Complex Interaction as Emergent Behaviour: Simulating Mid-Air Virtual Keyboard Typing using Reinforcement Learning

Lorenz Hetzel*
University of Cambridge
ETH Zürich

John Dudley†
University of Cambridge

Anna Maria Feit‡
Saarland University

Per Ola Kristensson§
University of Cambridge

Abstract— Accurately modelling user behaviour has the potential to significantly improve the quality of human-computer interaction. Traditionally, these models are carefully hand-crafted to approximate specific aspects of well-documented user behaviour. This limits their availability in virtual and augmented reality where user behaviour is often not yet well understood. Recent efforts have demonstrated that reinforcement learning can approximate human behaviour during simple goal-oriented reaching tasks. We build on these efforts and demonstrate that reinforcement learning can also approximate user behaviour in a complex mid-air interaction task: typing on a virtual keyboard. We present the first reinforcement learning-based user model for mid-air and surface-aligned typing on a virtual keyboard. Our model is shown to replicate high-level human typing behaviour. We demonstrate that this approach may be used to augment or replace human testing during the validation and development of virtual keyboards.

Index Terms—Reinforcement learning, virtual reality, user model.

1 Introduction

A central interaction challenge in virtual and augmented reality is allowing users to comfortably and efficiently input text and commands. However, a major challenge is that successful text entry methods tend to 1) provide high immediate efficacy—in other words, they are effective with minimal learning effort; and 2) exhibit high efficiency in relation to other text entry methods users are already familiar with, such as typing on their laptops, tablets or mobile phones [28–30]. This poses a considerable design challenge as it is difficult to create an unfamiliar text entry method that is both easy-to-learn and efficient. As a result, a promising direction of research is to identify means to effectively transplant an established text entry method to virtual and augmented reality headsets that is both efficient and familiar to users. One such candidate is mid-air virtual keyboard typing, which allows users to type using both hands on a virtual keyboard reminiscent of a laptop or tablet keyboard.

However, an effective high-performance mid-air virtual keyboard relies on using a machine learned model to infer users' intended text from noisy sensor data. It is therefore necessary to involve users to both 1) iteratively evaluate system designs; and 2) collect representative development, training and testing data of users' typing behaviour. While this is a well-known challenge in many human-computer interaction (HCI) systems, the cost of instrumenting users for interactive virtual and augmented reality systems exacerbates the problem.

One established solution is user modelling, which is practically as old as the field of human-computer interaction itself. However, these models tend to be meticulously handcrafted to approximate human behavior for specific interaction tasks [53]. The highly specific nature of these models has resulted in most of them not being widely used by practitioners developing new HCI systems. Additionally, most existing user models are designed to approximate known human behaviour. While this is a valid approach for interactive scenarios that are well understood, it is not useful for more complex tasks or cases where limited user data is available, such as typing on a mid-air virtual keyboard.

*e-mail: hetzell@ethz.ch
†e-mail: jjd50@cam.ac.uk
‡e-mail: feit@cs.uni-saarland.de
§e-mail: pok21@cam.ac.uk

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

In contrast, recent work in reinforcement learning (RL) has demonstrated that complex human-like behaviour may emerge from simple cost functions [17]. In particular, RL agents have been shown to approximate human behaviour during simple targeted reaching movement tasks [6, 14].

In this paper we build on and extend this prior work by showing that these RL agents are able to learn more complex interactive tasks, in particular two-finger typing on a mid-air virtual keyboard. Using reinforcement learning has the notable advantage that absolutely no user data is required to develop or train the model. In contrast to heuristic-based user models, our RL model does not explicitly consider the keyboard geometry, the type of text being typed, or the keyboard's size and position. For example, typing speed and accuracy are never explicitly enforced but emerge naturally from a simple cost function, a muscle model and the kinematics of the skeletal model. In other words, it is a truly generative model of two-finger virtual keyboard typing behaviour. This model can be retrained and applied to examine different keyboard designs and interaction mechanisms without any changes to the underlying architecture of the model. Therefore, models such as ours can, in many cases, be extended beyond their original scope with computational effort alone, for example, without requiring any manual work by engineers.

Our RL model is useful in a variety of applications. For example, we show that the learned policy approximates human behaviour and can therefore be used to reliably and objectively select hyperparameters during the development of virtual keyboards (and possibly other HCI systems). Since our model makes no assumptions about the underlying keyboard layout it may additionally be used to comprehensively qualify parameters, such as key size, which are prohibitively expensive to evaluate with real users.

Our RL model offers an attractive trade-off between modelling effort and usefulness. While, for example, Fitts' law is simple to apply to a novel system it only predicts average movement times as a function of an index of difficulty. On the other end of the spectrum, complex handcrafted models, such as the queuing network model of transcription typing [53], can accurately predict human behaviour but are also much less versatile. Our RL model offers much of the fidelity of the handcrafted user models, while retaining much of the ease of use and generality that practitioners appreciate about heuristics, such as Fitts' law.

The central objective of this work is to demonstrate that we can learn a model of two-finger virtual keyboard typing using model-free reinforcement learning and a naïve cost function. As we will demonstrate later in this paper, we use reinforcement learning to learn a model of two-finger virtual keyboard typing that approximates two-finger mid-air

typing data from real users without our model being trained on *any* empirical user data at all or tuned to match any evaluation metric, such as entry or error rate.

