Decoding Surface Touch Typing from Hand-Tracking

Mark Richardson
Facebook Reality Labs
Redmond, WA
echo@fb.com

Matt Durasoff
Facebook Reality Labs
Redmond, WA
durasoff@fb.com

Robert Wang
Facebook Reality Labs
Redmond, WA
rywang@fb.com

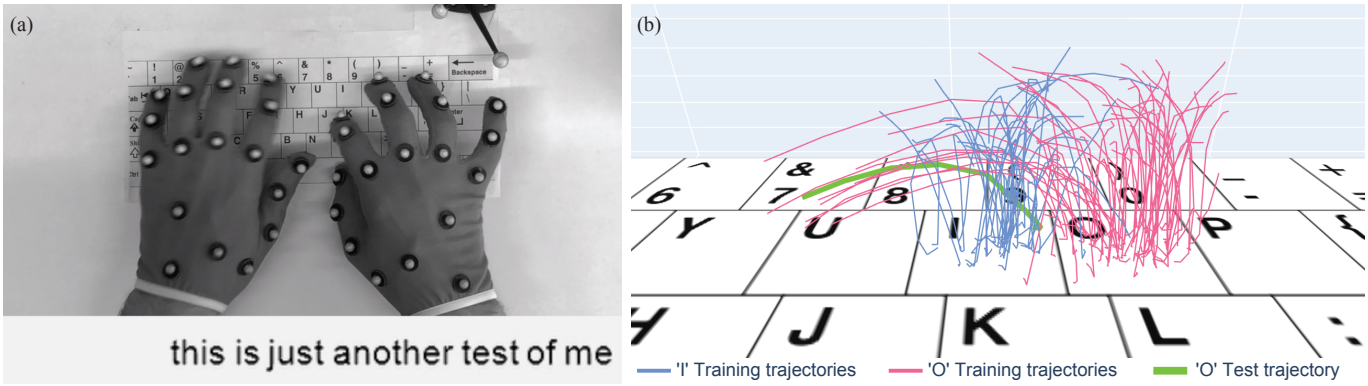

Figure 1. (a) We demonstrate touch typing on a flat surface using a decoder that translates hand motion from a skeletal hand-tracking system into text. Compared to contact-based sensing, e.g., capacitive touch, hand motion additionally encodes the trajectory of a finger as it reaches for a key, which we can leverage. For example (b), we show two clusters of trajectories of the right middle fingertip from the training dataset; one for 'O' keystrokes (magenta) and one for 'I' keystrokes (blue). The sample trajectory at test time (green) lands on the 'I' key but has a more similar path to the 'O' key. Our approach uses this trajectory information to correctly classify this case while a contact-based approach cannot.

ABSTRACT

We propose a novel text decoding method that enables touch typing on an uninstrumented flat surface. Rather than relying on physical keyboards or capacitive touch, our method takes as input hand motion of the typist, obtained through hand-tracking, and decodes this motion directly into text. We use a temporal convolutional network to represent a motion model that maps the hand motion, represented as a sequence of hand pose features, into text characters. To enable touch typing without the haptic feedback of a physical keyboard, we had to address more erratic typing motion due to drift of the fingers. Thus, we incorporate a language model as a text prior and use beam search to efficiently combine our motion and language models to decode text from erratic or ambiguous hand motion. We collected a dataset of 20 touch typists and evaluated our model on several baselines, including contact-based text decoding and typing on a physical keyboard. Our proposed method is able to leverage continuous hand pose information to decode text more accurately than contact-based methods and an offline study shows parity (73 WPM, 2.38% UER) with typing on a physical keyboard. Our results show that hand-tracking has the potential to enable rapid text entry in mobile environments.

Author Keywords

text input; hand-tracking; augmented reality; virtual reality

CCS Concepts

•**Human-centered computing** → **Text input;** Virtual reality;
•**Computing methodologies** → Natural language generation;

INTRODUCTION

Text entry is an important task for communication and productivity in augmented reality and virtual reality (AR/VR). While conventional physical keyboards and touch screens can be incorporated into AR/VR input systems, added peripherals detract from mobile use cases and accessibility. Automatic speech recognition is more accessible, but may not be socially acceptable for certain environments or for private communication. Recent advances in computer vision have shown that hand pose can be accurately estimated using commodity depth, RGB or monochrome cameras. Commercial AR/VR devices such as the Oculus Quest and the Microsoft Hololens have started using hand-tracking for text entry. However, existing hand-tracking-based text entry solutions for AR/VR are relatively low throughput.

In this paper, we investigate the use of hand-tracking to enable typing on any flat surface at speeds comparable to typing on a physical keyboard. This has several advantages to existing approaches. Hand-tracking is typically achieved through on-headset sensors, without extra peripherals, making it suitable

UIST '20, October 20–23, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7514-6/20/10.
http://dx.doi.org/10.1145/3379337.3415816

for on-the-go use cases. Typing on a virtual keyboard is more discreet than speaking. Most importantly, there already exists a wide audience of effective typists. Surveys of internet users show that even average typists can achieve speeds greater than 50 words per minute (WPM) with the fastest 90^{th} percentile achieving more than 78 WPM [3]. We show that these users can transfer their existing skills typing on a keyboard to typing on a flat surface.

People type faster on a physical keyboard [3] than a soft keyboard and also faster on a surface [4] than in the air. In our study we compromise between accessibility and speed by requiring users to type against a surface but eschewing the use of a physical keyboard or a touchpad. Additionally, surface typing has been shown to be more comfortable than mid air typing [4] and confers haptic feedback, which enhances presence in VR [16]. Our investigation caters particularly to the fastest typists–*touch typists*, which we mean as those who type without the sense of sight to find the keys. Without the feel of physical keys and without using sight to find the keys, fingers will drift during typing. Instead of requiring users to make precise contacts on a fixed keyboard layout, we investigate using a motion model to recognize finger trajectories, and we further explore how statistical decoding techniques affect performance.

