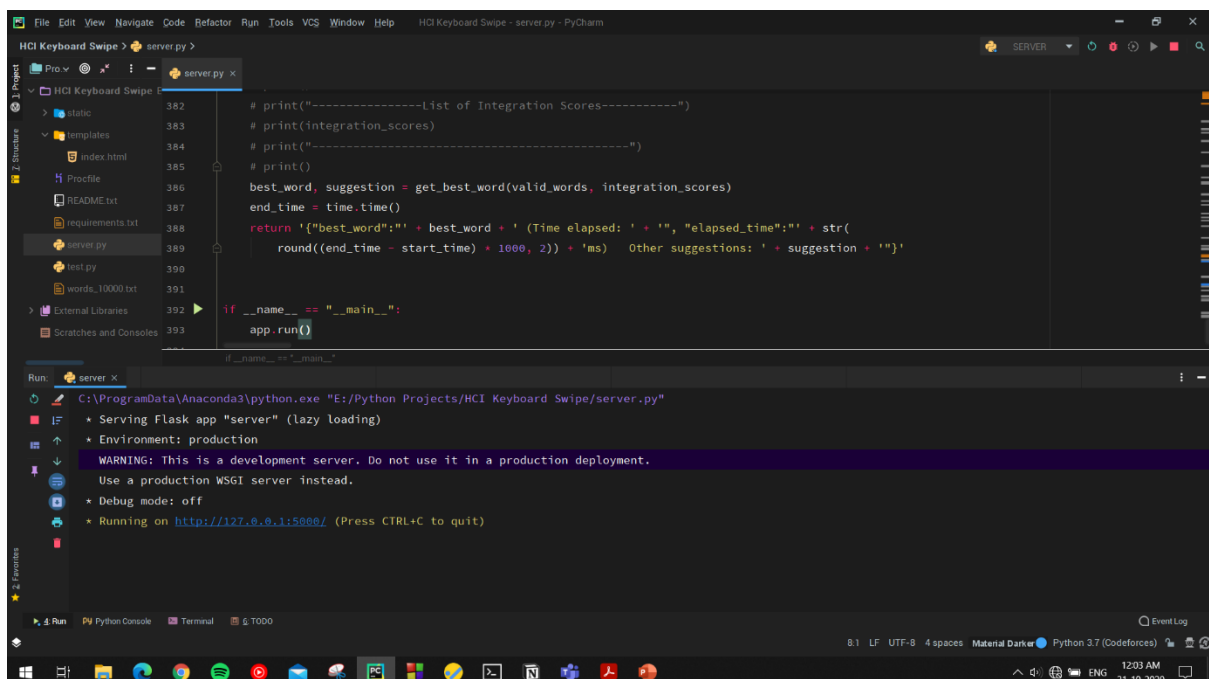
1. Every instance of word entry, except adding new words to the lexicon by tapping, has to be realized by sokgraph recognition. To the system there is no distinction between visually guided tracing and recall-based gesturing. The recognition system has to be able to recognize all words an individual user may use in regular writing. The size of the lexicon should be in the order of 10,000 words.
2. The system has to be extensible to new words a user may adopt in writing.

3. The system has to be “real time”. The recognition latency can only be a small fraction of the entire duration of writing a word.
4. The system should be compatible with the SHARK2 paradigm, supporting gradual transition from visually guided tracing to recall driven gesturing.
5. The system should give the user the maximum amount of flexibility and least amount ambiguity.

