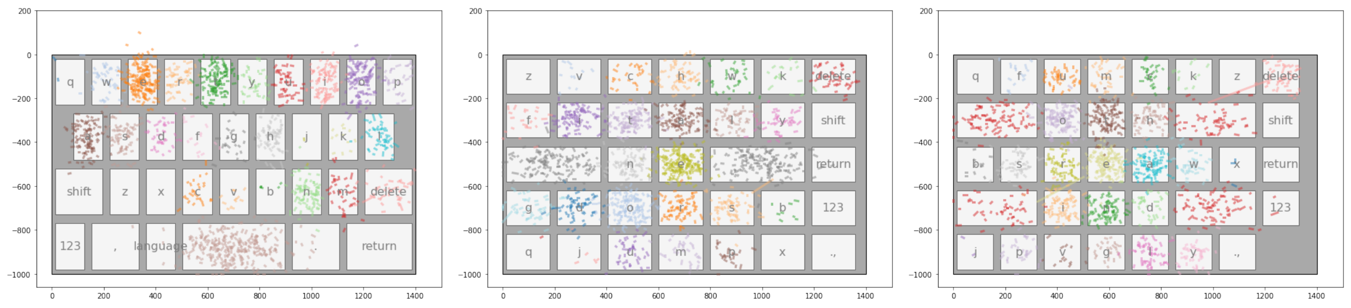
RoboType: Realistic Mobile Text Entry Evaluations with Synthetic Users

Andreas Komninos
akomninos@ceid.upatras.gr
University of Patras
Rio, Greece



(a) QWERTY **(b) FITALY** **(c) OPTI-II**

Figure 1: Example of synthetic participants output during transcription tasks with various keyboard layouts.

ABSTRACT

Mobile text entry research is contingent on user-based evaluations, which are frequently carried out in small-scale lab experiments. Recruitment and execution of experiments are costly in terms of time and effort required. Further, the results are hard to generalise, because of the small number of participants in the experiments. Studies are sometimes replicable, but not reproducible. RoboType is a realistic open-source simulator for mobile text entry, written in Python and based on the consolidation of state-of-the-art understanding of human behaviour during entry tasks. It aims to aid researchers to produce fast and accurate evaluations of mobile text entry ideas with fully reproducible results. It can be used to significantly reduce time required to explore the design space of new prototypes on various user bases, or as a benchmark to compare results with other researchers. This paper presents an early implementation of RoboType and demonstrate its promising potential.

CCS CONCEPTS

• **Human-centered computing → Ubiquitous and mobile computing systems and tools**; **Text input**; **Touch screens**; **Ubiquitous and mobile computing design and evaluation methods**.

KEYWORDS

mobile text entry, touchscreens, evaluations, modelling, simulation

ACM Reference Format:
Andreas Komninos. 2023. RoboType: Realistic Mobile Text Entry Evaluations with Synthetic Users. In *25th International Conference on Mobile Human-Computer Interaction (MobileHCI '23 Companion), September 26–29, 2023, Athens, Greece.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3565066.3608693

1 INTRODUCTION

Fast and accurate mobile text entry is critical for the usability of all services running on modern smartphones, tablets and smart watches. Novel ideas and prototypes can be designed either through inspiration or through participatory design methods, while computational design has been gaining ground as a prototype development method in recent years (e.g. [8, 12]). To evaluate prototypes, researchers typically employ lab-based methods centered around the transcription task, where participants have to copy phrases presented to them by the researchers, as fast and as accurately as possible. Transcription tasks are thus considered the de-facto evaluation method for mobile text entry [21]. Less frequently, prototypes are evaluated in field studies (e.g. [13, 22, 26]). In the field of Mobile Human-Computer Interaction, for both lab and field studies, participant recruitment and the actual execution of the evaluation is time-consuming and requires significant effort by the researchers [6]. This fact limits the number of participants in most mobile text entry experiments to an average of approximately 15 [14], with very few notable exceptions (e.g. [18]). Although in many cases researchers manage to find statistically significant differences with such low numbers of participants, the generalisability of results is debatable. Often, participants are recruited through convenience sampling, and address a very specific part of the population (typically, students in engineering faculties), as do many HCI studies [5, 23]. To examine how prototypes might work with other populations, researchers must recruit more participants and carry out more experiments.

Theoretical performance modelling is a type of evaluation that attempts to overcome the challenge of recruiting large participant numbers. Using models driven by theory or derived from empirical data, researchers can estimate the upper and lower bounds of text entry performance against baselines (e.g. theoretical performance of a new keyboard layout vs. that of QWERTY), as, for example, in [2, 3, 15, 27]. However, these models include many assumptions which do not apply in real life, such as "the user always taps the desired key", or "the user needs no time to search for the next key to type". In many cases, these preliminary evaluations are followed up with user studies, to verify the modelling results.