We are inspired by the work of Dudley and colleagues [4] that shows the high potential of human typing efficiency and error rate given a text decoding oracle with knowledge of the text being typed. In this work, we take the first steps to reducing this oracle to practice by building a neural model of text decoding that combines motion modeling of fingers and a state-of-the-art language model. Notably, our model is the first technique to our knowledge that converts skeletal hand motion directly into text. Instead of relying on surface contact information (e.g., from capacitive touch), we investigate using the output of a marker-based hand-tracking system [12]. Hand-tracking provides potentially richer sensing information (including finger identities and trajectories) than the contact modality. While marker-based hand-tracking is still an optimistic approximation of the fidelity achievable from an AR/VR headset, our experiments shed light on the potential of hand-tracking-based text decoding. Specifically, we make the following contributions:

- We propose a motion model, represented as a temporal convolutional network (TCN), that can translate hand motion directly into (a probability distribution over the) typed text.
- We show that we can combine our motion model with a language model, also represented as a neural network, using an efficient beam search decoding technique.
- We show that tracking hands typing on a flat surface combined with our statistical decoding method has the potential of achieving speeds comparable to typing on a physical keyboard while maintaining low-error rates. We also explore why decoding continuous hand motion data could be advantageous to decoding a discrete set of contacts on a touch surface by isolating the value finger trajectory information, finger identity information and continuous decoding.

RELATED WORK

Surface and eyes-free keyboards

Typing on a flat surface is analogous to typing on a soft keyboard on a smartphone and inherits similar problems such as noisy input and systematic offsets [14] which can be addressed via statistical decoding methods [7, 10, 25] (see below). Touch typing, by which we mean eyes-free input, on a flat surface has been studied in the context of gesture keyboards. The i'sFree system [30] is an example of a gesture keyboard that learns keyboard location from gesture input over time. Yang et al. [27] improve first touch accuracy for a gesture keyboard by centering the keyboard around first touch. Our work is most similar to that of Zhu et al. [29], who enforced touch-typing on a smartphone surface by asking users to tap on an invisible keyboard. We show that by using hand pose rather than contact points and a richer motion model, we can achieve more accurate decoding.

AR/VR text entry

Many other text entry systems have been proposed for AR/VR applications, and for brevity, we concentrate on those that use a QWERTY keyboard layout. The ARKB system [18] showed a very similar concept of typing on a QWERTY virtual keyboard on a surface using hand-tracking through color-based segmentation and markers. Since then, hand-tracking technology has advanced significantly. The ATK system [28] uses skeletal hand-tracking (via a Leap Motion device) to drive text entry, although through midair typing. The VISAR system [5] also uses hand-tracking (via a Hololens device) to enable typing, although with just the index fingers. VISAR also integrates a more sophisticated decoding strategy [25] to achieve superior mid-air typing on Hololens. Vulture [21] uses a high-precision marker-based hand-tracking system to capture a pinch and the tracing of a path (of a word-gesture) through a QWERTY keyboard, which are then decoded into the best word proposals. Similarly, RotoSwype [11] uses a ring to capture the motion of a single finger tracing a path through a QWERTY keyboard in the air, achieving speeds of at least 14WPM with a 1% error rate by the end of a five-day study. PalmType leverages passive haptics by using the opposite hand as a tapping / typing surface, although this approach comes at the cost of limiting typing to a single hand. Our method is designed for typing on a flat surface rather than in the air or against the opposite hand. We believe the on-surface domain, while slightly less accessible, better leverages existing typing skills, allowing our method to achieve speeds comparable to typing on a keyboard at a low error rate. We also use complete hand pose of all the fingers on both hands rather than the pose of a single finger.

Statistical text decoding

Goodman et al. [7] first applied statistical decoding to improve the accuracy of a soft keyboard by modeling key-targets as bi-variate Gaussians and representing a language model as an n-gram distribution. Gunawardana et al. [10] improve on key-targets with dynamic resizing and respecting minimal key regions or anchors. Weir et el. [26] use Gaussian Process regression to model key targets. Kristensson et al. [17] model a trajectory of contacts as well as words in a language model geometrically and decode stylus gestures via a geometric matching method. Similar to our proposed method, Velocitap [25]

uses beam search with a language model to decode whole sentences at a time, although Velocitap uses contact as the input modality while our method uses hand-tracking.

Automatic speech recognition

Statistical decoding of typing is analogous to text decoding in automatic speech recognition. The combination of connectionist temporal classification (CTC) loss and beam search is a cornerstone of modern end-to-end speech recognition systems [1]. Graves and Jaitly [9] first applied CTC loss with a prefix beam search decoder to speech recognition, and Hannun and colleagues [13] showed how to directly incorporate a language model. We apply this same general framework, substituting recurrent neural networks with a TCN-based motion model, to decode text entry via typing.

APPROACH

To evaluate the feasibility of using hand-tracking to enable touch typing on virtual keyboards, we first collect a dataset of skeletal hand-tracking data from touch-typists transcribing short phrases while typing on a flat surface. Next, we design a system for decoding the skeletal hand-tracking data into the text the typists intended to type. Finally we measure typing speed and error rates of our system against typing on a physical keyboard and contact-based statistical decoding methods.

To collect a dataset of skeletal hand-tracking data, we make use of a high quality marker-based hand-tracking system [12] which is not subject to the current tracking limitations of consumer head mounted hand-trackers. Consumer hand-tracking systems on current generation HMDs are optimized for virtual manipulation tasks rather than text input. While using a marker-based tracking system may introduce a gap in tracking quality compared to what is achievable on an AR/VR headset, we can study the 'potential' of hand-tracking applied to this problem.

For decoding hand motion into text, we make no such concessions and investigate the accuracy we can achieve using practical language models and statistical decoding strategies presented in this work.

Flat surface typing is a paradigm that exists today in tablet computers. While we are targeting a different sensing modality (hand-tracking instead of capacitive touchpads), we can use touchpads as a baseline for our work. We first investigate touch typists ability to transfer their physical keyboard typing skills to typing on virtual keyboards imposed on flat surfaces, and we then compare the performance of decoding text from skeletal hand-tracking data to decoding text from 2D surface contact points from a touchpad.

SYSTEM DESIGN

The problem of generating text from hand motion has strong analogs to automatic speech recognition (ASR), and we use ASR as motivation to design a multi-component system with the following three pieces.

1. A motion model, analogous to an acoustic model, which takes a sequence of hand poses and for each frame outputs a likelihood over an alphabet of tokens, i.e., keys plus a blank token corresponding to no key.
2. A language model which can give the likelihood of an additional token given a prefix of tokens.
3. A decoder which can optimize an objective function combining the likelihoods from both the motion and the language models.

The motion model captures the mapping from finger trajectories to intended key presses. This mapping is inherently ambiguous, for example, when a finger strikes the boundary between two keys, and is exacerbated by drift of the users' fingers due to the lack of haptic feedback of physical keys. We apply a beam search decoder to resolve these ambiguities using the language model as a prior.