5.3 Libraries and Dataset

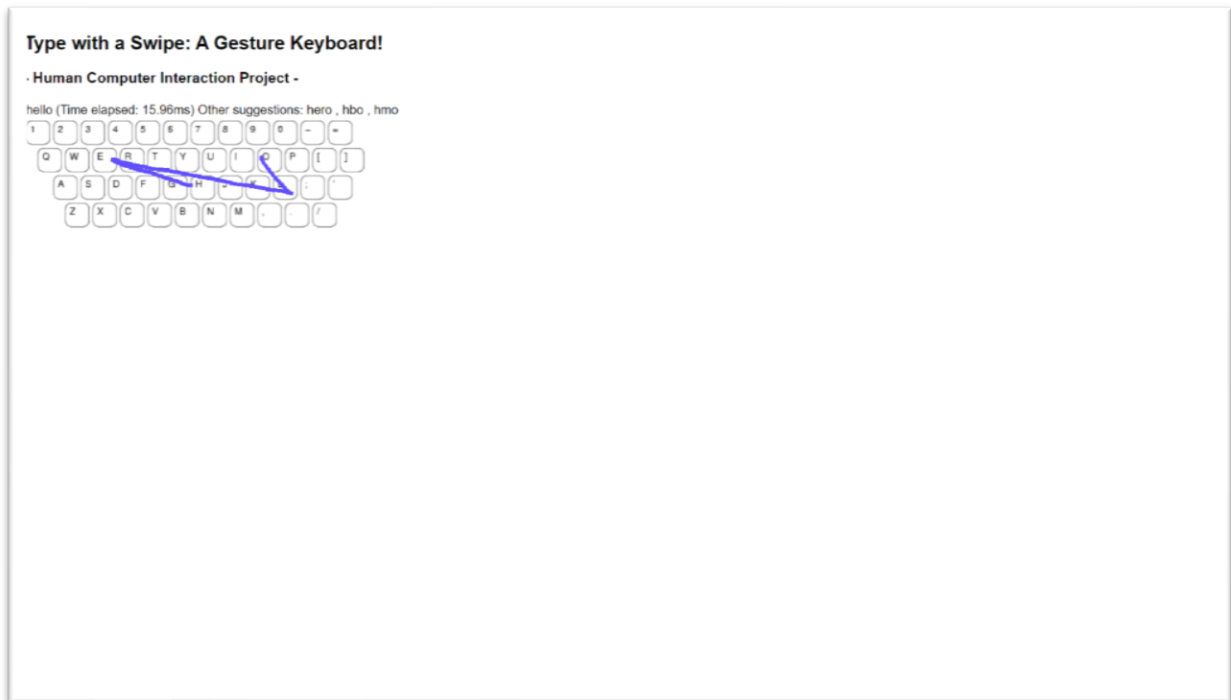
- Programming Languages:
 - Python
 - JavaScript
 - HTML + CSS
- Libraries imported:
 - Flask
 - SciPy
 - NumPy
- Dataset Used:
 - WORDS_10000

5.4 Output



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help HCI Keyboard Swipe - server.py - PyCharm
HCI Keyboard Swipe > server.py
server.py x
HCI Keyboard Swipe
  static
  templates
  index.html
  Profile
  README.txt
  requirements.txt
  server.py
  test.py
  words_10000.txt
  External Libraries
  Scratches and Consoles
382 # print("-----List of Integration Scores-----")
383 # print(integration_scores)
384 # print("-----")
385 # print()
386 best_word, suggestion = get_best_word(valid_words, integration_scores)
387 end_time = time.time()
388 return '{"best_word":"' + best_word + ' (Time elapsed: ' + ' ', "elapsed_time":"' + str(
389     round((end_time - start_time) * 1000, 2)) + 'ms) Other suggestions: ' + suggestion + '}"'
390
391
392 if __name__ == "__main__":
393     app.run()
```

```
Run: server x
C:\ProgramData\Anaconda3\python.exe "E:/Python Projects/HCI Keyboard Swipe/server.py"
* Serving Flask app "server" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```



6 Results

The implementation is hosted as a Web Application on localhost. Using Flask framework on Python as backend and JavaScript along with HTML and CSS as front end. The application was successful in demonstration of gesture typing with great accuracy. The virtual keyboard was able to identify all the words available in the dataset within fractions of second along with 3 suggested words which proved the feasibility for real-time implementation. The addition of enhancement in the pruning function on SHARK2 algorithm resulted in more filtered dataset and provided better performance. By incorporating a lexicon and a language model, the system maximizes input flexibility for the user. The system also has various feedback and output interfaces to allow the user to understand its mechanism and correct unintended text.