This paper proposes a different approach to evaluations that rests between those without users (i.e. theoretical performance modelling) and those with actual users (e.g. collection of empirical data), through the use of modelled, synthetic experiment participants, that non-deterministic outcomes during simulations. RoboType is an open-source simulator for text entry, which allows researchers to code new text entry systems, and evaluate them in virtual experiments with an unlimited number of synthetic participants. These synthetic participants operate on behavioural parameters which can be derived from empirical data, or set manually by the researcher, to simulate various archetypes of users. For example, researchers can generate participants who suffer from tremor, have wide fingertips, are not familiar with QWERTY, are already experts in whatever new prototype is being evaluated, or become better by learning in the process of multiple trials (or even tired and bored in the process). RoboType is still under development, but I demonstrate the flexibility of this approach by comparing the ability of RoboType to reproduce theoretical performance bounds in alternative keyboard layouts, as found in literature, and to produce more realistic results by introducing stochastic touch modelling. The code and data is released as an open-source project, inviting the community to contribute to further development. Synthetic participants are free, endlessly configurable, require no consent, and can perform exceptionally large experiments that would take weeks, in mere minutes, with highly realistic results.

2 ROBOTYPE DESIGN

To simulate text entry on mobile devices, RoboType uses two primary components (classes): Keyboard and Finger. Further components working alongside these fundamental constructs can be added (e.g. a statistical decoder). The paper describe the principles for these in the next sections. Robotype is currently limited to simulating index finger entry only.

2.1 Keyboard

The Keyboard object is constructed by reading an XML file that describes the keyboard layout. The XML specification is a simplified version of the Android XML keyboard layout specification. It allows the specification of standard or specific key sizes and inter-key gaps as a percentage of the keyboard's dimensions. Key labels can be either specified as text, or can be inferred from icon resource names if a label is not present. Upon construction, the Keyboard object reads the XML source and maintains a dictionary of all present keys, with their coordinate center, coordinate bounding box, key length and height and, of course, label. Further, an R-Tree index of the key bounding boxes is constructed and maintained in memory. This index is used in another function of the Keyboard object, to quickly turn touch coordinates into input stream output (i.e. to identify which "key" the virtual "finger" has pressed). The R-Tree index is also used to find the nearest suitable target from the user's finger current position, if the desired key is repeated in multiple locations on the keyboard (e.g., left and right *shift* keys), so that optimal selection when multiple alternatives are available to the user can be modelled. Finally, the Keyboard object contains a helper function to plot the present keys and thus assist the visualisation of the in-memory layout.

2.2 Finger

The Finger object is responsible for "moving" around a Keyboard object and selecting targets from it. The main use of this object is through a function that takes an input string as a parameter, and attempts to move the finger from target to target on the Keyboard in order to type the input string. To do so, it can be assumed that the Finger should move from its current position to the next key center, however, the actual landing coordinates are calculated stochastically, based on FFits' Law, namely a dual gaussian distribution which includes the inherent imprecision of the motor system without a desired horizontal displacement component, and the imprecision caused by the horizontal velocity at which the finger moves towards the next target [4]. Finally, it is known that fingers slide while tapping targets, after the finger has touched down on the screen. These slides are modelled as movements in the direction of the original finger trajectory [17].

2.2.1 *Finger landing coordinates.*

According to FFits Law, the dispersion of touch down coordinates from an intended target follows a dual gaussian distribution. Its first component is the inherent inaccuracy of the motor system - for example, tapping the same target multiple times will result in a normally distributed dispersion of touch down coordinates around the intended target center. In RoboType, this is modelled using zero as a mean, and a fraction of each key's width and height as the standard deviation in a gaussian distribution, to provide a random set of coordinates away from the intended target center. The second component is inaccuracy due to movement speed. However, there is no bibliography exploring how user fingers' velocities change in different trajectories across a keyboard.

Based on [20], one can posit that users may achieve a faster average finger velocity across larger distances, since the lognormal distribution of velocity profiles allows for a greater top speed on longer distances. Jiang et al.'s dataset [10] was used to explore user velocity profiles, as it is the only openly available dataset that contains both finger tracking and touch data for mobile text entry. Visual inspection of plotted velocity-distance data, showed that maximum horizontal speed correlates linearly with distance, but average speeds calculated from both hand-tracking and touch event data appeared to follow a logarithmic function. To verify, a function of the general form $U_{(d)} = \alpha \times ln(\beta + d) + \gamma$ was fitted to the data of each individual participant, resulting in an average goodness of fit $R^2 = 0.655$ ($\sigma = 0.125, min = 0.056, max = 0.770$) (Fig.2a). Hence, in the previous equation, $U_{(d)}$ provides the average finger velocity to travel a distance d between two keys for a given participant.