MAPPING SKELETAL MOTION TO TEXT

We use a temporal neural network as a motion model. This model takes as input a sequence of frames of hand-tracking features and for each frame outputs a probability distribution over the set of possible keys, plus one additional label for blank / no-key. The model should assign more likelihood to a key when the user is hitting a key, and otherwise assign likelihood to the blank label.

For temporal modeling, we opt to use a temporal convolutional network (TCN) rather than a recurrent model because a fixed window of hand motion data is typically sufficient to make predictions about key presses. Longer term context is useful for prediction in the context of language, but since our system design separates motion modeling from language modeling, a TCN works well for the former. An advantage of using a TCN is efficiency during training, since TCNs can process entire sequences in parallel rather than sequentially as with recurrent neural networks. We follow the TCN architecture proposed by Bai et al. [2] which consists of causal dilated convolutions and weight normalization arranged in residual blocks. We use three layers of residual blocks with 64, 64, and 32 hidden units respectively, a kernel size of 2 and a dilation factor of 3. This architecture allows for 46 past frames of hand features (around 0.75 seconds of motion at 60Hz) to be referenced to generate the current probability distribution over the set of keys.

The input features to the network are frame-to-frame deltas of wrist position and rotation along with 3D fingertip positions. All positions are represented in the coordinate frame of the keyboard (the touchpad, or a virtual keyboard in the runtime). We chose these features because they are somewhat invariant to differences in hand scale across people.

The network is trained with a batch size of 32, where each individual sample consists of input features $\{u_i\}_{i=1}^{T}$, where T depends on how long it took the participant to type the phrase. The target for each sample is the sequence of keys $\hat{W} = \{\hat{w}_j\}_{j=1}^{N}$ that were prompted to the participant. The network is trained using the CTC loss function[8], which allows for sequence level labels without needing known alignment of labels to individual frames of input data. The output of the network $V = \{v_i\}_{i=1}^{T}$ is a corresponding sequence of T frames, each containing a probability distribution $v_i(k)$ over the set of the $K+1$ possible keys (with one extra for the blank label). This output can be interpreted as a $T \times (K+1)$ heat map

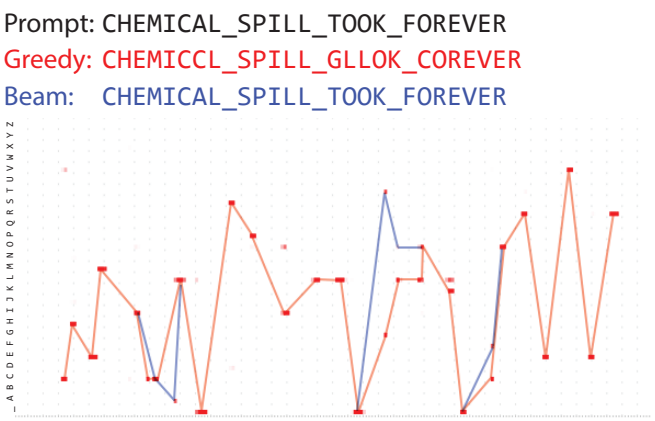

Figure 2. The output of our motion model is a heat map of predictions of which key was hit at which time. A greedy decoding of this heat map can result in errors, while incorporating a language model and beam search significantly reduce the error rate.

containing the prediction of which key was hit at which time (Figure 2). To transcribe the probabilities V into typed text, this output must be *decoded* into a sequence of keys W.

TEXT DECODING WITH A LANGUAGE MODEL

A naive greedy approach to decoding is to generate a label for each frame $t \in T$ by taking $\ell_t = \text{argmax}_k v_t(k)$. These labels $\{\ell_t\}_{t=1}^{T}$ are *compacted* into a transcription W by first combining adjacent frames with the same label into one of that label, and then by removing all blank labels.

We can do better than greedy decoding by using a prefix beam search decoder [13]. A prefix beam search decoder approximately maximizes $p(W;V)$ by incrementally constructing W by tracking a set of B best candidates (beams) at any given step. Furthermore, we can incorporate a joint probability from both the likelihood of the beam according to the motion model $p(W;V)$ as well as the likelihood of the compacted text according to a language model $p_{lm}(W)$, i.e., $p_{total} = (p(W;V)p_{lm}(W)^{\gamma})^{\frac{1}{1+\gamma}}$ where γ is a hyperparameter to control the balance between the two likelihoods. This allows the language model to steer decoding when the motion model is uncertain (See Figure 2). As has been shown in the speech recognition community [9], beam search decoding is also well-suited to the CTC loss with which we train the network.

We use a beam search implementation with beam compaction which maximizes the objective $W = \text{argmax}_W p_{total}$. After decoding, we compute the uncorrected error rate $\text{UER}(W,\hat{W})$ as the Levenstein edit distance between the decoded and prompted strings divided by the number of characters of the longer of the two strings. For all of our results, we use $B = 100$ beams. For our language model, we use a Transformer [24] model similar to the "small-two" model described in [15] with 4.76M parameters. This language model was trained on using a window of 500 characters on text sampled from 2 million articles in the CC-News dataset [20].

Figure 3. Our data collection setup consists of a marker-based hand-tracking system using OptiTrack cameras and a touchpad for recording contact. A printed 2D keyboard guide is attached to the touchpad for reference.

DATA COLLECTION

In our study we collected data by having participants complete a text transcription task for short phrases. To evaluate hand-tracking as an input modality compared to contact data from touchpads or to physical keyboards, we asked participants to type each phrase on either a physical keyboard or on a Sensel pressure sensitive touchpad with a printed 2D keyboard layout affixed (See Figure 3). In both cases, we also captured their hand motion using a marker-based hand-tracking system [12].

We recruited 20 participants who passed a pre-screening questionnaire designed to select for experienced typists. In this questionnaire all participants self identified as expert typists who type for more than 30 minutes per day and who use at least 3 distinct fingers to type the word "ghost" on a QWERTY keyboard.

Participants were given prompts with 3-6 word phrases containing only lowercase alphabetical characters and spaces. After typing each phrase, participants would use one of two foot pedals to mark that they felt they correctly typed the phrase, or that they typed the phrase but felt they made mistakes. Since our goal is to produce the text people intended to type, we filtered samples where the user believed they made a mistake since we don't know what text the user expects to be produced in those cases.