In summary, this paper makes the following contributions:

- We demonstrate that reinforcement learning for control of a biomechanical model can be used to create a generative model that approximates users' virtual keyboard typing behaviour.
- We show how such a model can be used to objectively identify good hyperparameters with high confidence and augment, or replace, testing with human users.

2 Related Work

Model-free reinforcement learning agents have recently achieved groundbreaking results in many domains including multi-agent coordination [2], recommendation systems [55] and dexterous manipulation [39]. In particular, they have increasingly been used to solve challenging biomechanical control tasks [37]. Recent works have begun to leverage these advances to model specific aspects of simple mid-air interactions. For example, Fischer et al. [15] show that an RL agent trained to hit targets in mid-air using a biomechanical model of the human arm observes the two-thirds power law and Fitts' law. In a more practical sense, RL agents using a simplified arm model have been shown to predict perceived fatigue (the "gorilla arm effect") in mid-air interaction [6]. We extend these approaches by modelling mid-air text entry—a substantially more complex task.

Beyond AR and VR, predicting user intent has been widely explored in the HCI literature for pointing tasks [15, 31] and beyond [17]. For example, Lank et al. [31] predict the most likely 2D selection intent from noisy mouse movements. More recent work [1, 54] proposes stronger predictive methods that rely on continuous hand tracking to predict intended touch points from 3D to accommodate the more complex motions that can occur in 3D space.

As transcription typing is pervasive in typing studies [10, 11, 13, 51], we focus our efforts on modelling this task. In a transcription task the user is instructed to type predefined stimulus phrases. The cognitive and motor aspects of transcription typing have been widely investigated [41]. Based on these findings, queuing network based methods [53] have been used to explicitly model transcription typing. In contrast to our approach, these models are carefully designed by hand to reflect specific empirically observed phenomena in transcription typing and do not explicitly consider the biomechanical characteristics of the human arm.

While some recent prior work has begun to explore the kinematics underlying mid-air reaching movements [32], their fidelity is still limited compared to the understanding built through decades of research on conventional keyboard and mouse-based entry methods. Unlike user models of traditional input methods, modelling mid-air interaction is therefore particularly challenging since the underlying phenomena are not yet well understood.

Text entry is perhaps one of the most challenging input tasks in virtual and augmented reality and many different text entry methods have been investigated [3]. While researchers have studied appropriating physical keyboards for VR [19, 27, 43], a likely future dominant solution is the transplanting of full-sized keyboards into VR and AR environments as full-sized virtual keyboards available for either mid-air typing or surface-aligned typing supported by a reappropriated nearby hard surface. However, to support such typing is highly challenging, as the targets (keys) are tightly packed, requiring highly precise hand and finger tracking [40], as well as strong predictive models to account for incorrect key hits. Additionally, direct comparisons with regular keyboards and the associated user expectations regarding speed and accuracy are difficult to manage. Similar to text entry on small touchscreens (e.g. [18, 52]), most existing virtual keyboard work leverages statistical modelling to improve typing accuracy [12, 40]. Previous work has shown that aligning virtual keyboards with a physical surface can significantly improve the entry rate [11]. Dudley et al. [11] found that corrected entry rates were significantly higher for two-finger typing than ten-finger typing on a mid-air virtual keyboard. This motivates our choice to use reinforcement learning to learn a model of two-finger typing for mid-air and surface-aligned keyboards.

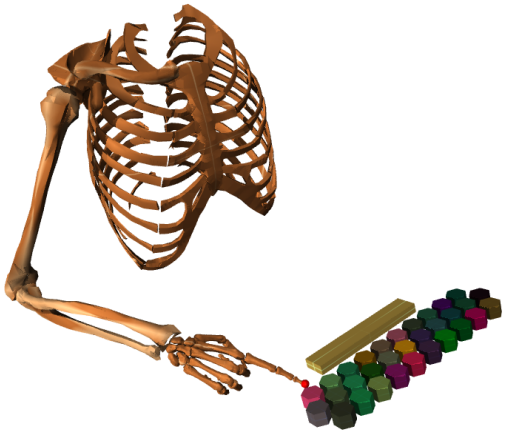

Fig. 1. Arm model and keyboard geometry. At inference time the left arm is simulated as described in Sect. 3.6.

3 Approach

Mid-air typing is a complex process since the user needs to plan and execute a trajectory between many tightly packed targets—the keys on the keyboard. Simulating such a task requires careful consideration. In this section we therefore initially consider the problem from a purely biomechanical perspective and establish the kinematics and actuation of the system. After introducing the simulated mechanical system, we outline how the process of typing can be formulated as a Markov decision process and subsequently learned using reinforcement learning.

3.1 Kinematics

It is important to consider the considerable kinematic complexity of the human arm and hand. The human arm is often assumed to have seven principal degrees of freedom (DOF): three in the shoulder, one in the elbow, one for forearm rotation, and one each for wrist deviation and flexion [42]. The human hand is considerably more complex with 27 kinematic degrees of freedom and a number coactivations between them [35]. While there has been some success in RL-based control of highly complex hand models [37, 38], we chose to fix the hand in a rigid pointing pose to reduce the complexity of the RL problem and leave the investigation of fully actuated ten-finger typing for future work. Additionally, there is empirical evidence suggesting that two-finger virtual keyboard typing may be preferred by many users for mid-air interaction [11].