The relatively low R^2 values are obviously owed to the dispersion of actual data points across the function line, however, the fitted function can be used as a base, from which to build realistic synthetic average velocity estimation data. To do so, for each participant, their trajectory data was grouped into 16 bins based on the distance between consecutively typed keys (Fig.2b). For each bin, the standard deviation of average finger velocities from the touch data was calculated. Then, to generate synthetic finger velocity data for any given participant and for any distance travelled, the fitted logarithmic function parameters for that participant was used to derive the mean velocity value for that distance. By adding gaussian noise to this value, using the respective binned distance velocity standard deviation, realistic samples of average data can be generated in a stochastic manner. The results capture real behaviour quite well, as can be seen in Fig. 3.

Therefore, finger kinematics for a particular participant in [10] can be modelled accurately using a set of parameters for the logarithmic value, and the binned velocity standard deviations as described previously. By extension, one can propose that entirely synthetic participants can be modelled by tuning the average speed function parameters using $\alpha = 0.98020925, \beta = 136.88089169, \gamma = -4.83177533$ as a starting point (these are the fit parameters using all participant data, see Fig.2a). Velocity standard deviation $S_{(d)}$ for travelling a distance d, can also be modelled using the 4th order polynomial $S_{(d)} = 5 \times 10^{-13} \times d^4 - 6 \times 10^{-10} \times d^3 - 8 \times 10^{-7} \times d^2 + 0.0014 \times d - 0.0213$, fitted across all participants' data, and adding gaussian noise to $S_{(d)}$ (Fig.2b).

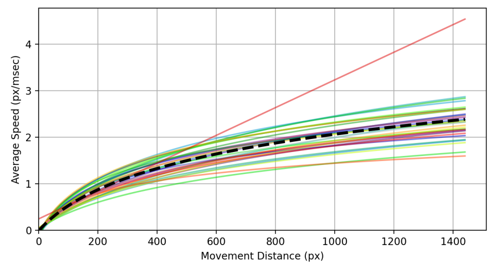
Finally, the simulator is afforded the ability to adjust the calculated finger movement velocity by a set scaling factor, to simulate users typing more slowly or faster than normal. While implemented and demonstrated in Section 2.2.7, this function is not used in the evaluation.

2.2.2 *Finger sliding.*

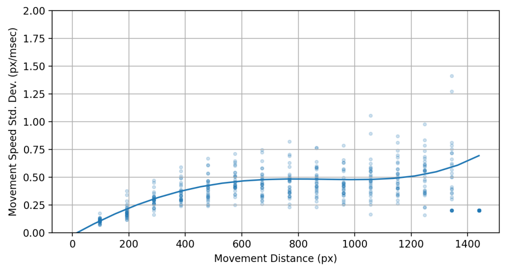
To model the slide magnitude, the dataset in [7] was used. This, to my knowledge, is one of the few, if not the only mobile text entry dataset that contains touch down and touch up events for each keystroke. Examining the dataset, it was not possible to find any relationship between movement distance or movemement speed against slide magnitude. On the other hand, it was noted that slides in most keys display a χ^2 distribution, though some keys had very few data points (Fig.4). Therefore the slide magnitude was modelled as a random value picked from such a distribution, with the respective key's average slide magnitude used as the degree of freedom parameter. Finally, both landing and slide coordinates were limited to the left, bottom and right bounds of the keyboard, as they correspond to the screen edges where the finger has no effect. The upper touches were not limited, since they might fall on areas that are outside the keyboard, but are still sensitive to touch.

2.2.3 *Keystroke timings.*

The previous modelling allows the calculation of timings for each keystroke, by dividing the distance between starting and landing coordinates by the calculated average



(a) Fitted $U_{(d)}$ parameters for each individual participant in [10] and across all data (black dotted line).



(b) Velocity standard deviations per distance bin, across all participants in [10], and fitted $S_{(d)}$ polynomial.

Figure 2: Velocity/distance profile models

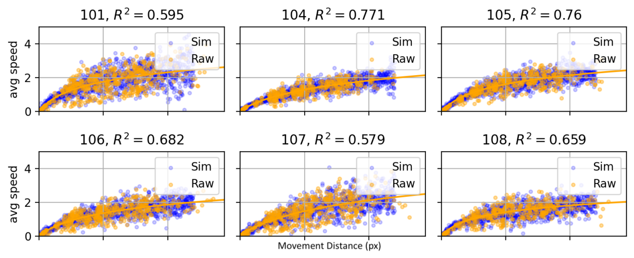


Figure 3: Example synthetic finger velocity data (2k data points) vs actual data for a small subset of participants (ids 101-108), demonstrating the realistic output of the speed estimation technique.