Participants typed phrases up to 40 characters long drawn from two corpora; samples used to fit or train models were randomly sampled from Daily Dialog [19], while samples for testing and evaluation were randomly sampled from Mackenzie and Soukeroff [23]. For each participant, data was collected in three distinct blocks in the following order:

1. 5 minutes of test set phrases on a physical keyboard
2. 60 minutes (in 5 minute blocks) of training set phrases on a touchpad
3. 5 minutes of test set phrases on a touchpad

Aside from the prompted phrase, no visual feedback was afforded in any of the blocks. Participants typed a median of 362 training phrases (approximately 6650 characters), and a median of 131 testing phrases (approximately 3400 characters). Test phrases were randomly sampled from a corpus of 500 phrases giving a possibility of overlap between the physical

keyboard and surface test set phrases. Using the foot pedals participants discarded a median 15.89% of surface phrases compared to 12.5% of physical keyboard phrases; a Wilcoxon signed rank test found no significance ($p = 0.328$).

Hand-tracking information was recorded while participants typed during all three blocks. Participants wore elastic mesh gloves with 19 retro-reflective motion capture markers affixed, and the method from Han and colleagues [12] was used to generate skeletal hand tracking information–specifically, joint angles and wrist transforms necessary to drive a fitted hand mesh. The touchpad was calibrated by placing three retro-reflective hemispheres at corners of the touchpad and fitting a transform to align those 2D touchpad coordinates with the corresponding 3D marker positions. This transform enabled all 2D contact points to be lifted into the 3D coordinate space in which the skeletal hand poses were represented.

Because contact-based text decoders are sensitive to accidental or missed touches, we took care to clean the contact-based training data from the touchpad captures. We filtered out captures where the number of touchpad contact events did not agree with the number of characters in the prompted phrase. In each of the remaining samples we expect that the N^{th} contact event corresponds to the N^{th} character in the prompted phrase. To filter samples where both an extra and a missing contact event occurred (leading to an agreement on counts but an error in contact-key sequence alignment), we build up a distribution for each key of how many times each finger hit that key. We then filter samples where multiple keys in the sample were struck by a finger which hits that key in less than 1% of cases overall.

The resulting touchpad training dataset consists of samples where we have 1) a sequence of length N of keys that were typed, 2) a sequence of length N of 2D contact events from the touchpad, and 3) a sequence of length T of skeletal hand poses while those keys were typed, recorded at 60Hz.

The testing dataset used for all decoders is the same, except that we only filter samples where the user felt they made a mistake, which means that the number of characters typed might not equal the number of contact events. We wish to quantify our ability to produce the text people intended to type, which should include all of the samples where participants felt they typed the phrase correctly.

While the contact-based baselines can only be trained with known key/contact-point correspondences, our motion model is trained using CTC loss, which allows for sequence level labels and can deduce the frame-level alignment of the labels. Thus our motion model was able to be trained on the original training dataset, only filtering samples where users felt they made a mistake. We note that this less filtered training dataset more closely mirrors the testing dataset.

EVALUATION

We present two evaluations of our proposed text entry method using the data collected from our user study. The first evaluation is a direct comparison against physical keyboards, and the second evaluation consists of two experiments aimed at better understanding which aspects of the motion model make it perform so well.

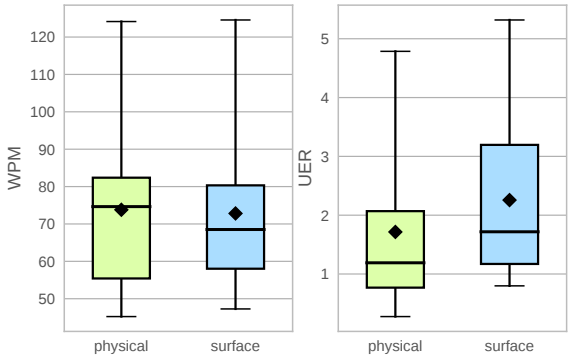

Figure 4. Boxplots depicting median, quartiles, and means comparing typing speed and accuracy between physical keyboard and flat surface typing.

For our motion model we trained person-specific models that captured each user's typing style from the training set portion of the data. All of our results were computed over the test set portion of the data. While this is not an interactive result, it simulates the same condition of decoding additional text from the user after a model had been trained (although without providing visual feedback to the user).

Physical keyboard versus virtual keyboard

Our first evaluation is to determine if participants are able to transfer their existing skills typing on a physical keyboard to typing on a flat surface (with the help of our proposed decoder). Participants of our user study type at different speeds, so we compare the relative speed of typing on a flat surface versus a physical keyboard. Because our participants were tasked with typing test-set phrases on both a physical keyboard and on a touchpad, we can make this comparison directly. We found that our expert typists generally typed efficiently both on physical keyboards (median: 75 WPM, mean: 74 WPM) as well as on surfaces (median: 69 WPM, mean: 73 WPM). A Wilcoxon signed-rank test also showed no statistically significant difference ($p = 0.859$) with participants' typing speed across the two modalities.

Participants typed on physical keyboards with a mean UER of 1.72% (median: 1.19%) compared to a mean UER of 2.38% (median: 1.77%) when they typed on flat surfaces with our decoder. A Wilcoxon signed-rank test showed no statistically significant difference ($p = 0.131$), however the descriptive statistics might suggest that users can still type more accurately on physical keyboards (Figure 4).

This result is consistent with previous findings on the performance envelopes of human typing [4] which described the high potential of typing speeds given a limited text decoding oracle. Our result reduces this oracle to practice by substituting it with our proposed neural text decoder.

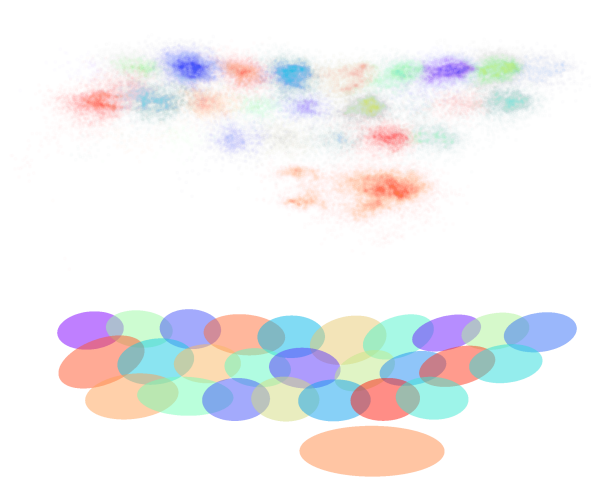

Figure 5. As a baseline, we compare our method to a contact-based spatial model that fits bivariate Gaussians (bottom) to all the contact events of each key (top).

