Typealike: Near-Keyboard Hand Postures for Expanded Laptop Interaction

NALIN CHHIBBER, School of Computer Science, University of Waterloo, Canada
HEMANT SURALE, School of Computer Science, University of Waterloo, Canada
FABRICE MATULIC, Preferred Networks Inc., Japan
DANIEL VOGEL, School of Computer Science, University of Waterloo, Canada

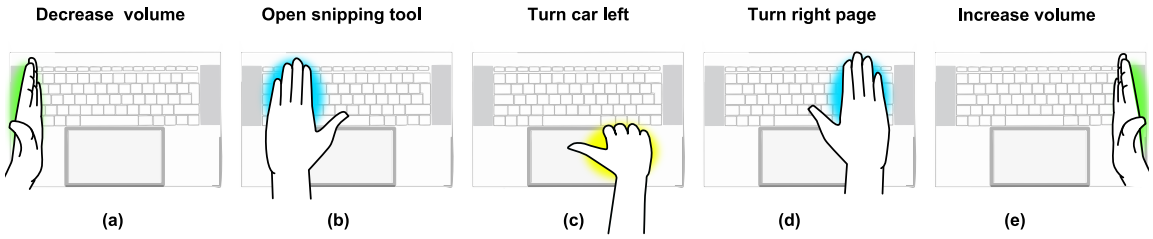

Fig. 1. Typealike postures are postures formed using the left or right hand, an open or closed hand form, and different wrist orientations, all further distinguished by hand position when on, just beside, or just below the keyboard. Different postures can trigger different interactions, for example: (a) an open left hand at 90° wrist rotation beside the keyboard can decrease the volume; (b) an open left hand at 0° on the keyboard can open a screen capture snipping tool; (c) a closed right hand 0° posture below the keyboard could turn the car in a racing game; (d) an open right hand at 0° on the keyboard could advance a document page; and (e) a right open 90° hand posture beside the keyboard could increase the volume.

We propose a style of hand postures to trigger commands on a laptop. The key idea is to perform hand-postures while keeping the hands on, beside, or below the keyboard, to align with natural laptop usage. 36 hand-posture variations are explored considering three resting locations, left or right hand, open or closed hand, and three wrist rotation angles. A 30-participant formative study measures posture preferences and generates a dataset of nearly 350K images under different lighting conditions and backgrounds. A deep learning recognizer achieves over 97% accuracy when classifying all 36 postures with 2 additional non-posture classes for typing and non-typing. A second experiment with 20 participants validates the recognizer under real-time usage and compares posture invocation time with keyboard shortcuts. Results find low error rates and fast formation time, indicating postures are close to current typing and pointing postures. Finally, practical use case demonstrations are presented, and further extensions discussed.

Additional Key Words and Phrases: Gestures; Typing; Dataset; Desktop/Laptop Interaction

ACM Reference Format:
Nalin Chhibber, Hemant Surale, Fabrice Matulic, and Daniel Vogel. . Typealike: Near-Keyboard Hand Postures for Expanded Laptop Interaction. 1, 1 (September), 20 pages.

Authors' addresses: Nalin Chhibber, nalin.chhibber@uwaterloo.ca, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada; Hemant Surale, hsurale@uwaterloo.ca, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada; Fabrice Matulic, fmatulic@preferred.jp, Preferred Networks Inc. Tokyo, Japan; Daniel Vogel, dvogel@uwaterloo.ca, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada.

. Manuscript submitted to ACM

1 INTRODUCTION

Keyboard-based interactions are crucial to efficiently operate conventional graphical interfaces on a desktop or laptop. In addition to standard text entry, a keyboard also enables shortcuts and hotkeys to trigger frequently used commands and provides a way to modify touchpad input (such as switching from selecting to panning by holding down SPACE). Various methods to further expand the physical keyboard input space have been proposed. For example, enabling new kinds of touch interactions on the keys [6, 11, 29, 35] or tracking mid-air hand gestures above the keyboard [22, 23, 36]. Both focus on integrating continuous input, like pointing and swiping, into a physical keyboard. Another family of methods use static hand gestures, often called "hand postures", to issue discrete commands much like a short-cut key. A common approach is to recognize unique postures at the moment a key is pressed [7, 40, 41]. This simplifies recognition since each keypress acts as a delimiter to trigger posture recognition, and combining which key was pressed with different postures creates a large input space. One risk is that eyes-free touch typing already uses automatic hand movements that combine certain hand positions with each keypress, so overloading how keys are pressed could disrupt natural typing flow. In addition, not all postures are comfortable to hold while also pressing a key.

We propose a related, but different approach where hand postures are performed without a simultaneous keypress delimiter and can be recognized using only the front camera of a laptop fitted with a mirror reflector. The postures use simple hand forms close to those used when typing, and they can be performed near the keyboard without lifting the wrist off the laptop case. The intent is to make posture actions similar to normal typing for increased flow, speed, and comfort, while remaining sufficiently unique to be reliably recognized. For these reasons, we refer to our approach as Typealike postures. We explore 36 posture variations: left or right hands, open hand or closed fist forms, three wrist rotation angles, and three surface locations on or around the laptop keyboard where people already place their hands when typing, resting, or when using the touchpad. Like previous approaches, the goal is to further extend the input space for a physical keyboard. Our postures can be used to invoke a command or change interaction modes, and they could co-exist with previously proposed gesture or posture approaches, including those that tie hand posture actions with a keypress.

We evaluate our approach by answering three questions: Is this style of postures preferable by users? Can they be recognized reliably? Are they quick and easy to perform? These questions are answered with a three-step methodology. First, a 30-participant formative study shows all postures are preferable, with some less so and therefore better suited for less frequent operations. Next, a large and diverse dataset of hand posture images from the formative study is used to train a deep learning model, it achieves 97.4% classification accuracy. Finally, a second study with 20 participants uses this model to examine posture performance with keyboard shortcuts serving as a relative performance baseline. The results show the postures are faster than shortcuts in many cases, especially during "mode-in" formation. We provide our code, experiment trial data, image dataset, and trained model for replication and extensions[1]. Overall, our work contributes:

- A set of near-keyboard hand postures that are acceptable to users, recognized reliably, and fast to perform.
- A dataset of nearly 350K posture images captured under different lighting conditions and two keyboard backgrounds.

2 RELATED WORK

There are many approaches to create an expanded or alternative keyboard input space. Some look beyond the keyboard, like using other sensors to capture touch and gestures beside the laptop [15, 21, 38] or methods that expand touchpad

[1]The supplementary material will be released on the following link: https://ANONYMIZED_FOR_REVIEW

interaction, such as gesture typing to trigger commands [8, 9]. Others change the fundamental form of the keyboard itself, like replacing a physical keyboard with a touchscreen, such as TapBoard 2 [16]. Our focus is on previous work that is conceptually, or literally, tied to using hand gestures (including static hand postures, hand forms, or hand motions) on or near a physical keyboard.

2.1 Hand Gestures on the Keyboard Surface

Even the act of pressing multiple keys for conventional shortcuts requires specific hand postures, though the fingers or hand form used are not directly sensed. A generalized extension of keyboard shortcuts is Au et al.'s method of pressing and releasing multiple keys to form chords that trigger commands [2]. Certain hand postures are used, but not explicitly controlled or sensed.

What is more directly related to hand gestures, are techniques to sense larger gestural interactions across the entire keyboard. Zhang and Li detect spatio-temporal features when multiple keys are stroked on a physical keyboard, like a touch screen [39]. Taylor et al. [34] detect a large set of motion gestures interlaced with key presses using a matrix of infrared proximity sensors embedded between the keys. Touch&Type [12] and FlickBoard [35] instrument the entire physical keyboard with capacitive sensing to enable pointing operations across multiple keys. Block et al. [6] push keyboard customization even further by adding capacitive touch sensing as well as a display on the surface of individual keys. The primary goal is to use hand gestures to bring continuous swiping or pointing-like operations into the keyboard, reminiscent of the motivation behind integrated isometric joysticks such as IBM Trackpoint [30]. In contrast, our goal is to trigger discrete commands quickly using static hand postures.