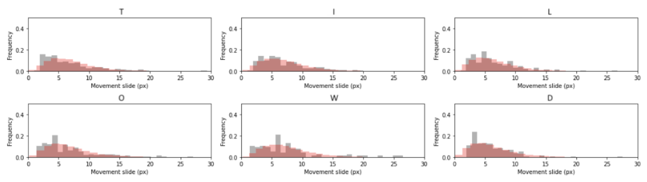


Figure 4: Example of slide length distribution for a small subset of keys (blue). The red histogram shows a histogram of 100k random samples from a χ^2 distribution.

speed to travel that distance. To model participant familiarity with the keyboard layout, a Visual Search Time (VST) component was included. This is calculated from a gaussian distribution, using a pre-determined base value expressed in milliseconds as a mean, and a scaling factor (e.g. 0.10) of this base value as a standard deviation. This allows the generation of synthetic participants with varying levels of familiarity with the keyboard. For example, a complete novice will require a mean of 1.2sec ($\sigma = 1.2 \times 0.1 = 0.12sec$) VST, which will be added to the movement time.

2.2.4 Keystroke errors. Based on the previous modelling, a synthetic participant may hit an unintended target, an area of the keyboard that does not correspond to a key, or a non-functional key. In the current level of development, modifier keys (e.g. shift, numeric mode, punctuation) are considered inactive areas of the keyboard. Therefore a mechanism was implemented, whereby a simulated touch that does not result in the desired keystroke is repeated, until it is successful. If the user inadvertently presses a backspace, the mechanism also ensures that the the synthetic user attempts to replace the deleted letter first, before proceeding to the desired target input. This mechanism assumes that the user has full and immediate awareness of the effect of their actions, which is not entirely realistic as users typically commit to a few characters before checking for errors [13]. Better simulation of the visual attention component is left for a further stage of development.

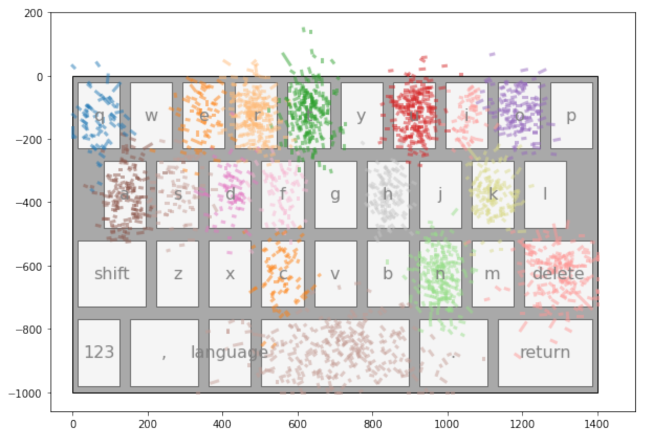
2.2.5 Replicability. In all previous sections, where random sampling is used, this can be done using a pre-determined seed for the random generator. In this way, RoboType can ensure the complete replicability of any results.

2.2.6 Input stream generation. At the end of any simulation run, RoboType produces the complete input stream for each synthetic participant, together with related metadata, therefore facilitating post-processing to derive text entry metrics (e.g. WPM, Error Rates etc.). The input streams are produced as an array of Python dictionary objects, which can be saved as JSON files, as per Fig. 5. Contrary to evaluations with real users, we are able to log not just the user's actual input, but also their intention. In the example in Fig. 5, the user intended to tap the letter "i", but overshot and hit the area above the keyboard bounds (y-coords are negative), therefore resulting in "none" as the actual input.

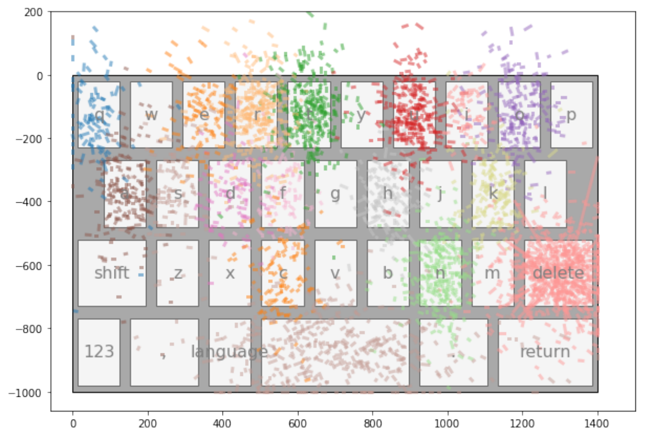
2.2.7 Demonstration. Fig.6 demonstrates the operation of RoboType using the profile of one participant in [10] (id=114) to type the phrase *"thanks for the quick turnaround"* fifty consecutive times. In the first run, the simulated participant attempts the task at normal input speed, resulting in a typing performance at 22.44WPM (hits:1705, misses:568, backspaces:119). In the second run, the participant attempts the task at twice their normal speed. This time they are faster (34.88WPM) but not actually twice as fast - this is expected as they make many more mistakes, which need to be fixed (hits:2224, misses:1599, backspaces:392). In this demonstration, a hit is any keystroke that finds its intended target. A miss is any keystroke that lands on an unintended target (other key, gap or inactive key, or inadvertent backspace).

```
[
  {
    "letter": "none",
    "intent": "i",
    "tap": {
      "x": 1045,
      "y": -91
    },
    "touchdown": {
      "x": 1042,
      "y": -86
    },
    "touchup": {
      "x": 1045,
      "y": -91
    },
    "speed": 5.476022073477649,
    "vst": 10.44116709260419,
    "time": 104.1144770855489,
    "motionlength": 512.9571132170797
  },
  ...
]
```

Figure 5: Sample input stream generated by RoboType. Each "tap" is represented as one JSON object.