Inspired by Fisher et al [14], we adapt a biomechanical model of the human arm [42], which was originally created for the *Stanford Open Sim* and converted to the *Mujoco* physics simulator [48] by Ikkala and Hämäläinen [24]. The thorax is fixed to the world coordinate system. We manually adjusted the bones of the hand to form a rigid pointing posture. A sphere with a diameter of 10 mm is attached to the finger tip of the index finger to approximately model the tip of the human index finger. It is important to note that all bones of the hand and arm are treated as rigid bodies, any of which may make contact with the virtual keyboard.

3.2 Actuators

The original version of the arm model we employ [42] included 50 Hill-type muscle-tendon, that is, linear actuators. These actuators act upon just 7 degrees of freedom, which require careful coordination between the individual actuators. To dramatically reduce the dimensionality of the control problem, we remove the 50 linear actuators in our model and replace them with just 7 direct torque actuators attached to the joints indicated in Table 1. This approach is commonly used when controlling biomechanical models using reinforcement learning [6, 14].

Table 1. Joint limits and peak torques [14].

Joint Name	Limits (°)	Torque Limit (Nm)
Elevation plane	[-90, 130]	36.01
Shoulder elevation	[0, 180]	60.97
Shoulder rotation	[-90, 20]	19.37
Elbow flexion	[0, 130]	12.57
Forearm rotation	[-90, 90]	1.03
Wrist deviation	[-10, 25]	2.14
Wrist flexion	[-70, 70]	1.53

To approximate the torque response of the original muscle-driven joints, we include a muscle model that computes the instantaneous joint torque from the control inputs. We now briefly introduce some of the key characteristics that define the torque response of muscle-driven joints. For highly explosive efforts, such as a sprinter pushing off the starting block at the beginning of a race, the torque response is not instantaneous, since muscle fibres need to be activated. For example, the maximum torque around the knee joint is only reached after about 250 ms [47]. Another significant factor that influences the torque response is how close the current joint position is to the limits of the movement range. For example, the peak torque around the elbow joint is reduced by up to 50% at the joint's angle limits [26]. Intuitively, peak joint torque is also decreased significantly after prolonged exertion [34].

Guided by related work in controlling biomechanical models of the human arm [6, 14], we identify two potentially suitable models: a three compartment muscle model called 3CCr [34], which emphasises predicting fatigue, and a second-order model [14], which approximates the torque-time response of muscle groups. Each of these models can approximate only some of the phenomena discussed in the previous paragraph. For example, neither model considers the considerable torque decrease when approaching the angular limit of the joint.

Additionally, our preliminary experiments revealed that adding either of these models to our arm model significantly increased the complexity of the RL problem. Without a muscle model we observed satisfactory performance in the first stage of training after around one million iterations. In contrast, nearly three million iterations were required to achieve comparable performance with a second-order muscle model. The 3CCr model, which has previously only been used on arm models with simplified kinematics [6], did not achieve satisfactory performance.

We therefore opted to pursue a simplified approach and introduce both signal dependent and constant noise into our actuators, which is a tactic that has been shown to generate realistic reaching movements [49]. Following the typical notation for a Markov decision process, we call the control input to the system the *action*. The perturbed action or control input a_t is computed from the nominal action (control signal) $\hat{a}_t$ as follows:

$$a_t = \min(\max(\mu \cdot \hat{a}_t + \varepsilon, -1), 1). \tag{1}$$

The signal-dependent perturbations, μ, and signal-independent perturbations, ε, are drawn at each time step from a Gaussian distribution with standard deviations chosen to be as large as possible without impacting the stability of the training regime subsequently introduced in Sect. 3.5. We limit our actuators to the natural torque limits reported by Fischer et al. [14]. These limits were computed by performing inverse kinematics on motion capture data of a human performing a mid-air pointing task. We assume here that the action (control signal) $\hat{a}_t$ is bounded on $[-1, 1]$. As indicated in Equation 1, the perturbed action is clipped to the same interval to ensure the torque limits are observed at all times. Using the vector of peak torque magnitudes, g, described in Table 1, the instantaneous joint torques, τ, are computed from the perturbed action a_t as follows:

$$\tau = g \cdot a_t. \tag{2}$$

Therefore, in contrast to related work, we chose not to explicitly model the time and fatigue dependence of peak joint torques. Considering the near instant response of human joints for small changes in torque amplitude [47], and the fact that our model observes the peak torques reported by Fischer et al. [14], we expect our model to approximate natural motions with reasonable accuracy. Additionally, since we initially only aim to model user behaviour during transcription typing of short stimulus phrases, we propose that fatigue-driven peak torque decline can be safely ignored. Further, we expect typing of short phrases to be the most common use-case for mid-air text entry, since sustained writing without a supporting surface is highly fatiguing [6].

3.3 Agent

The underactuated nature of our biomechanical model makes computing optimal control inputs from inverse kinematics highly challenging [9]. This in combination with the fact that we would like to abstain from using questionable [22] heuristics, such as minimising jerk or energy, leaves model-free reinforcement learning as the obvious choice for actuating the biomechanical model. An RL agent is trained to maximise the expected reward given to it. RL, in contrast to imitation learning [23], does not rely on any preexisting data. While this eliminates the concern of overfitting, it can be challenging to control the specific behaviour of an RL agent, since behaviour emerges from a simplistic cost function. For example, an agent may learn to 'cheat' at a game, if the environment and cost function are not sufficiently restrictive [2]. RL using direct torque control of skeletal models has recently been shown to synthesise movements that closely mimic humans [14, 25].