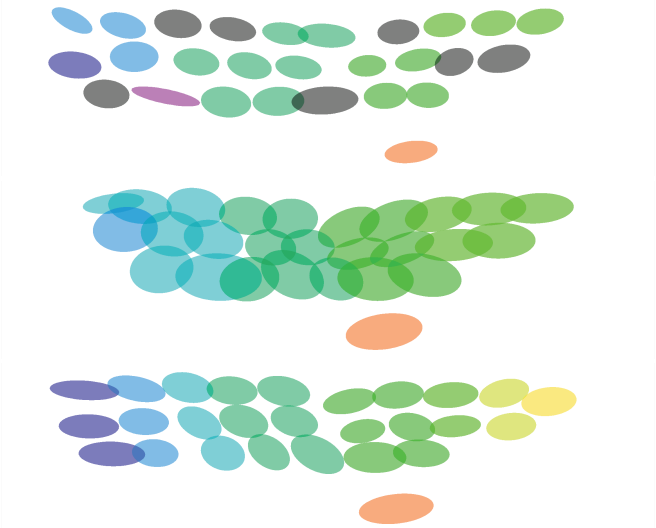

Figure 6. Variation in typing consistency and key-finger correspondences shown in three participants from our dataset. Each ellipse is a bivariate Gaussian fitted to contact points for a key colored by which finger accounts for 90% of contacts of that key.

Comparison of Input Features and Motion Models

Our system differs from prior work in several ways. We use a higher capacity neural motion model to decode a richer input signal (i.e., with finger identity and trajectory information) into text. Our decoding is also continuous (at 60Hz) rather than constrained to producing characters at discrete contact events. We attempt to tease apart the contributions of these differences with two experiments.

In our first experiment we study the contribution of finger identity information. To do so we create two contact-based baselines for comparison, one that uses only anonymous, discrete contact events, and another that adds finger-identity information. We compare these baselines to each other and to our hand motion based decoding to isolate the impact of finger identity information.

In our second experiment we investigate the contribution of continuous motion decoding versus discrete contact-based decoding. We observed that many mistakes made by contact-based decoding were due to its discrete nature: Characters are generated if and only if there is a corresponding contact event, which means that spurious or omitted contacts necessarily result in decoding errors. We attempt to ablate away the continuous nature of hand motion decoding by creating a filtered test dataset containing only samples with matched contacts and keys. We repeat our evaluation of the two baselines on this corresponded contacts dataset, isolating the value of trajectories from the value of continuous decoding and finger identity information.

Experiment 1: Comparison with Finger Identity Information

For our first baseline, *Gaussian contact decoding*, we evaluate performance using the spatial modeling approach described by Zhu et al. [29]. This model lacks the continuous text decoding capabilities of our motion model, but rather has to decode text only at discrete contact events. This model also lacks the

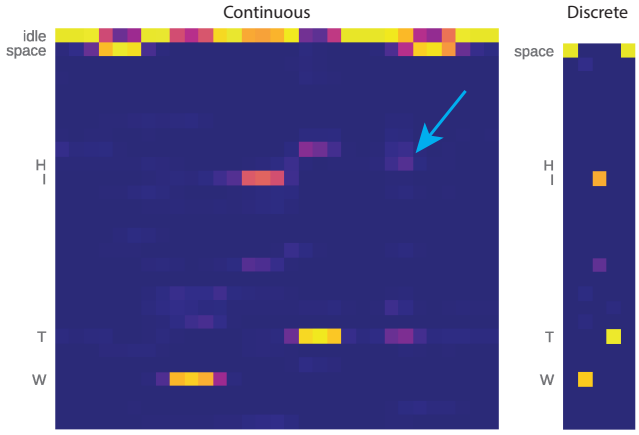

Figure 7. A typist types the word "WITH" but fails to make contact for the letter 'H'. Continuous decoding makes one prediction every 60^{th} of a second and can predict the possibility of the missing contact (indicated by the blue arrow) based on hand motion near the surface. Discrete contact-based decoding only predicts when contacts occur and thus cannot predict the missing letter.

richer input features of hand-tracking, e.g., which finger is hitting a key or which path the finger took to reach a key.

Using the cleaned training dataset, we obtain a sequence of keys pressed $\{\hat{w}_i\}$ and a sequence of 2D contact points from the touchpad $\{\hat{x}_i\}$ (See Figure 5). Since these sequences are in correspondence, we can bucket 2D contact points for each key on the keyboard. For each key k we fit a bivariate Gaussian $g_k(x) = N(x; \mu_k, \Sigma_k)$ from the collection of 2D contacts. During inference, the likelihood of a novel 2D contact point is measured according to each Gaussian, and the resulting distribution over all keys is then normalized $p(k;x) = g_k(x) / \sum_{k'=1}^{K} g_{k'}(x)$. This distribution over the set

of keys can be processed using the same beam search decoder and language model we described for hand motion decoding.

For our second baseline, *Per Finger Gaussian contact decoding*, we specifically investigate if augmenting the spatial model input with finger identity information improves performance of the contact-based model. Recent studies show that touch typists and faster typists tend to have lower entropy in their finger to key mapping [6]. As shown in Figure 6, we found our expert typist participants tended to have consistent key-finger correspondences. Motivated by this finding, we hypothesized that providing a richer input feature that includes which finger pressed a key can help a model disambiguate which key was pressed.

To use finger identity information, we make two changes to our Gaussian contact decoding baseline. First, we find the closest fingertip f for each contact using hand-tracking data and then bucket by both key k and finger f to fit 2D Gaussians $h_{k,f}(x)$ to these subsets of contact points. Second, at inference time, we use the known fingertip f to compute the distribution over the set of keys for a specific finger $p(k;x,f) = h_{k,f}(x)/\sum_{k'=1}^{K} h_{k',f}(x,f)$.

Using our touchpad evaluation dataset we compared the two baselines with our motion model approach (Figure 8). We decoded all test set samples using the Gaussian, Per Finger Gaussian, and our hand motion model with a greedy decoder with no language modeling. We then tested a beam search decoder with a language model on the same three conditions.