(a) Synthetic participant typing at normal speed.



(b) Synthetic participant typing at 2x normal speed.

Figure 6: Demonstration of RoboType on a QWERTY keyboard. Typing at an increased speed causes more erroneous touches.

3 SYNTHETIC EVALUATIONS WITH ROBOTYPE

To further demonstrate the ability of RoboType to quickly evaluate novel designs, a comparison between QWERTY, FITALY and OPTI-II is perfomed, following the classic paper by Zhai et al. [27]. This paper was selected as it models single-digit (or stylus) input like RoboType, and in order to compare the ability of RoboType to reproduce these theoretically bounded results, as well as to explore the resulting differences when stochastic simulation is introduced. In Zhai et al. [27], the theoretical performance for these alternative layouts in terms of text entry speed, was calculated to be 28WPM (QWERTY), 36WPM (FITALY) and 38WPM (OPTI). These metrics assume that the user always aims for the key centers and succeeds to tap it, and that the user always makes an optimal choice of space key with OPTI and FITALY. The latter assumption also holds for RoboType, but as explained, RoboType presents a stochastic distribution of touch-down coordinates. The theoretical bounds in [27] have been validated by similar findings in alternative modelling approaches [9, 16].

3.1 Synthetic Participants

An entirely synthetic experiment was configured, consisting of 30 participants modelled after those in [10]. Each synthetic participant uses the same $U_{(d)}$ parameters as their real counterpart. Under this experiment, three cases were investigated: In the first case (*IDEAL*), participants are fully cognizant of the key locations, therefore $VST = 0$, and participants have fully accurate fingers so they always tap the center of the target (as per [27]). In the second case (*EXPERT*), it is assumed that $VST = 0$ but participant fingers are not fully accurate (motor system inherent distribution factor = 0.07, paired with a velocity-based distribution). This case represents a fully expert user having memorised the location of each key completely. Finally, the third case (*REAL*) presents a more realistic version of the expert users in the previous case. Having trained for a considerable period of time, participants have VST and VST scaling factor that are randomly picked from gaussian distributions with $\bar{x} = 100ms, \sigma = 50ms$ and $\bar{x} = 0, \sigma = 0.2$ respectively. In [11], a model for learning new keyboard layouts shows VST to drop logarithmically from approx 1.2sec to 0.7sec after 2.5 hours of training, but although the fit appears to be good, there is no validation on the model performance after longer training (e.g. a few *months* worth of keyboard use). However, in [10], it is argued that more experienced typists have less need to guide fingers with their eyes, thus a low VST (in this case, 100ms) is a realistic choice.

3.2 Experiment design

In every case, each synthetic participant was given a corpus of 50 phrases picked randomly for each participant, from the Enron Mobile Phrase Set [24]. Obviously, with synthetic participants, there are no learning effects therefore there is no need to counterbalance or otherwise account for human bias. The keyboard size was set to 1440 × 1000px.

3.3 Results

Table 1 and Fig.7 show the resulting WPM metrics for each participant setting and keyboard. In the Ideal condition, the produced metrics are within the ranges predicted by simulation in other papers [9, 16, 27], bearing in mind that these were conducted with diverse corpora. These results demonstrate that the modelling successfully captures theory-predicted results, and the modifications towards realism (motor system inaccuracy, visual search time) produce a better indication of performance for actual, well-trained users. Execution times for the entire experiment (all cases) was under 1 minute. In the next sections the paper outlines the statistical test results comparing performance with each keyboard under each case. Statistical tests were selected after examination of the required assumptions. The results show that the statistically significant differences predicted by theoretical modelling are still maintained in realistic simulations that overcome the assumptions of these models, as has been found empirically.

	QWERTY	FITALY	OPTI
IDEAL	30.971 (4.423)	36.168 (5.041)	36.475 (5.174)
EXPERT	30.238 (4.079)	35.354 (4.876)	35.593 (4.813)
REALISTIC	27.651 (4.777)	32.060 (6.252)	32.460 (6.418)
Theoretic bounds [9, 16, 27]	28.0 - 31.8	36.0 - 42.4	38.000

Table 1: Average WPM (standard deviations in parentheses).