We chose a soft actor critic (SAC) [20] network ensemble, since it has recently been successfully applied in many domains of dynamic control [14, 21]. Additionally, SAC is comparably sample efficient, which is favourable since our training is primarily bottle-necked by the single threaded physics simulation in *Mujoco*.

For details about the algorithm we refer to the original publication [20]. However, we provide a brief overview here for completeness. SAC concurrently trains a policy (the actor) and two Q-networks (the critic). In high level terms, the Q-networks learn to jointly predict the expected reward during training and the actor learns to act in a way that maximises the reward predicted by the Q-networks.

While standard RL formulations maximise the expected reward, SAC is a variant of maximum entropy RL, which adds a bonus reward based on the entropy H of the policy π to the objective J:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}\left[r(s_t, \hat{a}_t) + \alpha H(\pi(\cdot|s_t))\right]. \tag{3}$$

Figuratively speaking, given a state s_t the agent tries to optimise the expected reward $r(s_t, \hat{a}_t)$, while also acting as randomly as possible. The entropy regularisation coefficient α determines the trade-off between these two factors. We use an implementation of SAC that balances exploration and exploitation by varying α throughout training. Further details on this approach are available in the original implementation[1] and associated publication [21]. The actor and both Q-networks that act as the critic have two hidden layers with 256 nodes and ReLu [36] activation. The parameters for all three networks are optimised using the Adam Optimiser. The two outputs of the actor network are treated as a standard deviation σ_t and a mean μ_t, from which the nominal control signal or action $\hat{a}_t$ is computed as follows:

$$\hat{a}_t(s_t, \xi) = \tanh(\mu_t(s_t) + \sigma_t(s_t) \circ \xi), \quad \xi \sim \mathcal{N}(0, I). \tag{4}$$

Since it is the output of a tanh, the action is bounded on the interval [-1, 1], which is a significant advantage over other off-policy algorithms, such as PPO [45] and TRPO [44]. As outlined in Sect. 3.2, we can therefore ensure that biomechanically accurate torque limits are observed at all times. Importantly, and in contrast to other applications of SAC, we do not disable the action noise in Equation 4 at inference time as we want to encourage stochastic behaviour.

[1] `https://github.com/DLR-RM/stable-baselines3`

Table 2. State of the Markov decision problem.

Quantity	Size
Controllable Joint Positions	7
Controllable Joint Velocities	7
Curriculum Tolerance	1
End-Effector Position	3
Center of Target Key	3
Target to End-Effector Offset	3
Total:	24

Fig. 2. The keyboard layout used during training and evaluation. All labels in metres. Hexagonal keys have a nominal diameter of 25 mm.

3.4 Environment

We model typing on a mid-air keyboard as a Markov decision process (MDP). An MDP describes a discrete-time stochastic control process. It is defined by the tuple $\{s_t, a_t, r\}$, where $s \in \mathbb{R}^{24}$ is the observable system state, as described in Table 2. The reward function r, which changes throughout training as outlined in Sect. 3.5, assigns a value to any action a_t given a state s_t. Reinforcement learning approximates a policy that maximises the expected reward for such an MDP [50]. Due to the exploratory nature of our work, we do not consider the cognitive aspects of typing. Therefore, the typing problem is reduced to a sequence of reaching movements. Since the model is trained on English stimulus phrases, it may, however, learn to optimise its trajectory based on the most likely subsequent key. This is made possible by the fact that Soft Actor Critic optimises the reward over its entire replay buffer, which may contain experiences from multiple subsequent episodes.

The agent takes actions $a_t \in [-1, 1]$, which are converted into instantaneous joint torques at the controllable joints as indicated in Equation 2. We note that the agent is only given information about the next key in the stimulus. To facilitate direct comparison with an existing dataset of users typing on a mid-air and surface-aligned virtual keyboard, the keyboard layout was directly taken from the study performed by Dudley et al. [11]. The keyboard was placed 40 cm in front and 20 cm below the manubrium (i.e. upper part of the sternum) [35] to approximate the setup used by Dudley et al. [11]. Additionally, the keyboard was tilted towards the user by 20 degrees. The keyboard is a QWERTY layout with a reduced punctuation set (,'?!.). Each key, except the space bar, is hexagonal with a nominal diameter of 25 mm. Importantly, the keyboard does not contain a backspace key and the model is not given the opportunity to correct errors. The top row of the layout is approximately 250 mm wide, making it approximately 30% wider than a typical physical layout.

Each key on the keyboard is modelled as a rigid body. Key hits are detected as collisions between any part of the arm and a key. To avoid accidental key repetition we enforce that after hitting a key the hand and arm must leave the keyboard plane completely before another key hit can be registered. Mid-air interaction is modelled by making the contact between the key and hand extremely soft, so that essentially no contact force is generated. This implementation allows us to switch between mid-air and surface-aligned typing very easily.