With greedy decoding, the 2D contact-based methods appeared to have higher variance, though the median UER was comparable across the three strategies. When we added the beam search language model decoder, however, we found that both Gaussian and Per Finger Gaussian contact-based methods performed comparably with a mean UERs of 5.66% (median 2.70%) and 4.76% (median 2.64%) respectively, while our motion model approach outperformed both with a mean UER of 2.38% (median 1.77%). A Wilcoxon signed-rank test found the motion model performance to be significantly better than both Gaussian ($p = 0.0258$) and Per Finger Gaussian decoding ($p = 0.011$), indicating that factors beyond finger identity had a strong impact on performance.

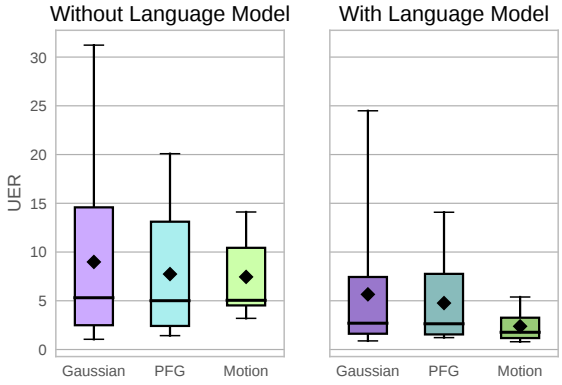

Figure 8. Uncorrected error rate for various decoding strategies. G=Gaussian contact model, PFG=Per Finger Gaussian contact model, Motion=Hand motion model.

Experiment 2: Comparison with Corresponded Contacts

The previous experiments show that our continuous hand motion decoding method is more accurate than decoding of discrete contacts with either a simple Gaussian spatial model or a richer model with finger identity information. When we examined the results qualitatively, we noted several differences between the contact-based decoding approach and our hand-tracking-based approach, which we illustrate in Table 2. Some of the error discrepancies could be explained by augmenting with finger identity information. Others could be attributed to the addition of finger trajectory information. Figure 1(b) shows an example where our motion model is able to learn to detect the signature trajectory of reaching for a particular key. Finally, others were due to the fundamentally discrete nature of contact decoding.

At first, we assumed that the discrete nature of contacts was an advantage for contact decoding. Explicit contact information is a strong signal for a key-press, whereas our motion model has to learn to detect the signature trajectory of a finger as it presses a key. However, when using a touchpad, we assume no ambiguity in whether a contact event occurred, only 2D spatial ambiguity about which key was pressed. There is no mechanism to detect an accidental touch or to guess when a touch event should have happened but was missed. Because contact-based decoding occurs only at discrete touch events, for an accidental touch, the system must assign a key to the event, and for a missing touch, the system cannot insert an extra key-press. On the other hand, our motion model generates a probability distribution of which key was pressed (including no-key) at every frame of the tracking, which provides the decoder flexibility to insert or remove key-presses to accommodate the language model prior (Figure 7).

In order to disentangle the benefit of continuous motion decoding versus discrete contact decoding, we filter our test dataset similarly to our training set to contain only samples where we know the correspondences between contact events and ground truth character labels.In this filtered corresponded contacts dataset, there are no samples with missing or spurious contacts. Any remaining errors from the contact-based methods can only be due to the richness of the input signal.

Repeating the evaluation of the previous two baselines on this filtered corresponded contacts dataset, we found a substantial drop in the mean UER for all methods (Table 1), suggesting that removing spurious or missing key-presses makes for an easier dataset. However, the decrease is greater for the contact-

	Gaussian	Per Finger Gaussian	Hand Motion
Unfiltered UER	5.66 (2.70)	4.76 (2.64)	2.38 (1.77)
Filtered UER	4.84 (1.90)	3.97 (1.84)	2.22 (1.24)
% Decrease	14.6 (29.5)	16.6 (30.5)	6.72 (29.9)

Table 1. Comparison of mean (median) accuracy across decoding strategies, and the deltas when only samples with corresponded contacts are evaluated.

Test Phrase	Gaussian Contact	Per Finger Gaussian Contact	Hand Motion Decoding
a) PHYSICS AND CHEMISTRY ARE HARD	PHTWUCS AND CHWNUSTEYNARE HARE	PHYWUCS AND CHEMISTRY ARE HARE	PHYSICS AND CHEMISTRY ARE HARD
b) HAVE A GOOD WEEKEND	GAVE A GOOD SEREND	GADE A GOOD WEEEND	HAVE A GOOD WEEKEND
c) QUICK THERE IS SOMEONE KNOCKING	QJICK THERE IS SOMEONE KNOCMING	QJICK THERE IS SOMEONE KNOCKING	QUICK THERE IS SOMEONE KNOCKING
d) THE DOW JONES INDEX HAS RISEN	THE DOWN JONES INDEX HAS ROSEN	THR DOWN JONES INDEX HAS RISEN	THE DOW JONES INDEX HAS ROSE
e) I WATCHED BLAZING SADDLES	I WATCHED BLAZING SADDLES	I WATCHED BLAZING SADDLES	I WATCHED BLAZING SALES

Table 2. Sample phrases decoded with contact decoding baselines and our method (Hand Motion Decoding). In some phrases (a), we see the benefit of finger identity information. In others (c), using the trajectory of the finger allows for more accurate character classification. Operating on a continuous stream of hand motion allows our method to vary the number of characters (b,d,e) while contact-based decoding can only generate the same number of characters as the number of contacts.

based methods, which makes the differences between hand motion decoding and the two baselines no longer statistically significant ($p = 0.300$ for Gaussian and $p = 0.109$ for Per Finger Gaussian) according to a Wilcoxon signed rank test. This suggests that continuous hand motion decoding is better able to cope with the mistakes in the unfiltered dataset. Still, the remaining discrepancy in descriptive statistics between hand motion decoding UER and both baselines suggests that finger trajectory information is helpful for further disambiguating typed text. Note that while more sophisticated contact-based decoders such as the one described in Velocitap [25] can add or remove characters during decoding, they would not have access to finger trajectory information.