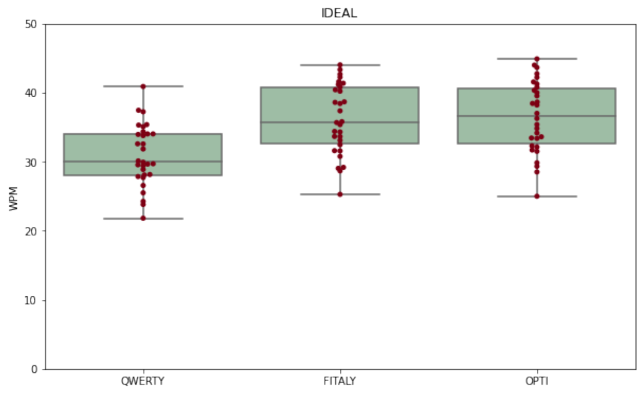
3.3.1 Case 1: Ideal. An ANOVA test showed statistically significant differences ($F_{(2)} = 11.999, p < 0.001$). Pairwise t-tests with post-hoc Bonferroni correction (p threshold at 0.017) showed the differences to be statistically significant between QWERTY and FITALY ($t = -17.744, p < 0.001$), QWERTY and OPTI ($t = -19.123, p < 0.001$) but not between OPTI and FITALY ($t = -2.365, p = 0.025$).

3.3.2 Case 2: Expert. An ANOVA test showed statistically significant differences ($F_{(2)} = 12.955, p < 0.001$). Pairwise t-tests with post-hoc Bonferroni correction (p threshold at 0.017) showed the differences to be statistically significant between QWERTY and FITALY ($t = -16.409, p < 0.001$), QWERTY and OPTI ($t = -15.872, p < 0.001$) but not between OPTI and FITALY ($t = -0.941, p = 0.355$).

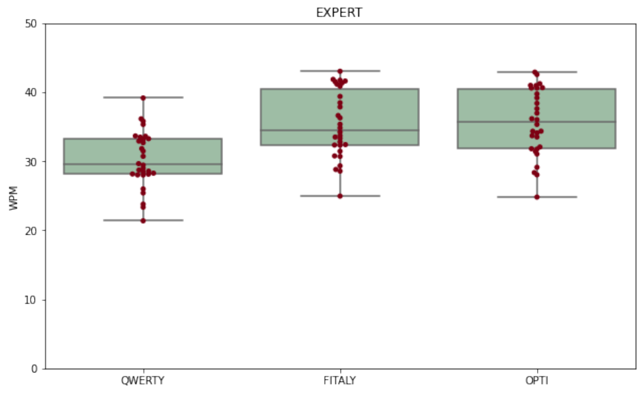
3.3.3 Case 1: Realistic. An ANOVA test showed statistically significant differences ($F_{(2)} = 6.218, p = 0.003$). Pairwise t-tests with post-hoc Bonferroni correction (p threshold at 0.017) showed the differences to be statistically significant between QWERTY and FITALY ($t = -11.635, p < 0.01$), QWERTY and OPTI ($t = -12.000, p < 0.01$) and OPTI and FITALY ($t = -3.158, p = 0.004$).

4 DISCUSSION

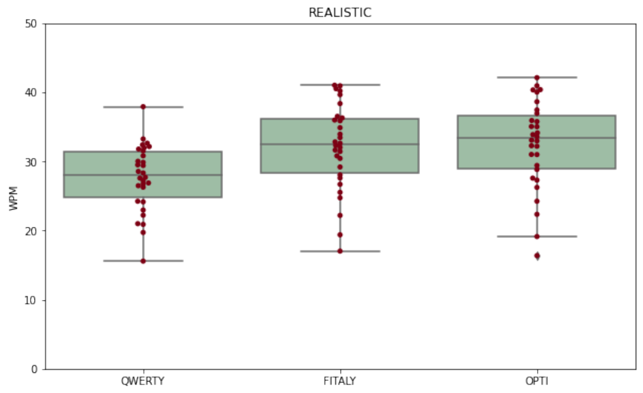
This paper presented RoboType, a simulator for mobile text entry research. Although development is still in progress, this late-breaking work demonstrates the potential of the simulator in realistically evaluating novel interaction methods for text entry, without the need for physical participants. Looking ahead, there are numerous components to the simulator which I am currently working on and (or) would like to see implemented.



(a) Synthetic participants typing at ideal conditions.



(b) Synthetic participants typing with $VST = 0$ and normal motor system inaccuracies.



(c) Synthetic participants typing at realistic conditions ($VST \neq 0$, normal motor system inaccuracies)

Figure 7: Results from a synthetic experiment with RoboType

In terms of finger kinematics, work is being made towards implementing findings from [1] that demonstrate that touch distributions are offset from key centers. This is likely due to handedness and posture during use, but it is difficult to validate this assumption without collecting and analysing new empirical data. Further, as RoboType currently only supports index-finger text entry, it should be extended to support thumb entry and two-thumb entry, further incorporating physical dimension aspects (e.g. device size, thumb length, thumb reach).