3.5 Multi-Stage Training

Shaping the reward function to achieve the desired behaviour is an essential aspect of creating an RL agent. The most naïve reward function to model keyboard typing may simply give a large positive reward if the correct key is hit, no reward if no key is hit, and a small negative reward if the incorrect key is hit. We call this a "hit-based" reward. In our initial investigations we found that learning keyboard typing from scratch using this reward function is not feasible, even if combined with a tolerance curriculum. We therefore propose a dual stage training strategy. Additionally, an automatic curriculum is used during both stages to gradually increase the difficulty as training progresses.

First Stage: Initially the agent is rewarded for reaching a target (a sphere around the centre of a key), even if it would have hit other keys along the way. Upon reaching the target sphere, or after three seconds in simulation time, the episode is terminated. The agent is trained in this stage until convergence. The tolerance (radius of the target sphere) is initialised to 300 mm and can be incrementally reduced, at most, every 20,000 steps. It is reduced by 10% if the agent reaches the target within the time limit more than 90% of the time, until the radius of the target is equal to the nominal key radius (12.5 mm). During this stage the model is trained on 40–80 character long randomly generated stimuli. The stimuli are generated using a uniform distribution over all keys on the keyboard.

Second Stage: In this stage we use the "hit-based" reward introduced previously in this section. Key hits are detected as collisions between the hand, or arm, and the keys. The first key the agent hits terminates the episode and the agent is rewarded based on whether it hit the correct key or not. A small negative reward is given at each time-step to accelerate convergence. Episode length is limited to three seconds in simulation time. If no key is hit before the end of the episode this is considered an incorrect key hit. A key hit is considered correct if its centre lies within the current tolerance of the centre of the nominal target key. The tolerance is initialised to 100 mm and can be reduced every 20,000 steps until it converges to zero. The tolerance is reduced if the agent hits the correct key more than 90% of the time during evaluation at that training step. During this stage, the model is trained on 20–100 character long stimuli, extracted from Sherlock Holmes novels. When the episode ends, the agent is only reset if it reaches the end of a stimulus phrase. Otherwise, the agent begins the next episode in the configuration where it finished the previous episode. Figuratively speaking, in the first stage of training the agent learns how to move and in the second stage it learns how to move correctly.

3.6 Bimanual Typing

All considerations up until this point have considered only the right arm and hand. We acknowledge that prior work has found differences in how humans use their left and right hands while typing. For example, the average user globally moves their right hand significantly more than their left hand [13]. While it would be ideal to train an agent that controls two coordinating arms at the same time, this is infeasible in practice since it significantly increases the complexity of the RL problem. In addition to having twice as many degrees of freedom, the agent would also have to learn to dexterously coordinate both hands above the keyboard.

Here we therefore pursue a simplified approach: an agent controlling a right arm is trained to press keys on the entire keyboard. Our two-handed (bimanual) model is guided by a simple geometric realisation: reflecting the arm along the centre plane of the thorax (effectively turning a right arm into a left arm) is equivalent to flipping the keyboard along the centre plane. Bimanual typing is simulated at the time of inference by running two simulations in parallel, and flipping the keys hit in the left simulation along the centre plane. While one arm moves towards the next key in the stimulus, the other simulation is frozen. This is a conservative approximation, as users generally tend to prepare the next key-press for one hand, while the other hand is about to hit a

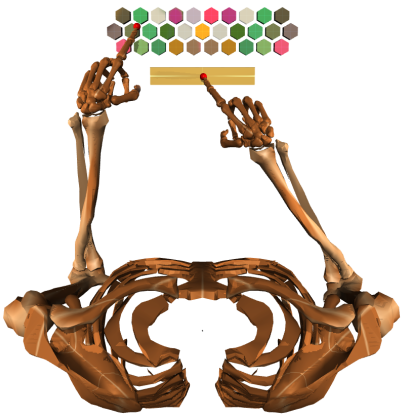

Fig. 3. Illustration of our bimanual composite model. Only one arm may move at a given timestep.

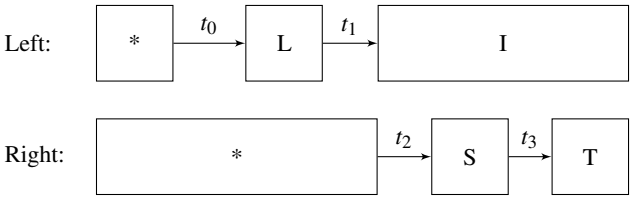

Fig. 4. Illustration of bimanual interleaving for the example stimulus word "LIST". Starting positions for the left and right hand are marked as "$*$". Total time, t, is calculated starting from the first key in the stimulus as a weighted sum of IKI t_i. In this case: $t = t_1 + \beta \cdot t_2 + t_3$.

key [13]. To compensate for the lack of asynchronous movement in our model, and to approximate real user behaviour more closely, we scale the inter-key intervals (IKIs) for hand alternating bigrams by a factor $\beta = 141.28/170 \approx 0.8311$. This is the ratio of hand alternating and single handed IKIs for non touch-typists determined by Feit et al. [13]. The above approach is illustrated in Fig. 3 and Fig. 4.