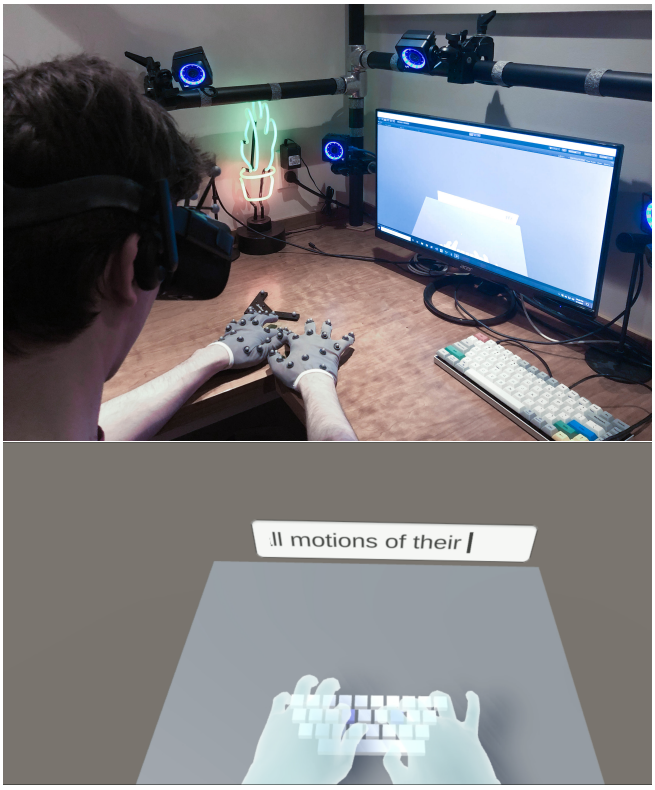

Figure 9. Interactive text input in virtual reality using using hand-tracking and a calibrated surface, with a virtual keyboard composed of standard sized 19mm keys.

Performance and interactive run-time

Our motion model network is compact, containing just 180KB of weights, enabling efficient evaluation on a GPU. Our language model consists of 19MB of weights. We are able to run both models, in addition to decoding, at interactive rates (i.e. 120 Hz) on a PC with an Nvidia RTX 2080Ti graphics card. These models were not optimized for compute and can be further accelerated using modern distillation and quantization techniques for neural networks [22].

We created a toy text entry application with an Oculus Rift and the Unity game engine to demonstrate real-time touch typing on a flat surface in VR using hand-tracking (Figure 9).

In this interactive setting we constrained our beam search decoder to force convergence for any predictions older than 6 frames (0.1s) causing all beams to have a common prefix. We only rendered text in the common prefix of all beams, effectively imposing a fixed 0.1s delay. This prevented any visible retroactive changes and maintained a controlled latency, while allowing corrections of the most recent 1-2 characters for most typists. While this serves as a proof of concept for interactive utility, our system still lacks many fundamental features required of any practical text input system, such as backspace or punctuation.

CONCLUSION

Typing on any flat surface without the need to bring a physical keyboard would provide a valuable addition to AR/VR interaction. In this paper we take a major step towards realizing the viability of such a system by demonstrating decoding of text from hand motion captured with a high quality hand-tracking system. We have shown on a 20 person dataset that touch-typists can transfer their skills typing on a physical keyboard to typing on a flat surface, reaching comparable typing speeds (73WPM) while retaining an uncorrected error rate of less than 2.4%. We achieved this result through the introduction of a novel motion model mapping hand motion to text, represented as a temporal neural network, and the application of beam search decoding combined with a modern neural language model. We also show that our hand-tracking-based decoder can produce significantly lower error than two baselines using contact-based text decoding (with and without finger identity information).

Many challenges remain to making this system practical for general text input. We will need to handle a lower quality

hand-tracking input signal from mobile headsets, or alternatively improve the state-of-the-art of egocentric hand-tracking today. Our system currently does not support typing of non-dictionary words effectively. For this work, we trained per-user neural motion models over an hour's worth of typing samples and will need a more user-friendly version of user calibration. When we attempted to train a single between-users model on the combined training sets of all users, the performance did not match the user specific models (UER rose from 2.4% to 3.9%, which a Wilcoxon signed rank test found to be significant, $p = 0.001$). We will need to explore a more efficient user calibration process in future work. Our current experiments provide no visual feedback during typing, and more exploration of what visual feedback to offer, e.g., along the lines of Velocitap [25] is needed. Finally, we are only decoding the lower case letters of the alphabet on our keyboard. We will need to handle other keys, including backspace, numbers and symbols to support general interactive text editing.

REFERENCES

[1] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, Jie Chen, Jingdong Chen, Zhijie Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Ke Ding, Niandong Du, Erich Elsen, Jesse Engel, Weiwei Fang, Linxi Fan, Christopher Fougner, Liang Gao, Caixia Gong, Awni Hannun, Tony Han, Lappi Johannes, Bing Jiang, Cai Ju, Billy Jun, Patrick LeGresley, Libby Lin, Junjie Liu, Yang Liu, Weigao Li, Xiangang Li, Dongpeng Ma, Sharan Narang, Andrew Ng, Sherjil Ozair, Yiping Peng, Ryan Prenger, Sheng Qian, Zongfeng Quan, Jonathan Raiman, Vinay Rao, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Kavya Srinet, Anuroop Sriram, Haiyuan Tang, Liliang Tang, Chong Wang, Jidong Wang, Kaifu Wang, Yi Wang, Zhijian Wang, Zhiqian Wang, Shuang Wu, Likai Wei, Bo Xiao, Wen Xie, Yan Xie, Dani Yogatama, Bin Yuan, Jun Zhan, and Zhenyao Zhu. 2016. Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin. In *Proceedings of The 33rd International Conference on Machine Learning*, Vol. 48. PMLR, New York, New York, USA, 173–182. http://proceedings.mlr.press/v48/amodei16.html

[2] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *CoRR* abs/1803.01271 (2018). http://arxiv.org/abs/1803.01271

[3] Vivek Dhakal, Anna Maria Feit, Per Ola Kristensson, and Antti Oulasvirta. 2018. Observations on Typing from 136 Million Keystrokes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, New York, NY, USA, Article Paper 646, 12 pages. DOI:http://dx.doi.org/10.1145/3173574.3174220

[4] John J. Dudley, Hrvoje Benko, Daniel Wigdor, and Per Ola Kristensson. 2019. Performance Envelopes of Virtual Keyboard Text Input Strategies in Virtual Reality. In *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 289–300.