7 Conclusion & Future Work

In this project report, we have gone through several journals to understand the domain of Gesture Typing in depth. After gaining knowledge about the underlying concepts of the same and all the parameters and variables in play, we tried to implement already existing SHARK2 algorithm with some tweaks to enhance pruning and thus gain better performance.

It is interesting to compare SHARK2 with traditional pen-based stenography with respect to speed and overall ease of use. SHARK2 creates partially scale and translation invariant sokgraphs in a large vocabulary, similar to the word level shorthand symbols common in classic pen-based stenography for the most common words. However, in classic stenography a large part of the labour in high speed text writing is in transcribing the symbols to longhand text. In SHARK2 transcription is achieved automatically. From an information theory point of view SHARK2 takes advantage of the information redundancy in a lexicon and a language model to relax the requirement of precisely specifying words verbatim. To write a word the user only

has to make enough an effort to express the intention by the approximate shape and location of the word's sokgraph. The error tolerance in the system is inversely proportional to the size of the lexicon used.

For future works, the core technology of a word-gesture keyboard can conceivably be improved by using larger and long-span language models that take into account several previous words of context when they compute the language model's prior belief in a word candidate. Gesture keyboards can also be used with other modalities. For example, if gestures can be effectively delimited, they may be incorporated into eye-tracking systems or 3D full-body motion tracking systems, such as those used in Microsoft game products. Gesture keyboards can also be potentially integrated with speech input.

8 Reference

- [1] Simon, L., Xu, W., & Anderson, R. (2016). Don't interrupt me while I type: Inferring text entered through gesture typing on Android keyboards. *Proceedings on Privacy Enhancing Technologies*, 2016(3), 136-154.
- [2] Zhai, S., & Kristensson, P. O. (2012). The word-gesture keyboard: reimagining keyboard interaction. *Communications of the ACM*, 55(9), 91-101.
- [3] Burgbacher, U., & Hinrichs, K. (2016). Synthetic Word Gesture Generation for Stroke-Based Virtual Keyboards. *IEEE transactions on human-machine systems*, 47(2), 221-234.
- [4] <https://www.extremetech.com/extreme/97837-how-does-swype-really-work#:~:text=The%20cornerstone%20of%20the%20Swype,be%20achieved%20with%20minimal%20practice.>
- [5] <https://www.thenationalnews.com/arts-culture/what-is-gesture-typing-the-time-saving-texting-technique-we-might-soon-all-be-using-1.874986#:~:text=To%20achieve%20this%2C%20Basit%20us,lifting%20only%20between%20each%20word.>
- [6] <https://www.shuminzhai.com/post/a-guided-tour-of-my-recent-papers-on-keyboard-research>
- [7] Kristensen, P. O., & Zhai, S. (2004, October). SHARK2: a large vocabulary shorthand writing system for pen-based computers. In *Proceedings of the 17th annual ACM symposium on User interface software and technology* (pp. 43-52).
- [8] Royal, S., Zhai, S., & Kristensson, P. O. (2015, April). Performance and user experience of touchscreen and gesture keyboards in a lab setting and in the wild. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (pp. 679-688).
- [9] Smith, B. A., Bi, X., & Zhai, S. (2015, April). Optimizing touchscreen keyboards for gesture typing. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (pp. 3365-3374).

- [10] Gordon, M., Ouyang, T., & Zhai, S. (2016, May). WatchWriter: Tap and gesture typing on a smartwatch miniature keyboard with statistical decoding. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (pp. 3817-3821).
- [11] https://en.wikipedia.org/wiki/Virtual_keyboard
- [12] http://uist.acm.org/archive/adjunct/2004/pdf/doctoral_consortium/dc3-kristensson.pdf
- [13] <https://en.wikipedia.org/wiki/Swype>
-