In terms of the keyboard itself, a statistical decoder to the default keyboard has been implemented based on [25] but is not presented in this paper. Next letter and next-word predictions based on neural networks and deep learning methods are also being worked on. Further work will implement support for text entry aids in the default keyboard of the simulator, including z-ordering of keys to allow for magnification of key bounding boxes according to statistical probability, a word suggestion bar and full support for numeric and symbol mode switching. While RoboType focused on 26 letter keyboards in this study, it can be easily extended through code to support other input techniques (e.g. 12-key multitap and predictive, swiping, multi-page layouts, long-presses, multiple languages etc.). All can be coded with relative ease, by extending the base finger and keyboard mechanics already supplied in the current stage of development.

Finally, one critical area still to be developed is a distinct component for the participants' visual system. This will help simulate not just visual search times in a more realistic way, but also to model user attention switching between the keyboard and text entry area, delayed observation of errors in the input stream (and their correction), as well as attention to visual on-screen artifacts related to text entry (e.g. suggestion bar, error highlighting etc.). The visual system model should be combined with further modelling to simulate accumulated participant experience as per [11], which will affect both finger kinematics and visual attention as synthetic participants gain exposure to new tasks and input methods.

I hope that in the future, RoboType can be used to generate huge synthetic text entry datasets, using instances of RoboType as remote collaborative agents that communicate with each other, for example through conversational interaction powered entirely by large language models such as ChatGPT, an appoach that has been recently proven feasible [19]. Such datasets could open up new pathways for text entry research, which are currently only available to industry-based researchers with access to data from commercially deployed keyboards (e.g. GBoard).

From an ethical perspective, RoboType presents a solution that may remove obstacles due to privacy (e.g. creating large datasets from in-the-wild studies). Additionally, further modelling of the sensorimotor and cognitive system may open up new options for research to benefit persons belonging to special categories, such as children, persons with visual, motor or cognitive impairments who are harder to recruit and require careful handling of informed consent.

In this paper, I have presented RoboType as an open-source tool for the mobile text entry community, which can be freely downloaded from GitHub at https://github.com/komis1/RoboType. I invite the community to make use of this tool and to contribute towards its further development. This release also includes all model parameters based on the work of [10], as used in the evaluation presented here.

REFERENCES

[1] Shiri Azenkot and Shumin Zhai. 2012. Touch Behavior with Different Postures on Soft Smartphone Keyboards. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services (MobileHCI '12)*. Association for Computing Machinery, New York, NY, USA, 251–260. https://doi.org/10.1145/2371574.2371612

[2] Nikola Banovic, Varun Rao, Abinaya Saravanan, Anind K. Dey, and Jennifer Mankoff. 2017. Quantifying Aversion to Costly Typing Errors in Expert Mobile Text Entry. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. Association for Computing Machinery, New York, NY, USA, 4229–4241. https://doi.org/10.1145/3025453.3025695

[3] Nikola Banovic, Ticha Sethapakdi, Yasasvi Hari, Anind K. Dey, and Jennifer Mankoff. 2019. The Limits of Expert Text Entry Speed on Mobile Keyboards with Autocorrect. In *Proceedings of the 21st International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3338286.3340126

[4] Xiaojun Bi, Yang Li, and Shumin Zhai. 2013. FFitts Law: Modeling Finger Touch with Fitts' Law. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. Association for Computing Machinery, New York, NY, USA, 1363–1372. https://doi.org/10.1145/2470654.2466180

[5] Kelly Caine. 2016. Local Standards for Sample Size at CHI. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. Association for Computing Machinery, New York, NY, USA, 981–992. https://doi.org/10.1145/2858036.2858498

[6] Karen Church, Denzil Ferreira, Nikola Banovic, and Kent Lyons. 2015. Understanding the Challenges of Mobile Phone Usage Data. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '15)*. Association for Computing Machinery, New York, NY, USA, 504–514. https://doi.org/10.1145/2785830.2785891

[7] Mark Dunlop, Andreas Komninos, and Emma Nicol. 2016. "OATS_201411 Highlighting Keyboard Study 2" Dataset. https://doi.org/10.15129/5fce2dfd-4a12-4637-8eaa-641f0eb319be

[8] Anna Maria Feit, Mathieu Nancel, Maximilian John, Andreas Karrenbauer, Daryl Weir, and Antti Oulasvirta. 2021. AZERTY Amélioré: Computational Design on a National Scale. *Commun. ACM* 64, 2 (Jan. 2021), 48–58. https://doi.org/10.1145/3382035

[9] Ana Beatriz Herthel and Anand Subramanian. 2020. Optimizing Single-Finger Keyboard Layouts on Smartphones. *Computers & Operations Research* 120 (Aug. 2020), 104947. https://doi.org/10.1016/j.cor.2020.104947

[10] Xinhui Jiang, Yang Li, Jussi P.P. Jokinen, Viet Ba Hirvola, Antti Oulasvirta, and Xiangshi Ren. 2020. How We Type: Eye and Finger Movement Strategies in Mobile Typing. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3313831.3376711