[5] John J. Dudley, Keith Vertanen, and Per Ola Kristensson. 2018. Fast and Precise Touch-Based Text Entry for Head-Mounted Augmented Reality with Variable Occlusion. *ACM Trans. Comput.-Hum. Interact.* 25, 6, Article Article 30 (Dec. 2018), 40 pages. DOI: http://dx.doi.org/10.1145/3232163

[6] Anna Maria Feit, Daryl Weir, and Antti Oulasvirta. 2016. How We Type: Movement Strategies and Performance in Everyday Typing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. Association for Computing Machinery, New York, NY, USA, 4262–4273. DOI: http://dx.doi.org/10.1145/2858036.2858233

[7] Joshua Goodman, Gina Venolia, Keith Steury, and Chauncey Parker. 2002. Language Modeling for Soft Keyboards. In *Proceedings of the 7th International Conference on Intelligent User Interfaces (IUI '02)*. Association for Computing Machinery, New York, NY, USA, 194–195. DOI: http://dx.doi.org/10.1145/502716.502753

[8] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *Proceedings of the 23rd International Conference on Machine Learning (ICML '06)*. Association for Computing Machinery, New York, NY, USA, 369–376. DOI: http://dx.doi.org/10.1145/1143844.1143891

[9] Alex Graves and Navdeep Jaitly. 2014. Towards End-To-End Speech Recognition with Recurrent Neural Networks. In *Proceedings of the 31st International Conference on Machine Learning*, Vol. 32. PMLR, Bejing, China, 1764–1772. http://proceedings.mlr.press/v32/graves14.html

[10] Asela Gunawardana, Tim Paek, and Christopher Meek. 2010. Usability Guided Key-Target Resizing for Soft Keyboards. In *Proceedings of the 15th International Conference on Intelligent User Interfaces (IUI '10)*. Association for Computing Machinery, New York, NY, USA, 111–118. DOI: http://dx.doi.org/10.1145/1719970.1719986

[11] Aakar Gupta, Cheng Ji, Hui-Shyong Yeo, Aaron Quigley, and Daniel Vogel. 2019. RotoSwype: Word-Gesture Typing Using a Ring. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, Article Paper 14, 12 pages. DOI: http://dx.doi.org/10.1145/3290605.3300244

[12] Shangchen Han, Beibei Liu, Robert Wang, Yuting Ye, Christopher D. Twigg, and Kenrick Kin. 2018. Online Optical Marker-Based Hand Tracking with Deep Labels.

ACM Trans. Graph. 37, 4, Article Article 166 (July 2018), 10 pages. DOI: http://dx.doi.org/10.1145/3197517.3201399

[13] Awni Y. Hannun, Andrew L. Maas, Daniel Jurafsky, and Andrew Y. Ng. 2014. First-Pass Large Vocabulary Continuous Speech Recognition using Bi-Directional Recurrent DNNs. *arXiv preprint arXiv:1408.2873* (2014). http://arxiv.org/abs/1408.2873

[14] Niels Henze, Enrico Rukzio, and Susanne Boll. 2012. Observational and Experimental Investigation of Typing Behaviour Using Virtual Keyboards for Mobile Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. Association for Computing Machinery, New York, NY, USA, 2659–2668. DOI: http://dx.doi.org/10.1145/2207676.2208658

[15] Hongzhao Huang and Fuchun Peng. 2019. An Empirical Study of Efficient ASR Rescoring with Transformers. *arXiv preprint arXiv:1910.11450* (2019).

[16] Brent Edward Insko. 2001. *Passive Haptics Significantly Enhances Virtual Environments*. Ph.D. Dissertation. Advisor(s) Brooks, Frederick P. AAI3007820.

[17] Per-Ola Kristensson and Shumin Zhai. 2005. Relaxing Stylus Typing Precision by Geometric Pattern Matching. In *Proceedings of the 10th International Conference on Intelligent User Interfaces (IUI '05)*. Association for Computing Machinery, New York, NY, USA, 151–158. DOI:http://dx.doi.org/10.1145/1040830.1040867

[18] Minkyung Lee and Woontack Woo. 2003. ARKB: 3D vision-based Augmented Reality Keyboard. In *Online Proceeding of the 13th International Conference on Artificial Reality and Telexistence*.

[19] Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Asian Federation of Natural Language Processing, Taipei, Taiwan, 986–995. https://www.aclweb.org/anthology/I17-1099

[20] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019). http://arxiv.org/abs/1907.11692

[21] Anders Markussen, Mikkel Rønne Jakobsen, and Kasper Hornbæk. 2014. Vulture: A Mid-Air Word-Gesture Keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. Association for Computing Machinery, New York, NY, USA, 1073–1082. DOI: http://dx.doi.org/10.1145/2556288.2556964

[22] Antonio Polino, Razvan Pascanu, and Dan Alistarh. 2018. Model compression via distillation and quantization. In *International Conference on Learning Representations*. https://openreview.net/forum?id=S1XolQbRW

[23] R. William Soukoreff and I. Scott MacKenzie. 2003. Metrics for Text Entry Research: An Evaluation of MSD and KSPC, and a New Unified Error Metric. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*. Association for Computing Machinery, New York, NY, USA, 113–120. DOI: http://dx.doi.org/10.1145/642611.642632

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 5998–6008. http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[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. DOI: http://dx.doi.org/10.1145/2702123.2702135

[26] Daryl Weir, Henning Pohl, Simon Rogers, Keith Vertanen, and Per Ola Kristensson. 2014. Uncertain Text Entry on Mobile Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. Association for Computing Machinery, New York, NY, USA, 2307–2316. DOI: http://dx.doi.org/10.1145/2556288.2557412

[27] Zhican Yang, Chun Yu, Xin Yi, and Yuanchun Shi. 2019. Investigating Gesture Typing for Indirect Touch. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 3, Article Article 117 (Sept. 2019), 22 pages. DOI: http://dx.doi.org/10.1145/3351275

[28] Xin Yi, Chun Yu, Mingrui Zhang, Sida Gao, Ke Sun, and Yuanchun Shi. 2015. ATK: Enabling Ten-Finger Freehand Typing in Air Based on 3D Hand Tracking Data. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology (UIST '15)*. Association for Computing Machinery, New York, NY, USA, 539–548. DOI: http://dx.doi.org/10.1145/2807442.2807504

[29] Suwen Zhu, Tianyao Luo, Xiaojun Bi, and Shumin Zhai. 2018. Typing on an Invisible Keyboard. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, New York, NY, USA, Article Paper 439, 13 pages. DOI: http://dx.doi.org/10.1145/3173574.3174013

[30] Suwen Zhu, Jingjie Zheng, Shumin Zhai, and Xiaojun Bi. 2019. I'sFree: Eyes-Free Gesture Typing via a Touch-Enabled Remote Control. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, Article Paper 448, 12 pages. DOI:http://dx.doi.org/10.1145/3290605.3300678