2.2 Hand Gestures Above the Keyboard

Tracking hand movements above the keyboard in mid-air has been a popular approach. Space-Top [18] detects finger touch and hand gestures above the keyboard using a downward facing depth camera. AirMouse [23], Fingermouse [22], and Ramos et al. [28] also use cameras with computer vision to track a pointing finger above the keyboard, enabling mouse-like interaction. Wilson [36] examined a wider set of cursor control and related continuous control interactions using one and two-handed thumb and finger pinch gestures, also captured using a top-down camera. The goal of these projects is to bring continuous pointing-like interaction closer to the keyboard. Using unsupported mid-air gestures for a full range of motion could result in muscle fatigue. Our goal is to trigger commands with static postures that can be performed very close to the keyboard surface.

2.3 Hand Gestures while Pressing a Key

A general approach is to overload a keypress action by recognizing how one or more keys were pressed. Additional features of a keypress can be sensed using pressure, touch position, and force [3, 11, 20, 25], for example. These techniques are not explicitly about hand gestures or postures, but they can be used to detect continuous gestures performed on individual keys. For example, the PreSense Keypad [29] uses touch and pressure sensors to recognize different positions and pressures of the finger contact to change what a keypress means. GestAKey [31] adds 2D capacitive sensing to each keycap to detect small touch gestures to change or modify a command associated with pressing a key.

The most related examples are those that either directly or indirectly use different hand postures to press keys in unconventional ways to trigger different commands. Finger Aware Shortcuts [41] use different fingers, left or right hands, and either an open or closed hand, when pressing a key to trigger different commands. Sensing is done through the

built-in laptop camera using a mirror reflector, but relies on the keypress event to trigger recognition. The recognition method uses heuristics and the $1 recognizer [37] on the hand contour, but only 3 postures are experimentally tested. No recognition rates are reported. Similarly, FingerArc[40] uses a closed hand with a thumb-out posture to press keys, but with the purpose of providing interactive visual guidance for learning posture-to-command mappings. Recognition is based on retro-reflective markers attached to the hand. Buschek et al. [7] examined a method for adding continuous control to keypress actions by tracking arm and wrist rotation gestures using IMU sensors. We also incorporate the idea of wrist rotation in our discrete hand posture set. We adopt the camera and mirror reflector method from Finger Aware Shortcuts and combine their open and closed hand forms with the thumb-out posture from FingerArc. We also incorporate the wrist rotation from Buschek et al.. However, we tackle the different problem of designing and recognizing discrete hand postures that do not rely on a simultaneous keypress.

3 TYPEALIKE POSTURE SET

Our posture set was chosen to support our goal of a discrete command invoking method that aligns with natural laptop usage. The postures are variations of hand posture styles, all based on three design considerations:

Recognition without a key press. This "disentangles" command execution from key-press events, so postures will not conflict with touch-typing hand motor behaviour when pressing keys. Upholding this design consideration means postures used for commands must be unique from those used when typing, using the touch-pad, or resting. They should also have distinctive features to delimit the posture action.

Performed on or near the keyboard surface. Posture hand shapes, and any movement to form postures, should be compatible with maintaining a near-resting arm posture where the wrist is in constant contact with the laptop case. This should minimize transition time from typing to command-triggering postures, and be more comfortable than forming postures in mid-air or on adjacent desk surfaces. This is analogous to the goal of minimizing pointing device homing time by enabling pointing gestures on the keyboard surface (e.g. [12, 35]).

Simple and easy to perform. While the postures should be close to the keyboard and distinguishable for detection, they should naturally be comfortable to perform for the user to make them practical.

3.1 Posture Parameters and Variations

With these design considerations in mind, we created a style of postures that use open or closed hand forms to be *simple and fast to perform*, but with a slightly extended thumb to appear unique for *recognition without a keypress*. To support a large number of postures, we add two further characteristics for variations based on wrist rotation angles and the surface location where the posture is performed. The hand forms, the kinematics of wrist rotation, and the distinct locations on or near the keyboard, means all postures can be *performed on or near the keyboard surface*. To further expand the input space, all postures can be performed with either hand. Our set of 36 postures are fully defined by variations along four parameters, illustrated in Figure 2 and described below:

- *Hand (dominant, non-dominant)*: all postures can be performed with either hand. This is a distinguishing feature during recognition and may be used for different command mappings.
- *Form (open, closed)*: all postures use either a predominantly open or closed hand form. In both cases the thumb is extended. In practice, these postures can be performed in a relaxed way, without forcibly clenching the fist, forming a fully flat hand, or rigidly extending the thumb.

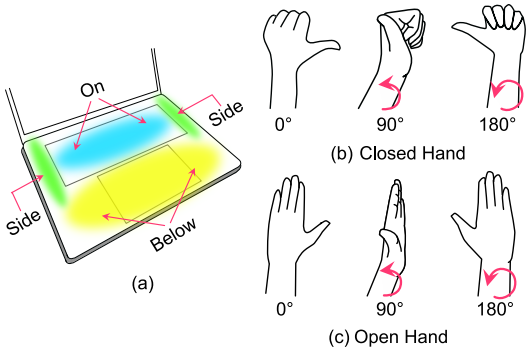

Fig. 2. Posture style and parameters: (a) *Surface* where posture is performed (on, below, or beside keyboard); (b) Form of hand used in posture (open or closed); (c) Orientation of wrist used in posture (0°, 90°, and 180°). Note all postures use an extended thumb, and can be performed with either hand.

- *Orientation (0°, 90°, 180°)*: all postures can be performed with the palmar side of the hand touching the surface (0°), with the side edge of the hand touching the surface (90°), or with the dorsal side touching the surface (180°). We chose three orientations over this range for simplicity and distinguishability for robust recognition.
- *Surface (on, beside, below)*: postures can be performed on the central keyboard area, beside the keyboard (either on the left or right side depending on hand used), or immediately below the keyboard. These correspond to locations where users already place their hands when typing, resting, or using the touchpad.

While this set of postures was guided by our design considerations, there are variations in the degree to which these considerations are upheld. For example, a larger orientation requires more wrist rotation, and therefore deviates more from normal typing postures and may be less preferred by users. Similarly, a closed form deviates more from normal typing postures than an open hand. In addition, setting interaction surfaces to be on-keyboard and below-keyboard could introduce false positives with normal typing or touchpad operation. These aspects are formally tested in the experiments described below.

4 EXPERIMENT 1: PREFERENCE AND DATA GATHERING

The purpose of this experiment was to identify the most comfortable and preferred postures and gather training data to create a recognizer. All experiments were approved by our institution's research ethics board.

4.1 Participants

We recruited 30 participants: 10 female and 20 male; 19 to 31 years old (M=24.6, SD = 2.4); 23 right-handed. All reported extensive experience with laptop computers with daily usage between 4 to 16 hours (M=7.4, SD = 3.2). A pre-study typing test found a mean typing speed 38 words-per-minute (SD = 7.8) with a mean error rate of 5.8% (SD = 3.4).

4.2 Apparatus

The experiment was performed on a 2018 15-inch Macbook Pro with QWERTY keyboard with experiment code written in Java using the Processing library. Video of the hands and keyboard was recorded at 30 FPS in 1280×720 resolution using the built-in laptop camera and a mirror reflector (Figure 3a). The video logger was written in Python with the OpenCV library. Participants were instructed to avoid tilting the screen during the experiment for consistent data

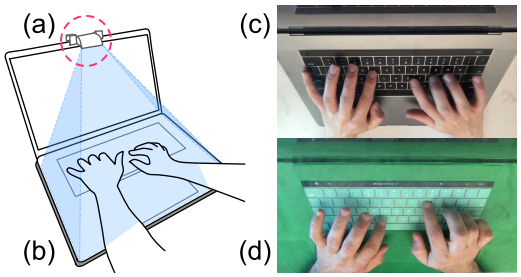

Fig. 3. Experiment apparatus: (a) mirror reflector mounted on built-in laptop camera; (b) approximate hand capture volume; (c) example camera view; (d) example camera view with green keyboard cover.

capturing. Note that a real deployment could use an ultra wide-angle lens, a pop-out mirror, or a rotating camera like the one used in some laptops (e.g. Lenovo N22).