3.7 Map-Based Keyboard Decoder

To illustrate the usefulness of our model we subsequently investigate if it may be used to select hyperparameters for a map-based keyboard decoder during our evaluation. For completeness, we briefly introduce the decoder under consideration here. The objective of the decoder is to translate an observation sequence of touch inputs into intended text. The decoder operates on individual words. Words may be delimited by space, comma, full-stop, question-mark and exclamation-mark characters. The decoder combines predictions from a touch model and two language models to find the most likely intended word.

The touch model assigns a probability $P(o_i|l_i)$ to a touch point o_i given a key l_i using a 2D Gaussian distribution with diagonal variance centred on the key. The probability $P(O|L)$ of an observation sequence O with length N given a character sequence L is computed as follows:

$$P(O|L)P_c(L) = \prod_{i=1}^{N} P(o_i|l_i)P_c(l_i). \tag{5}$$

The set L also contains the options that any given touch was preceded by an omission or is itself an insertion error. The probabilities of these options are reduced by multiplying with e^{κ}, where κ is the insertion or deletion penalty. This prediction is combined with a character-level language model that predicts the accumulated likelihood of a character sequence $P_c(L)$. Finally, we form a weighted average with the prediction $P_w(L|S)$ from a word-level language model that considers the full sentence S to find the most likely character sequence $\hat{L}$. The two predictions are weighted by a factor λ:

$$\hat{L} = \arg\max_{L} \left[\lambda P(O|L)P_c(L) + (1-\lambda)P_w(L|S)\right]. \tag{6}$$

The character-level language model is a 5-gram model trained on a subset of texts from the Gutenberg Project. The word-level language model is the 'tiny' variant of a publicly available 4-gram model.[2]

In our hyperparameter selection task we consider the following decoder parameters:

- insertion and deletion penalties as integers in [-20, -2];
- variance in x and y direction used by the touch model as real values in [6.25, 25];
- and model weighting λ, a real value in [0.1, 0.9].

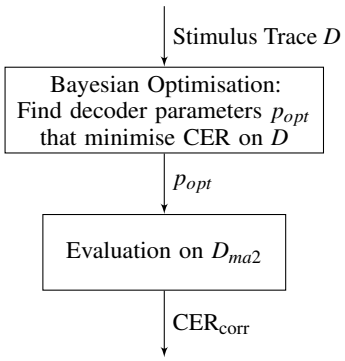

Fig. 5. Optimisation scheme used in Sect. 4.2. The Stimulus trace D contains stimulus phrases and corresponding touch points generated by our model or a real user.

4 Evaluation

We evaluate our model in two ways: First, we show that it can predict the results of a user study (with real users) for both mid-air and surface-aligned typing on a virtual QWERTY keyboard. Second, we illustrate how the model can be used to augment or replace human testing, by using our model to select good hyperparameters for the keyboard-decoder introduced in Sect. 3.7.

In both parts of our evaluation we compare our model against 24 real users from a recent user study by Dudley et al. [11]. The study includes data for two-finger typing on a surface-aligned and mid-air virtual QWERTY keyboard (see Fig. 2). Each participant typed a unique set of 160 phrases per condition, resulting in two datasets, each containing 3,840 stimulus phrases. Following the notation of the original publication we subsequently refer to the datasets as D_{sur2} for surface-aligned typing and D_{ma2} for the mid-air condition. Unless otherwise specified, all evaluations of our model were carried out using stimuli sampled from the same corpus of phrases used in the benchmark user study. The stimulus phrases consist of four words or more and are less than 40 characters long.

4.1 Alignment with User Behaviour

In this part of the evaluation we show that our model is well aligned with many behaviours observed in the benchmark study introduced at the beginning of the previous section. We trained two versions of

[2] https://aactext.org/imagine/

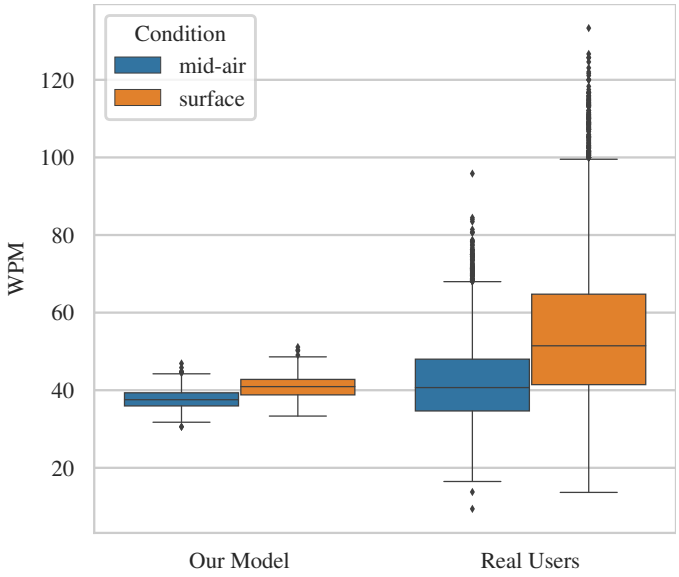

Fig. 6. Box-and-whisker plots of per stimulus entry rate (WPM). Whiskers are 1.5 times longer than the interquartile range (IQR). Our model predicts that users will type faster on surface-aligned keyboards, which is well aligned with empirical data [11].