[11] Jussi P. P. Jokinen, Sayan Sarcar, Antti Oulasvirta, Chaklam Silpasuwanchai, Zhenxin Wang, and Xiangshi Ren. 2017. Modelling Learning of New Keyboard Layouts. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. Association for Computing Machinery, New York, NY, USA, 4203–4215. https://doi.org/10.1145/3025453.3025580

[12] Andreas Komninos and Mark Dunlop. 2014. Text Input on a Smart Watch. *IEEE Pervasive Computing* 13, 4 (Oct. 2014), 50–58. https://doi.org/10.1109/MPRV.2014.77

[13] Andreas Komninos, Mark Dunlop, Kyriakos Katsaris, and John Garofalakis. 2018. A Glimpse of Mobile Text Entry Errors and Corrective Behaviour in the Wild. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct (MobileHCI '18)*. Association for Computing Machinery, New York, NY, USA, 221–228. https://doi.org/10.1145/3236112.3236143

[14] Andreas Komninos, Vassilios Stefanis, and John Garofalakis. 2023. A Review of Design and Evaluation Practices in Mobile Text Entry for Visually Impaired and Blind Persons. *Multimodal Technologies and Interaction* 7, 2 (Feb. 2023), 22. https://doi.org/10.3390/mti7020022

[15] Per Ola Kristensson and Thomas Müllners. 2021. Design and Analysis of Intelligent Text Entry Systems with Function Structure Models and Envelope Analysis. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3411764.3445566

[16] Yanzhi Li, Lijuan Chen, and Ravindra S. Goonetilleke. 2006. A Heuristic-Based Approach to Optimize Keyboard Design for Single-Finger Keying Applications. *International Journal of Industrial Ergonomics* 36, 8 (Aug. 2006), 695–704. https://doi.org/10.1016/j.ergon.2006.04.009

[17] Yan Ma, Shumin Zhai, IV Ramakrishnan, and Xiaojun Bi. 2021. Modeling Touch Point Distribution with Rotational Dual Gaussian Model. In *The 34th Annual ACM Symposium on User Interface Software and Technology (UIST '21)*. Association for Computing Machinery, New York, NY, USA, 1197–1209. https://doi.org/10.1145/3472749.3474816

[18] Kseniia Palin, Anna Maria Feit, Sunjun Kim, Per Ola Kristensson, and Antti Oulasvirta. 2019. How Do People Type on Mobile Devices? Observations from a Study with 37,000 Volunteers. In *Proceedings of the 21st International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3338286.3340120

[19] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative Agents: Interactive Simulacra of Human Behavior. https://doi.org/10.48550/arXiv.2304.03442 arXiv:2304.03442 [cs]

[20] Réjean Plamondon, Christian O'Reilly, Céline Rémi, and Thérésa Duval. 2013. The Lognormal Handwriter: Learning, Performing, and Declining. *Frontiers in Psychology* 4 (2013). https://www.frontiersin.org/articles/10.3389/fpsyg.2013.00945

[21] Shyam Reyal, Shumin Zhai, and Per Ola Kristensson. 2015. 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 (CHI '15)*. Association for Computing Machinery, New York, NY, USA, 679–688. https://doi.org/10.1145/2702123.2702597

[22] André Rodrigues, Hugo Nicolau, André Santos, Diogo Branco, Jay Rainey, David Verweij, Jan David Smeddinck, Kyle Montague, and Tiago Guerreiro. 2022. Investigating the Tradeoffs of Everyday Text-Entry Collection Methods. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22)*. Association for Computing Machinery, New York, NY, USA, 1–15. https://doi.org/10.1145/3491102.3501908

[23] Anam Sohail. 2020. *Challenges in Recruiting Participants for Studies in HCI.* Master's thesis. RWTH Aachen University, Aachen.

[24] Keith Vertanen and Per Ola Kristensson. 2011. A Versatile Dataset for Text Entry Evaluations Based on Genuine Mobile Emails. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI '11)*. Association for Computing Machinery, New York, NY, USA, 295–298. https://doi.org/10.1145/2037373.2037418

[25] Keith Vertanen, Haythem Memmi, Justin Emge, Shyam Reyal, and Per Ola Kristensson. 2015. VelociTap: Investigating Fast Mobile Text Entry Using Sentence-Based Decoding of Touchscreen Keyboard Input. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. Association for Computing Machinery, New York, NY, USA, 659–668. https://doi.org/10.1145/2702123.2702135

[26] Christoph Wimmer, Richard Schlögl, Karin Kappel, and Thomas Grechenig. 2019. Measuring Mobile Text Entry Performance and Behaviour in the Wild with a Serious Game. In *Proceedings of the 18th International Conference on Mobile and Ubiquitous Multimedia (MUM '19)*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3365610.3365633

[27] Shumin Zhai, Michael Hunter, and Barton A. Smith. 2002. Performance Optimization of Virtual Keyboards. *Human–Computer Interaction* 17, 2-3 (Sept. 2002), 229–269. https://doi.org/10.1080/07370024.2002.9667315