We captured our data under different conditions to account for variations between laptops and environments and make our recognizer more generalizable. First, we used a green keyboard and laptop surface cover ~~cover~~ to be replaced with different backgrounds via chromakeying when processing the captured frames. The green keyboard was used for 20 participants, while the remaining 10 used the laptop without the cover. Second, we changed the lighting conditions for each of the 3 blocks in the experiment. In one block, an ambient diffuse light source was used which resulted in no pronounced shadows. In the other blocks, a light source was placed on the left and right side of the laptop to create side shadows.

4.3 Task

We create simple game-like tasks representing one of four types of interaction: typing, clicking, shortcuts, and postures. These various tasks establish a realistic context and meaningful baseline when rating the comfort and preferences of postures. We need examples of hand postures representing normal laptop activity to expose these cases to our recognizer in order to avoid false positives. Each trial lasted a maximum of 6 seconds, with the passing time indicated by a red ball "falling" from the top to the bottom of the display. Upon completion of each trial, a sound was played to indicate either success or failure. Participants were instructed to complete tasks as quickly and accurately as possible.

The *typing* task displayed a five-letter English word randomly sampled from the NLTK brown corpus [5]. Participants typed the exact characters and pressed return to complete the trial (Figure 4c). If they made a mistake, the input field was reset, and they tried again. The *shortcut* task displayed a prompt showing a modifier key, "Shift", "Ctrl", or "Alt" and a letter, picked randomly from a set of predefined shortcuts. Participants pressed the modifier and letter key to complete the trial (Figure 4e). Mistakes were handled like for typing. The *clicking* task displayed a button at a random location, and participants used the touchpad to move the cursor and click on the target to complete the trial (Figure 4d). If they missed the target, they had to try again. The *posture* task displayed an image representing one of the 36 postures (Figure 4b). Participants performed the posture with the required hand, form, orientation, and surface until the 6-second trial ended. In order to increase the diversity of posture data for training, we animated the rendered hand in the prompt to wiggle a small amount. Participants were told to mimic this motion while maintaining the posture. This increased engagement during the task, but more importantly made sure different variations of the same posture were recorded for our dataset. Experiments were monitored by a human observer to ensure participants were performing postures correctly given the task prompt.

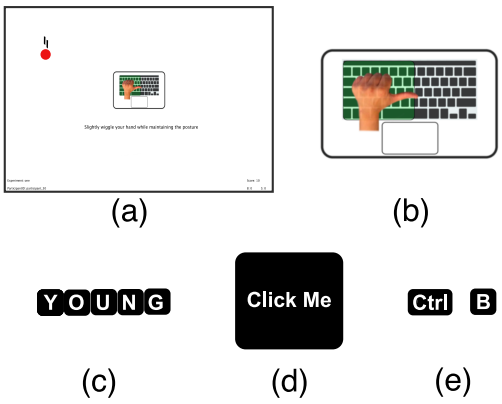

Fig. 4. Experiment tasks: (a) full screen view showing falling trial timer ball; (b) close up of a posture task prompt; (c) close up of a typing task prompt; (d) close up of a clicking task prompt, and (e) close up of a shortcut task prompt.

4.4 Design

The experiment design was within-subject with three blocks of trials. In each block, there were 36 trials of posture tasks, one for each posture in the set. Postures were presented in random order. For analysis, independent variables represent the posture parameters: HAND (dominant, non-dominant), FORM (open, closed), ORIENTATION (0°, 90°, 180°), and SURFACE (on, below, beside). A reference task (typing or clicking) was included between each posture trial, adding 72 trials for each block. Shortcuts were considered a different form of typing and were included within the typing reference task. These were selected pseudo randomly to ensure each type of REFERENCE task was equally present.

In summary, there were 3 BLOCKS × 2 HANDS × 2 FORMS × 3 ORIENTATIONS × 3 SURFACES × 2 reference tasks per posture = 216 trials per participant.

4.5 Procedure

Each session proceeded as follows. First, participants completed a one-minute typing test on a 35 letter-long pangram with punctuation marks, covering a range of keys. This provided the typing speed estimates reported above. Next, participants were shown how to perform the different posture variations, and the experiment tasks were explained. Participants completed practice trials of all tasks for approximately 5 minutes. Then, the main part of the experiment began where participants performed three blocks of trials where their hands were captured while typing, not typing, and when performing postures. All input events were logged and video of the entire session recorded. Participants were given the opportunity to take breaks between blocks. At the end of the experiment, participants rated each posture for comfort and overall preference. The full procedure took approximately 25 minutes.

4.6 Hand Image Dataset Extraction

Recall that one of the goals of the experiments was to gather image data to train a recognizer. For this purpose, the video stream was first synchronized with the experiment event logs using observable input events such as keystrokes. Once synchronized, individual frames were extracted from the task trials and other experiment segments. To make sure we have completely formed postures and not partially-forming movements, we only extracted frames from the last 2s of the trial. For non-posture examples, we extracted frames to form typing and non-typing classes. The typing class represented hands from the typing and shortcut tasks, as determined by the key events. The non-typing class

represented hands during the clicking task, as determined by the touchpad events, and when resting on or around the laptop surface in between trials and during breaks. The best segments to use for the latter were determined by examining video logs, for example participants typically rested their hands in different positions during the break between blocks. Note that even frames where no hands were visible were important for training a non-typing class. Overall, online data augmentation and our frame selection technique ensured that extracted frames were clear and non-overlapping with each other. This was also visually verified and outliers were removed manually.

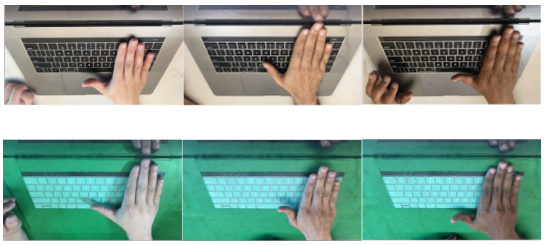

Fig. 5. Sample images of open right hand with 0°wrist rotation on the keyboard surface captured under different lighting conditions and background.

Our three blocks of repeated trials with 60 frames per posture trial including small hand movements for posture variability, three different lighting conditions, and two different backgrounds generate an automatically annotated dataset that is both diverse and large (see Figure 5 for examples of frames). There are nearly 350K labelled frames in total, which is comparable or larger than related hand image datasets [4, 26, 27].

4.7 Results

After completing all three blocks of trials, participants rated each posture for *Comfort* and *Overall Preference*, each using a 5-point interval scale. They were encouraged to perform the postures again while rating. Due to non-normal distributions in the rating data, non-parametric Friedman and Wilcoxon signed-rank tests are used for main effects and pairwise comparison.

4.7.1 Comfort Rating. The question asked participants to consider both "ease of use" and "hand fatigue" when selecting a rating. The average *Comfort* rating across all postures was 3.25 (SD = 1.32), with individual ratings ranging from 1.83 (SD = 1.05) for the 'non-dominant 180° close-hand beside the keyboard' to 4.36 (SD = 0.99) for the 'dominant 0° open-hand on the keyboard'. Examining the effect of each posture parameter, we found the highest 180° wrist orientation to be rated less comfortable than lower orientations, but no differences for hand, form, or surface parameters. All mean parameter ratings are above a neutral 3.0 rating, with the exception of 2.2 for postures with a 180° wrist orientation. This discomfort in orientation is likely due to greater divergence of rotated wrists from natural typing hand formations.