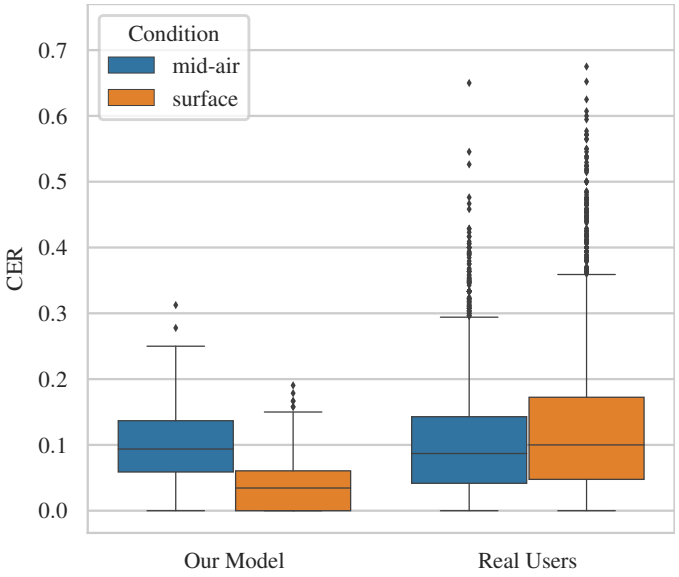

Fig. 7. Box-and-whisker plots of uncorrected, per stimulus, character error rate (CER). Whiskers are 1.5 times longer than the interquartile range (IQR). While the z-scores for both conditions are low, our model does not predict that the uncorrected character error rate will be higher for surface-aligned typing.

our model: one for mid-air and one for surface-aligned typing. Both the model for surface-aligned and mid-air typing were trained for one million iterations during the first training stage and four million iterations during stage 2 of the training regime outlined in Sect. 3.5. We evaluate uncorrected character error rate (CER), words per minute (WPM) and inter-key intervals (IKIs). Words per minute is calculated as the number of characters typed per minute divided by a nominal word length of five characters. Character error rate is computed as the Levenshtein distance between the stimulus and response text, divided by the length of the stimulus phrase (including spaces).

We quantify how well our model fits into the population of real users using z-scores (also known as standard scores). The z-score corresponds to the number of standard deviations $\text{std}(x)$ that a sample x is away from the mean $\mathbf{E}(x)$ of a given distribution:

$$z = \frac{x - \mathbf{E}(x)}{\text{std}(x)}. \tag{7}$$

Therefore, we desire our model to have a z-score close to zero. All results are listed in Table 3 and Table 4. Our mid-air model lies within 0.5 standard deviations from the mean for real users for all metrics. This is a significant achievement considering that our model was not explicitly designed, or trained, to approximate *any* of these metrics. If, for illustrative purposes, we assume that entry rates (WPM) are normally distributed across the general population for mid-air typing, then our model can be considered to be more average than approximately 70% of real users. Further, our model correctly predicts that users can enter text faster on surface-aligned keyboards (see Fig. 6), which is well aligned with real user behaviour [11]. Additionally, we note that our model falls short of the average WPM achieved by real users in both conditions. While there may be other confounding factors, this may be caused by our conservative approximation of bimanual interleaving. However, even though our model approximates the "average user" with surprisingly high accuracy, it does not yet capture the full spectrum of human performance, as can be seen by the much lower variance expressed in Fig. 6. We will cover a number of ways to address this in our subsequent discussion of future work. For clarity, we would like to reiterate that while our model is trained in an environment that mimics the one in the original user study, the behaviour of real humans in that environment (i.e. the results of the study) was never considered during the design or training of the model. All similarities emerge from the skeletal kinematics, the muscle model and a simple cost function. The evaluation of both mid-air and surface-aligned conditions serves to illustrate how the model can generalise across both interaction settings.

Table 3. Results for mid-air interaction on 500 randomly sampled stimuli and mean result for all users from Dudley et al [11] (D_{ma2}).

Metric	D_{ma2}	Our model	z-score
IKI [ms]	306	342	0.4268
CER [%]	9.95	9.93	-0.0029
WPM	42.1	37.6	-0.4013

Table 4. Results for surface-aligned typing on 500 randomly sampled stimuli and mean result for all users from Dudley et al [11] (D_{sur2}).

Metric	D_{sur2}	Our model	z-score
IKI [ms]	244	306	0.7046
CER [%]	12.25	3.88	-0.7911
WPM	55.605	40.93	-0.729

As with any other user model, our model, while surprisingly accurate, does not approximate all aspects of human behaviour. As illustrated in Fig. 7, our model does not predict that real users make more errors when typing on a surface-aligned keyboard. The touch point distributions over the layout shown in Fig. 8 reveal a degree of consistency with Dudley et al. [11] in terms of additional variance in the lateral direction across the layout, but differ in terms of the generally higher variance observed with the mid-air model for keys near the centre of the layout. However, in the second part of our evaluation we will focus on whether the model is sufficiently accurate to be useful for keyboard design in practice.

4.2 Applications to Hyperparameter Selection

In the development of any system, a large number of design decisions have to be made. For example, when investigating the feasibility of a new keyboard layout in VR, factors such as key size, keyboard position and other parameters are often impractical or infeasible to be comprehensively evaluated in a user study. Instead, designers often rely

on initial tests with small sample sizes ($N \approx$ # authors) or simply do not optimise many of these parameters at all. We will now show how our model can be used objectively and efficiently to augment, or even replace, human testing. For this part of the evaluation we only consider mid-air typing.