We report supporting statistical tests and descriptive statistics. Wrist ORIENTATION had a significant effect on *Comfort* ($\chi^2 = 407.83, df = 2, p < 0.01$), with post hoc tests showing ratings for 180° postures (M=2.2, SD = 1.1) are significantly lower than both 90° (M=3.7, SD = 1.1) ($W = 45817, Z = 15.19, p < 0.01, r = 0.566$) and 0° (M=3.8, SD = 1.1) ($W = 45208, Z = 15.57, p < 0.01, r = 0.580$). There was no significant pairwise difference between 0° and 90° wrist rotations. No significant effect was found for HAND ($\chi^2 = 13.49, df = 1, p = .240$) suggesting comparable comfort when using the dominant hand (M=3.3, SD = 1.3) or non-dominant hand (M=3.2, SD = 1.3). Similarly, there was no effect

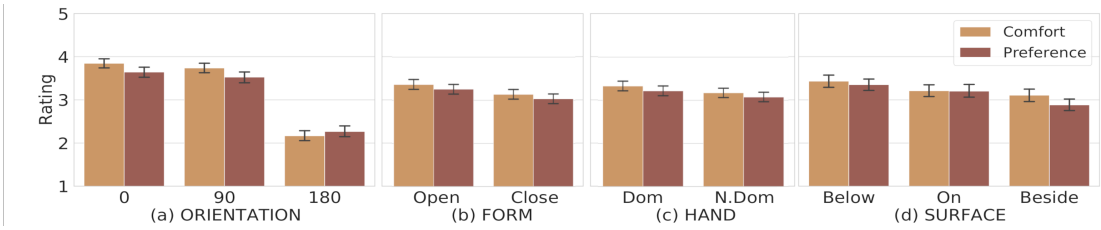

Fig. 6. Mean 'Comfort' and 'Overall preference' ratings for posture variations. Error bars in all graphs are 95% CI.

of FORM ($\chi^2 = 33.48, df = 1, p = .072$) suggesting comparable comfort for open hand (M=3.4, SD = 1.3) and closed hand (M=3.1, SD = 1.3) postures. And finally, there was no significant effect of SURFACE ($\chi^2 = 28.41, df = 2, p = .676$), again suggesting similar comfort levels for on the keyboard (M=3.2, SD = 1.3), below the keyboard (M=3.4, SD = 1.3), and beside the keyboard (M=3.1, SD = 1.4).

4.7.2 Preference Rating. The question asked the participant to rate each posture by considering their overall preference, including "comfort", "appropriateness", and "location". The average *Preference* rating across all postures was 3.14 (SD = 1.32), with individual ratings ranging from 2 (SD = 1.11) for the 'dominant 180° close beside keyboard' to 4.33 (SD = 0.99) for the 'dominant 0° open on keyboard'. Examining the effect of each posture parameter, we again found postures with a 180° wrist orientation to be less preferred than other orientations. In addition, participants preferred postures performed below the keyboard over postures performed beside the keyboard. All mean ratings are at or above a neutral 3.0 rating, with the exception of 2.3 for postures with a 180° wrist orientation and 2.9 for postures performed beside the keyboard. Comments from some participants suggested the surface beside the keyboard was considered too narrow to support their hand. This may have been the results of a too literal interpretation of "beside", our intention was that this surface could include some of the right or left end of the keyboard itself, creating a larger surface.

We report supporting statistical tests and descriptive statistics. There was a main effect of ORIENTATION on *Preference* ($\chi^2 = 311.24, df = 2, p < .0.01$). Post hoc test showed the maximum 180° orientation (M=2.3, SD = 1.2) was less preferred than orientations of 90° (M=3.5, SD = 1.2) or 0° (M=3.6, SD = 1.1) ($W = 36372, Z = 14.396, p < 0.05, r = 0.58$ and $W = 35102, Z = 13.749, p < 0.05, r = 0.57$ respectively). No significant pairwise difference was found between 0° and 90°. There was also a main effect of SURFACE on *Preference* ($\chi^2 = 44.71, df = 2, p < 0.01$). Post hoc tests showed postures performed below the keyboard (M=3.4, SD = 1.3) were rated higher than those beside the keyboard (M=2.9, SD = 1.3) ($W = 19422, Z = 6.52, p < 0.01, r = 0.19$). There was no significant differences with postures performed on the keyboard (M=3.2, SD = 1.3). There was no significant effect of HAND on *Preference* ($\chi^2 = 7.04$, df = 1,p > 0.05, r = 0.08), suggesting dominant hand postures (M=3.2, SD = 1.3) and non-dominant postures (M=3.1, SD = 1.3) are comparable. Finally, there was no effect of FORM on *Preference* ($\chi^2 = 26.67$, df = 1, p > 0.05), suggesting closed hand postures (M=3.0, SD = 1.28) and open hand postures (M=3.3, SD = 1.35) are comparable.

4.8 Summary

The results from our formative study show our set of typing-friendly command postures are largely acceptable. While some postures were preferred over others, even the lower rated postures found some appeal. For instance, many participants found a large 180° wrist rotation to be less comfortable and less preferred, but several suggested this variation could be useful due to its strong semantic relevance. For example, for commands setting a quantity, a direction, or sequences. 0° and 180° variations can provide a way to increase and decrease, move up or down, or navigate to

previous or next pages. Another use for these less preferred orientations may be for infrequent or critical commands, like 'delete' or 'quit'. For these reasons, we decide not to discard any postures at this stage and include all 36 posture variations to train our recognizer. We return to this discussion of design guidelines and appropriate mappings for different postures in the general discussion.

5 RECOGNITION USING DEEP LEARNING

In this section we describe the deep learning pipeline used to recognize these hand posture formations. It serves as a validation that a posture set designed to be similar to normal typing hand postures can be recognized reliably without an explicit keypress, especially considering potential false positives with similar looking hand shapes when typing, using the touchpad, or resting.

5.1 Model Architecture. Feature Selection, and Data Augmentation

We use a deep convolutional neural network (CNN) based on a ResNet-50 architecture [13] with weights pre-trained on the ImageNet dataset [10]. ResNet-50 was used over other architectures because it is popular for classifying natural RGB images and it could handle many characteristics of our setting: close-up partially visible hands, lens distortion, and the keyboard background.

Image frames are used exclusively as input features to the model. We experimented with time-based key-press events to assist with identifying the typing class, but found they were too dependent on individual typing speed, and did not cover situations when a user is prepared to type but not pressing a key. Training the recognizer on images of typing hands not only captures the intention to type, but also more robustly tests our hypothesis that our posture set uses hand formations that are unique enough to be distinguished from normal typing.

To increase the robustness of our network against background variations, we subtract the background from the hands in the images using chromakeying and replace it with random images downloaded from the Internet. This data augmentation step helps to teach the network to ignore the keyboard background, which is otherwise always the same. For all images, we apply further random data augmentations including varying brightness up to a 20% difference and adding some random jitter. These modifications further mitigate overfitting to the training dataset.

5.2 Training Protocol

The 30-participant dataset generated by Experiment 1 has 38 classes: 36 Typealike postures and two sets of regular hand poses corresponding to typing and non-typing actions. To create training and validation sets, data is split between participants to test classifier performance with unknown users. Good results in this case would mean that the model is sufficiently general and that in a real deployment, users would not be required to perform any calibration to use the postures. With 30 participants, we use 24 participants for training and hold 6 out for validation, which corresponds to a 80%:20% split. We perform a Monte Carlo cross-validation from scikit-learn 0.19.2 [24] with 10 random splits using this proportion, with the condition that the validation set includes data of 2 participants with the green keyboard and 4 with the regular keyboard.

Since we are using transfer learning on the ResNet, we first train the last fully connected layers of the network, while all other layers remain frozen. We use Smith's one-cycle policy [32], which varies the learning rate to achieve faster convergence. We use cross entropy as the loss function and output the error rate (the percentage of misclassified postures in the validation set) for each epoch. We train for 10 epochs using that configuration and a batch size of 40. After this first training sequence is complete, we unfreeze all layers of the network and train another 10 epochs with

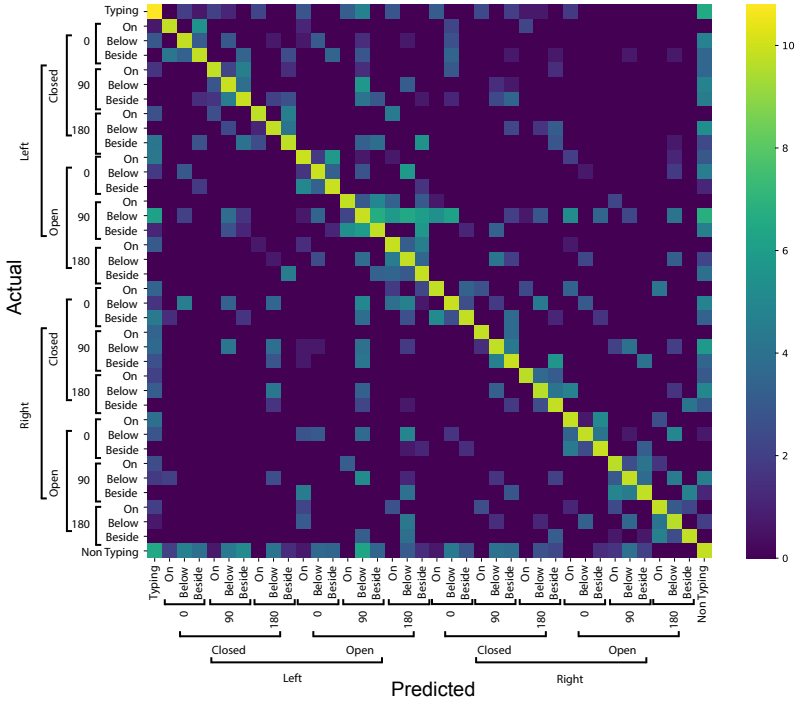

Fig. 7. Aggregate confusion matrix for all 38 classes with logarithmic colour mapping

differential learning rates between 10^{-6} and 10^{-5} distributed across the layers (with the lowest learning rate for the initial layers and the larger one for the latter layers). When this second stage is complete we record the lowest error rate output within those last 10 epochs. We also generate the confusion matrix at the last epoch.

We repeat the above procedure 10 times with different participant splits for our Monte Carlo cross-validation. This yields 10 error rates, which we average to obtain the overall accuracy of our recognizer. We further sum the obtained confusion matrices to obtain an aggregate confusion matrix (Figure 7).

5.3 Results

We obtain an average error rate of 2.6% (SD = 0.9%), which corresponds to a total accuracy of roughly 97.4%. The confusion matrix (Figure 7) shows that the recognition errors were primarily spread between "typing" and "non-typing" classes. These classes are mistaken for several types of postures (albeit with a relatively low probability), whereas confusion for other poses is more confined to similar postures in the same area of the keyboard. Taking a closer look at the data revealed that the automatic partitioning of images to create the dataset included a few false positive samples in the training set resulting in wrong labels. This happened to some extent for other postures as well. Correcting the labels of these samples and manually reviewing the dataset should therefore yield even higher recognition rates. For Experiment 2, we train a network on the entire dataset without holding out a validation set, since we want to use all the data to train the most robust model for inference.

Average posture recognition time using Python 3.6 and Fastai v2 [14] was 20ms which is as good as the processing speed of a human eye while distinguishing subsequent images. Since the video feed was captured at 30 FPS and only

the recognized labels were shared with the application in real-time through sockets, no additional lag was introduced due to the inference process.

6 EXPERIMENT 2: TECHNIQUE EVALUATION

The goal is to validate the recognizer in a real time interactive context, and also benchmark Typealike posture interaction using standard keyboard shortcuts as a relative baseline. Primary dependent measures are time, with special attention to the time required to transition between postures and typing and touchpad reference tasks. The tasks and protocol are similar to Experiment 1.

6.1 Participants

We recruited 20 new participants who did not participate in Experiment 1: 8 female, 12 male; 23 to 33 years old (M=27.25, SD = 3.17); 14 right handed, 6 left-handed. Self-reported daily computer usage ranged from 2 to 12 hours (M=8.35, SD = 2.75). Mean typing speed was 46 words-per-minute (SD = 9.3) with a mean error rate of 3.5% (SD = 2.4%). All were proficient typists, and had no trouble finding the location of shortcut keys.

6.2 Apparatus

The same code base as Experiment 1 was used to display tasks and log input events, and the same computer was used with a mirror reflector. In this experiment, no green keyboard cover was used and there were no manipulations to existing ambient room lighting. The recognizer ran as a separate Python process using OpenCV 4 to capture frames from the downward reflected laptop camera and scikit-learn 0.19.2 [24] to run the model in real time on each image frame. The recognizer code ran at 30 FPS on the same laptop, and communicated with the experiment task application using sockets.

6.3 Task

The typing, shortcut, and clicking tasks are identical to those used in Experiment 1, and the same 6-second trial timeline with the falling red ball was used. The posture task displayed a non-animated version of images used in Experiment 1, and instead of maintaining a posture for 6 seconds, the participant simply performed the required posture within the time limit. Like the other tasks, the trial ended when the correct posture was recognized, and the next trial immediately began. Participants were instructed to complete all tasks as quickly and accurately as possible.

6.4 Design

In each block, there were 36 trials of the posture task, one for each posture in the set. For analysis, independent variables represent the posture parameters: HAND (dominant, non-dominant), FORM (open, closed), ORIENTATION (0°, 90°, 180°), and SURFACE (on, below, beside). Different from Experiment 1, the keyboard shortcut task was treated as a primary SHORTCUT condition, and also had 36 task trials in each block.

The experiment design was within-subject with five blocks of measured trials. Each block had two sequences of trials: one sequence of 36 posture task trials alternating with a typing or clicking task trial and another sequence of 36 shortcut task trials also alternating with a typing or clicking task trial. The sequence order was counter-balanced across participants. The 36 posture task trials in each block covered all postures in randomized order, the same independent variables representing posture parameters are used in analysis: HAND (dominant, non-dominant), FORM (open, closed), ORIENTATION (0°, 90°, 180°), and SURFACE (on, below, beside). For shortcut tasks, we used a set of shortcut keys

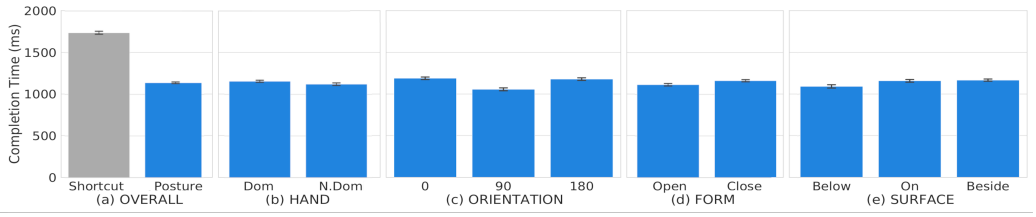

Fig. 8. Average Completion Time for shortcuts with respect to different (a) Wrist Orientations, (b) Hand Forms, (c) Hand involved, (d) Surface of the laptop supporting the hands, and (e) Overall Postures.

randomized across each block to measure motor performance and relationship to typing. Like Experiment 1, a non-posture reference task was sequenced between each posture trial, and the same was also done for shortcut trials. This creates another 36 trials of other reference tasks in each sequence. Overall, each block presented trials for every combination of reference task interspersed with all 36 postures. In summary, there were 5 BLOCKS × 2 sequences × 2 FORMS × 3 ORIENTATIONS × 2 HANDS × 3 SURFACES × 2 Reference Tasks = 720 trials per participant.

6.5 Procedure

The same procedure was used as Experiment 1: an initial typing speed test followed by 5-minutes of practice covering all tasks and conditions before the main experiment trials. Each experiment session took approximately 20 minutes to complete.

6.6 Results

We examined error-free task completion and transition times for posture and shortcut task trials to identify outliers more than 3 standard deviations from each condition mean, resulting in the removal of 1.4% of trials. Visual inspection of task completion time and transition time distributions suggested non-normality, which was confirmed by Shapiro-Wilk and Anderson-Darling tests. To compensate, we log-transformed all data points for stronger statistical analysis that rely on data normality. This is a known practice to improve the significance of results because parametric tests provide stronger evidence to reject a null hypothesis compared to their non-parametric equivalent that do not rely on normal data. A repeated measures ANOVA was used for effects, with post hoc Bonferroni-corrected t-tests used for pairwise differences.

6.6.1 Learning Effect. We compared the task completion time and error-rates across all experimental blocks. It was observed that BLOCK had a main effect on time ($F_{1,19}$ = 53.10 , p < 0.001, n_p^2 = 0.736). However, no such effect was observed on error. Post hoc test revealed that block 1 was significantly slower for postures than the remaining 4 blocks (all p < 0.05), suggesting an initial learning effect, despite our protocol including time for practice. In all subsequent analysis, we only use the last 4 blocks for the best estimation of how postures compare to keyboard shortcuts after learning.

6.6.2 Errors. We define two types of errors. There can be a *trial-error* when the trial was not completed, such as not typing the correct sequence, or not forming the correct posture within the 6s maximum allowable trial time. The typing task accounted for the most number of trial-errors (N = 74), followed by shortcuts (N = 13), and postures (N = 6). There were no trial-errors with the clicking task. Overall, the trial-errors were encountered in less than 1% of the total trials.

The second type of error is a *mode-error*, where the participant formed the wrong posture or performed the wrong shortcut key combination. Identifying shortcut mode-errors is simple, but posture mode-errors are more complex since recognition occurs for each frame, resulting in (often short) sequences of different posture recognition events before the participant finishes moving their hand to form the target posture. To compensate, we calculate the ratio between correct posture detected from last trial and incorrectly identified postures over 180 frames from the current trial. If this ratio is more than 50% in the total frames identified since the beginning of the trial, we considered it a mode-error. Mode-errors were encountered in 4% of the postures (N = 145) and 3% of shortcuts (N = 111). In total, 2% of the total trials contained either type of error. We do not include either type of error trials in time analysis.

6.6.3 Task Completion Time. This is the duration starting when the first task prompt appears until the task is successfully completed. Overall, the mean posture completion time (1139 ms) is faster than shortcuts (1738 ms) (Figure 8e) with a significant main effect for the posture versus shortcut CONDITION on *Task Completion Time* ($F_{1,19}$ = 142.3 , p < 0.001, n_p^2 = 0.882). Furthermore, we found no evidence that posture completion times were affected by the preceding reference task, i.e. whether it was typing or clicking, as no significant effect of the preceding reference task type (typing or clicking) was found on task completion time for postures or shortcuts (p=.88).

When considering postures analyzed by parameter variations, we found that postures with a 90° orientation are 123ms faster than 180°, but surprisingly, also 133ms faster than 0°. We also found small differences in other parameters: Non-dominant postures were 34ms faster than dominant ones, open forms 48ms faster than closed forms, and those performed below the keyboard surface 75ms faster than beside the keyboard and 67ms faster than those on the keyboard. These differences are relatively small, as they represent approximately 6% or less of typical mean times.

These observations are supported by statistical analysis. Pairwise comparison of postures with respect to wrist ORIENTATION showed that postures with 90° wrist rotation (M=1056.5, SD = 256.7) were significantly faster than 0° (M=1189.5, SD = 233.3) and 180° wrist rotations (M=1179, SD = 214.5) (all p < 0.001). No significant difference was noticed between 0° and 180° wrist rotations (p > 0.05). We analyzed the effect of HAND on task completion time. Postures performed with the non-dominant hand were found to be significantly faster (M=1122.07, SD = 264.55) than postures performed with the dominant hand (M=1156.6, SD = 219.9) (p < 0.05). Finally, hand FORM was found to have a main effect on the task completion time. Open-hand postures (M=1115.7, SD = 265.3) forming a flat palm were found to be significantly faster than close-hand postures (M=1163.4, SD = 217.3) forming a fist (p < 0.01). This implies that it is perhaps easier to open the hand than forming a closed hand posture when coming from a typing task. Investigating the effect of SURFACE on task completion time revealed that postures formed below the typing area were performed significantly faster (M=1091.4, SD = 294.0) compared to posture performed beside (M=1166.1, SD = 197.9) and on (M=1158.96, SD = 224.7) the typing area (both p < 0.05). No significant difference was found between beside and on (p > 0.05).

6.6.4 Transition Times. Understanding how long it takes to transition into, or out of, a posture or shortcut may reveal differences in kinematics, providing more understanding for the over difference in task completion time. We define *Transition-In Time* as the time interval from the task prompt appearing (which is also the end of the previous reference task trial) until the first input action. For shortcut tasks, the first input action is the first keypress event. For postures, it is the first time the correct posture was identified by the recognizer. We define *Transition-Out Time* as the time interval from the last input action until the task prompt appears in the following reference task trial. The last input action is the last key-release event for a shortcut or the last time the correct posture was identified by the recognizer.

Overall, postures have 600 ms faster Transition-In time compared to shortcuts when coming from a preceding task (Figure 9a and 9b), but 490 ms slower Transition-Out time when moving to the next task (Figure 9c and 9d). When also

considering which reference task is being transitioned from or to, we see a more nuanced pattern. The Transition-Out Time from postures to a typing task is faster than Transition-Out Time to a clicking task (1148 ms vs 1384 ms), but Transition-In Times are comparable (1165 ms vs 1178 ms). Similarly for shortcuts, the Transition-Out Time to a typing task is significantly faster than Transition-Out Time to a clicking task (105 ms vs 1464 ms), and Transition-In Times are comparable (1803 ms vs 1739 ms). This significance indicates that switching back to typing from a shortcut task seems almost as fast as coming from a typing task itself.

This summary is based on statistical analysis. With two-way ANOVA, we found significant main effects of the technique ($F(1,19)=274.36$, $p<0.01$, $\eta^2=0.84$) and preceding reference task ($F(1,19)=585.22$, $p<0.01$, $\eta^2=0.94$) on Transition-In Time. Likewise, significant main effects of the technique ($F(1,19)=412.38$, $p<0.01$, $\eta^2=0.89$) and next reference task ($F(1,19)=862.72$, $p<0.01$,$\eta^2=0.95$) was found on Transition-Out Time. We also found a significant interaction of technique with preceding reference task ($F(1,19)=323.12$, $p<0.01$, $\eta^2=0.65$) and next reference task ($F(1,19)=1011.21$, $p<0.01$, $\eta^2=0.91$). In post-hoc analysis, postures were found to be significantly faster when transitioning from a clicking task compared to a typing task ($p < 0.05$), but no such effect was found for typing or clicking tasks when transitioning out from a posture ($p = 0.85$).

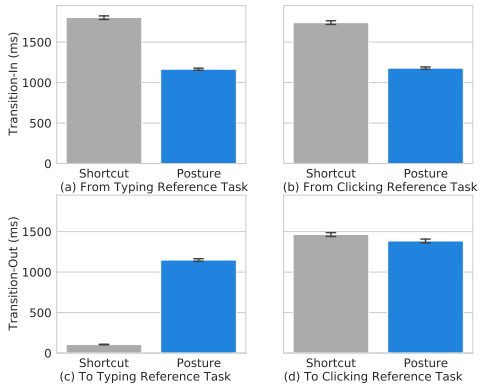

Fig. 9. Transition-In and Transition-Out Time for postures and shortcuts when transitions to or from typing and clicking reference tasks.

6.7 Summary

Results from the second experiment validates the recognizer, show that postures perform well in the context of different reference techniques, and compare quite favourably to our shortcut key baseline.

6.7.1 Comparison with shortcuts. We measured the task transition time to compare shortcuts and postures. It is encouraging that postures can be significantly faster than shortcuts when transitioning from typing tasks. This could possibly be due to the cognitive overhead from ambiguous finger-to-key mappings and involvement of multiple modifier keys to execute shortcuts. On the other hand, shortcuts are significantly faster than postures while returning to typing, which is no surprise considering shortcuts are very close to typing. Higher transition-out time for shortcuts when switching from clicking operations could be due to the additional homing time between keyboard and trackpad actions.

Note that this comparison only validates the compatibility of postures with existing keyboard operations and does not aim to present postures as an alternative to shortcuts. Shortcuts are inherently close to typing, have a larger interaction vocabulary, but are known to be underused by most computer users [17]. So a few Typealike postures might be all we need to safely augment this interaction space. It would be interesting to compare the memorability of Typealike postures compared to keyboard shortcuts as there is some evidence that gestures can be easier to remember than keyboard shortcuts [1].

6.7.2 Comparison with other techniques. We did not empirically test our postures with previous methods, but we can make meta-comparisons based on related quantitative results. Buschek et al. reported slower execution times for Arm and Wrist rotation gestures than our postures (3.66s vs our 1.2s) [7]. GestAKey appears faster (0.5s vs our 1.2s), but requires major keyboard augmentation with capacitive sensors incorportated into key caps, and the study specifically recruited touch typists [31]. FingerArc and FingerChord [40] techniques are slower in general (4.0s vs our 1.2s) but are more similar for simpler tasks (2.0s vs our 1.2s). However, both require users to press a key followed by a hand posture, something we believe could make the action more disruptive within normal typing tasks.

7 DISCUSSION

As reflected by the results of both studies, using hand postures can be an effective input technique to augment existing keyboard and trackpad operations. Since all Typealike postures are symmetric across both hands, command mappings can be configured separately for left and right-handed users. Based on the times in Experiment 2 and the user preferences in Experiment 1, we propose the following design guidelines for selecting the hand postures.

Postures without rotating the wrist should be avoided during text-entry tasks: Hand postures without wrist rotations are closest to typing postures and can potentially interfere with normal typing operations. This is evident from the results of Experiment 2 where task completion times for postures without wrist rotation are almost similar to those of postures with full wrist rotation. Postures without wrist rotation (0° orientation) not only hinder some user actions, but also confuse the posture recognizer, which could make them a poor choice for unambiguous command invocations.

Postures with maximum wrist rotation can be used for more explicitness in intent: Results from Experiment 1 indicate the postures with 180° maximum wrist rotation were not liked by most users in terms of comfort and overall preference. Further, Experiment 2 suggests these postures are slower to perform than 90° postures. This is possibly due to their extreme divergence from normal hand formations while typing. The two results together suggests that postures with extreme arm and wrist rotations should be used for less frequent and one-shot interactions that require more explicitness in user intent like locking the screen or closing all applications.

Local surfaces can be leveraged for in-situ command invocation: The pattern of transition-in and transition-out times for the postures and shortcuts between typing and clicking reference tasks indicates that the overall task transition time for a technique may depend on the reference task that precedes and follows the interaction. Shortcut invocation was slower when preceded by clicking tasks as both operations take place in different areas of the laptop (on and below the keyboard). However, the same shortcuts are very quick when returning to the typing reference task with significantly less mode-out time. Using local areas disentangles command invocation from keypress events, and facilitate new interactions like in-situ command invocation. For example, users can select text using the laptop's trackpad, quickly form a posture to invoke a copy command, select a destination (again using a trackpad), followed by a hand posture to paste the text. Note how the interaction is independent of the keypress and instead focuses on the in-situ command invocation. Such interactions are not possible with keypress-based command invocations.

Non-dominant hand postures for quick command invocation: In Experiment 2, postures performed with the non-dominant hand were found to be slightly faster than the the dominant hand postures. This suggests the equal, or even preferred use of non-dominant hand postures for command invocation during active engagement. For example, non-dominant hand postures can be considered as a general mode-switching technique, as also suggested in other related studies for pen or touch input [19, 33].

8 DEMONSTRATIONS AND EXTENSIONS

We demonstrate the use of Typealike postures in several applications for complementary commands, mode-switching, and game control. These operations can be performed in various use-cases for system-wide and application-specific interactions. The accompanying video also illustrates these demonstrations.

8.1 Complementary Command Postures

Command postures can allow users to perform certain actions without requiring them to press a key or click a button. This can be useful in situations when the primary user intent is not focused on standard text entry operation. For instance, directional postures can be used to control system audio (Figure 1a and 1e), or flip pages in reading applications (Figure 1d). Further, postures can also enable users to overload keys in multiple ways without needing to press additional modifier keys. This is helpful in situations when retaining the symbolic relevance of a key with the operation is desirable but the modifier keys are already used up by standard shortcut combinations. For instance, pressing 'S' with one hand while making different hand postures with the other hand can take a screenshot of the "entire screen", "selected window" or "selected portion" based on the posture type.

8.2 Mode Switching Postures

During continuous engagement, Typealike postures can be used to switch between different modes without needing to press any key or perform "cursor round trips". For example, in image drawing application like Photoshop, users can switch between drawing and erasing modes by simply forming a '90° close hand posture with the non-dominant hand' to activate the eraser without explicitly selecting it from the toolbox. Similarly, they can also "activate the scaling mode" or "open a snipping tool" by simply forming the relevant posture (Figure 1b). Such use cases wherein shape of the postures can be physically associated with certain actions may be useful for memorability that is worth exploring in future work.

8.3 Gaming applications

An promising use case for hand-posture based interaction is their application in games where the engagement may or may not require the full use of keyboard. Allowing users to control different game elements through hand-postures can make the overall experience more realistic and entertaining. For example, using postures to control basic navigation (Figure 1c), cast spells, or selectively switch between interfaces like dashboards, inventories, or maps can be some really interesting avenues to explore further in future work.

Beyond the above mentioned demonstrations, we also experimented with bi-manual hand postures by combining multiple one-hand postures to further extend this interaction space. While such extensions are only possible after applying additional techniques like transfer learning and rule-based heuristics on top of the existing recognizer, they uncover some exciting interactions such as locking the screen with closed 180° posture with two hands as shown in the video demo.

9 LIMITATIONS

While the results from our experiments are promising, it is important to reflect on the scope of these findings and discuss their applicability beyond the conditions described in the previous sections. First, our experimental task was designed to balance different interaction types such as typing, clicking, shortcuts and postures. While this provided equal number of datapoints for comparison, a realistic task may contain a different proportion of these interaction types. Future studies can draw a fair comparison by using more practical scenarios such as editing an image, or directly embedding other interaction types within a typing task. Further, the chosen baseline of randomly presenting a shortcut from a limited set may have introduced some mental overhead as most shortcut-based interactions are benefited through repeated use, motor memory, and contextual relevance. Shortcut invocation is also known to come from an intentional act rather than a prompted stimulus. While these factors may equally apply to hand-posture based interactions, we did not explicitly measure this through our experiments. Future studies can validate these results in more realistic and less controlled tasks.

Although we used representative techniques to introduce variations in the dataset by alternating lighting conditions and using different keyboard backgrounds, the overall experiment took place in a relatively static indoor setting. Participants were asked to avoid tilting the screen during the first experiment and all sessions took place on the same device. These constrained conditions and dependency on surroundings may limit the generalizability of results to other interaction scenarios which can be validated in future studies.

Another limitation of our system is the mechanism to tackle inadvertent recognition of hand-postures. While our trained recognizer faced limited mode-errors due to a controlled setup, a robust system can utilize additional context around the situation to calculate stronger confidence signals for inadvertent posture detection. Further, misrecognition due to incorrect surface mapping can be avoided by letting the recognizer itself predict the local surface for in-situ command invocation. Balancing the trade-off between processing time and recognition accuracy is an interesting challenge as the time to identify a hand-posture is influenced by the accuracy of the system. Our inference time of 20ms with 97% accuracy is good, but this can be further optimized. Future work can focus on improving the system performance by using larger datasets, more complex models, and personalized hand-postures tailored to individual users.

10 CONCLUSION

We proposed a set of "Typealike" hand postures for command invocation by detecting which hand is used to form the posture, whether the hand is open or closed, at what angle wrists are rotated during posture formation, and where the posture have been formed on the laptop surface. These hand postures are intended to be similar to natural typing poses where hands are always supported by the surface underneath, but different enough to be recognized reliably. This enables a larger input space for traditional keyboard interactions with increased expressivity through hand postures, and increased availability by differentiating between inputs from the normal keyboard and command postures.

Our experimental results show that our postures are a viable input method to augment existing interactions over the keyboard. The dataset of hand postures we generate contribute towards other egocentric hand posture datasets which is useful for both Human Computer Interaction as well as Machine Learning communities. The recognition algorithm performs very well in training, and was robust enough for conducting our controlled lab experiment and to demonstrate the technique. A more comprehensive analysis on mode switching techniques for hand postures and

further exploration of an extended set of this style of postures across different application domains would be natural extensions of our work.

REFERENCES

[1] Caroline Appert and Shumin Zhai. 2009. Using Strokes as Command Shortcuts: Cognitive Benefits and Toolkit Support. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, USA) *(CHI '09)*. Association for Computing Machinery, New York, NY, USA, 2289–2298. https://doi.org/10.1145/1518701.1519052

[2] Oscar Kin-Chung Au, Kira Yip-Wo Yeung, and Jacky Kit Cheung. 2016. DownChord and UpChord: A New Style of Keyboard Shortcuts Based on Simultaneous Key-down and Key-up Events. In *Proceedings of the Fourth International Symposium on Chinese CHI*. 1–10.

[3] Gilles Bailly, Thomas Pietrzak, Jonathan Deber, and Daniel J Wigdor. 2013. Métamorphe: augmenting hotkey usage with actuated keys. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 563–572.

[4] Sven Bambach, Stefan Lee, David J Crandall, and Chen Yu. 2015. Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions. In *Proceedings of the IEEE International Conference on Computer Vision*. 1949–1957.

[5] Steven Bird and Edward Loper. 2004. NLTK: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*. Association for Computational Linguistics, 31.

[6] Florian Block, Hans Gellersen, and Nicolas Villar. 2010. Touch-display keyboards: transforming keyboards into interactive surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1145–1154.

[7] Daniel Buschek, Bianka Roppelt, and Florian Alt. 2018. Extending Keyboard Shortcuts with Arm and Wrist Rotation Gestures. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 21.

[8] Sangwon Choi, Jiseong Gu, Jaehyun Han, and Geehyuk Lee. 2012. Area gestures for a laptop computer enabled by a hover-tracking touchpad. In *Proceedings of the 10th asia pacific conference on Computer human interaction*. ACM, 119–124.

[9] Wenzhe Cui, Jingjie Zheng, Blaine Lewis, Daniel Vogel, and Xiaojun Bi. 2019. HotStrokes: Word-Gesture Shortcuts on a Trackpad. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 165.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[11] Paul H Dietz, Benjamin Eidelson, Jonathan Westhues, and Steven Bathiche. 2009. A practical pressure sensitive computer keyboard. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*. 55–58.

[12] Wolfgang Fallot-Burghardt, Morten Fjeld, C Speirs, S Ziegenspeck, Helmut Krueger, and Thomas Läubli. 2006. Touch&Type: a novel pointing device for notebook computers. In *Proceedings of the 4th Nordic conference on Human-computer interaction: changing roles*. ACM, 465–468.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[14] Jeremy Howard and Sylvain Gugger. 2020. Fastai: A layered API for deep learning. *Information* 11, 2 (2020), 108.

[15] Shaun K Kane, Daniel Avrahami, Jacob O Wobbrock, Beverly Harrison, Adam D Rea, Matthai Philipose, and Anthony LaMarca. 2009. Bonfire: a nomadic system for hybrid laptop-tabletop interaction. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*. ACM, 129–138.

[16] Sunjun Kim and Geehyuk Lee. 2016. TapBoard 2: Simple and Effective Touchpad-like Interaction on a Multi-Touch Surface Keyboard. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 5163–5168.

[17] David M Lane, H Albert Napier, S Camille Peres, and Anikó Sándor. 2005. Hidden costs of graphical user interfaces: Failure to make the transition from menus and icon toolbars to keyboard shortcuts. *International Journal of Human-Computer Interaction* 18, 2 (2005), 133–144.

[18] Jinha Lee, Alex Olwal, Hiroshi Ishii, and Cati Boulanger. 2013. SpaceTop: integrating 2D and spatial 3D interactions in a see-through desktop environment. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 189–192.

[19] Yang Li, Ken Hinckley, Zhiwei Guan, and James A Landay. 2005. Experimental analysis of mode switching techniques in pen-based user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 461–470.

[20] Hugh McLoone, Ken Hinckley, and Edward Cutrell. 2003. Bimanual interaction on the microsoft office keyboard. In *INTERACT'03*. 49–56.

[21] Pranav Mistry and Patricia Maes. 2010. Mouseless. In *Adjunct proceedings of the 23nd annual ACM symposium on User interface software and technology*. ACM, 441–442.

[22] Thomas A Mysliwiec. 1994. Fingermouse: A freehand computer pointing interface. In *in Proc. of Int'l Conf. on Automatic Face and Gesture Recognition*. Citeseer.

[23] Michael Ortega and Laurence Nigay. 2009. AirMouse: Finger gesture for 2D and 3D interaction. In *IFIP Conference on Human-Computer Interaction*. Springer, 214–227.

[24] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research* 12, Oct (2011), 2825–2830.

[25] Thomas Pietrzak, Sylvain Malacria, and Gilles Bailly. 2014. CtrlMouse et TouchCtrl: duplicating mode delimiters on the mouse. In *Proceedings of the 26th Conference on l'Interaction Homme-Machine*. 38–47.

[26] P Pramod Kumar, Prahlad Vadakkepat, and Loh Ai Poh. 2017. The NUS hand posture datasets I. (2017).

[27] P Pramod Kumar, Prahlad Vadakkepat, and Loh Ai Poh. 2017. The NUS hand posture datasets II. (2017).

[28] Julian Ramos, Zhen Li, Johana Rosas, Nikola Banovic, Jennifer Mankoff, and Anind Dey. 2016. Keyboard Surface Interaction: Making the keyboard into a pointing device. arXiv:1601.04029 [cs.HC]

[29] Jun Rekimoto, Takaaki Ishizawa, Carsten Schwesig, and Haruo Oba. 2003. PreSense: interaction techniques for finger sensing input devices. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*. 203–212.

[30] Baer Sawin and S Baer. [n.d.]. *DesRoches (1998). Mobile Computer User Frustrations 1998 Research Summary.* Technical Report. IBM Internal Technical Report.

[31] Yilei Shi, Haimo Zhang, Hasitha Rajapakse, Nuwan Tharaka Perera, Tomás Vega Gálvez, and Suranga Nanayakkara. 2018. GestAKey: Touch Interaction on Individual Keycaps. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 596.

[32] Leslie N Smith. 2018. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820* (2018).

[33] Hemant Bhaskar Surale, Fabrice Matulic, and Daniel Vogel. 2019. Experimental Analysis of Barehand Mid-air Mode-Switching Techniques in Virtual Reality. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 196.

[34] Stuart Taylor, Cem Keskin, Otmar Hilliges, Shahram Izadi, and John Helmes. 2014. Type-hover-swipe in 96 bytes: a motion sensing mechanical keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1695–1704.

[35] Ying-Chao Tung, Ta Yang Cheng, Neng-Hao Yu, Chiuan Wang, and Mike Y Chen. 2015. FlickBoard: enabling trackpad interaction with automatic mode switching on a capacitive-sensing keyboard. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 1847–1850.

[36] Andrew D Wilson. 2006. Robust computer vision-based detection of pinching for one and two-handed gesture input. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*. ACM, 255–258.

[37] Jacob O Wobbrock, Andrew D Wilson, and Yang Li. 2007. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*. 159–168.

[38] Robert Xiao, Greg Lew, James Marsanico, Divya Hariharan, Scott Hudson, and Chris Harrison. 2014. Toffee: enabling ad hoc, around-device interaction with acoustic time-of-arrival correlation. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services*. ACM, 67–76.

[39] Haimo Zhang and Yang Li. 2014. GestKeyboard: enabling gesture-based interaction on ordinary physical keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1675–1684.

[40] Jingjie Zheng, Blaine Lewis, Jeff Avery, and Daniel Vogel. 2018. FingerArc and FingerChord: Supporting Novice to Expert Transitions with Guided Finger-Aware Shortcuts. (2018).

[41] Jingjie Zheng and Daniel Vogel. 2016. Finger-Aware Shortcuts. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 4274–4285.