In particular, we use our model to select hyperparameters for the map-based keyboard decoder introduced in Sect. 3.7. The decoder takes a sequence of touch points on a virtual keyboard as input and predicts the sequence of keys the user intended to hit.

Optimal hyperparameters are selected and evaluated as indicated in Fig. 5. Specifically, we seek to minimise the character error rate *after* corrections are applied by the map-based keyboard decoder. To differentiate from the CER values reported in Sect. 4.1 (based on closest key to touch point), we refer to this corrected character error rate as CER_{corr}. Note that in the experimental protocol for the transcription task applied by Dudley et al. [11], participants were not permitted to perform any active correction (i.e. no backspace/delete key was available) and so some remaining errors are to be expected.

We use Bayesian optimisation with a Gaussian process base estimator and a GP-Hedge acquisition function [4] to perform this hyperparameter optimisation. Bayesian optimisation does not rely on gradients and is therefore particularly well-suited for applications where sample acquisition is expensive. Instead, it approximates the objective using a surrogate model and selects the next points for evaluation based on some statistical metric (e.g. expected improvement). For an introduction to Bayesian optimisation we refer readers to Frazier [16]. The acquisition function is optimised using Limited-memory BFGS [33]. All parameters are initialised with uniform priors. Eight points are evaluated in parallel using the minimum-constant liar strategy [7].

To obtain a reliable estimate of our model's performance, we capture 10 different datasets of the model typing 160 stimulus phrases sampled from the original corpus. We run Bayesian optimisation on each of these datasets and compute the average CER_{corr} of the best set of hyperparameters from each optimisation on D_{ma2}. The results from this evaluation are illustrated in Fig. 9. In particular, the best hyperparameters found for the ten datasets captured using our model achieve an average CER_{corr} of 5.77% on D_{ma2}. We note that we are not primarily interested in this absolute value but rather in the relative difference between our model and real users.

4.2.1 Random Baseline

Realistically, most hyperparameters in real systems are never explicitly optimised. Instead they are arbitrarily chosen by the designers. While *gut feeling* is difficult to approximate we take inspiration from financial science, where portfolio allocations are often compared against a large number of randomly chosen portfolios made up of the same underlying assets [5, 8]. We show that our model can outperform random hyperparameter choices for the previously introduced keyboard decoder with high confidence. As a baseline, we randomly sample 400 sets of hyperparameters from the domain described in Sect. 3.7 and evaluate their performance on D_{ma2}. Comparing random choices to the intuition

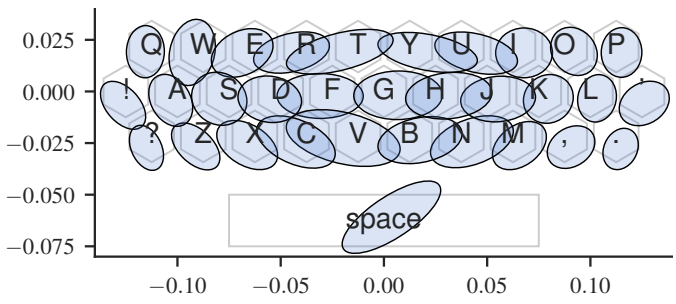

Fig. 8. Touch point covariance for each key represented as ellipses, which encompass the 50% confidence intervals. All labels are in metres. Points were collected using our mid-air model typing 500 random character sequences.

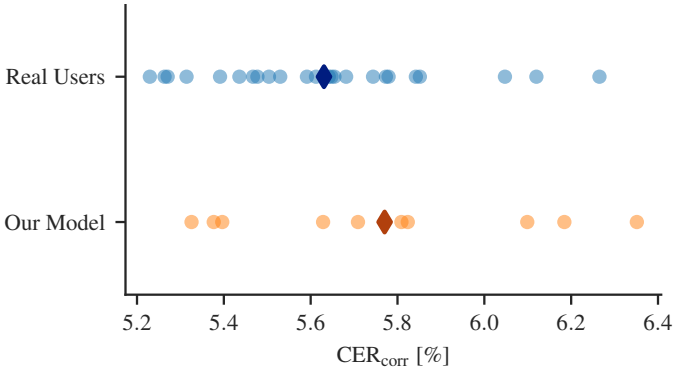

Fig. 9. Hyperparameter performance on D_{ma2}. Our model is shown to be nearly as predictive of hyperparameter performance as real users. This includes the best sets of hyperparameters found using 10 different datasets captured using our model, compared with the hyperparameter sets found using traces for each single real user.

of experienced engineers is of course a highly conservative simplification.Therefore, we consider only the best 25% of hyperparameters. Even so, in a pairwise comparison our model outperforms 99.8% of randomly sampled points. The 25% best randomly sampled points achieve an average CER_{corr} of 29.99% on D_{ma2}, compared to 5.77% for our model. The high CER_{corr} achieved using random sampling of the parameters illustrates how challenging the hyperparameter selection problem at hand is.

4.2.2 Single User Baseline

For the second part of this evaluation we now consider the use-case of a developer selecting hyperparameters based on their own data, a common tactic in practical design. In other words, we consider that our model may be used to select hyperparameters as good or better than some human users. We show that our model may in some cases be used to replace initial tests altogether with only a small real-world performance penalty. We again make use of the same keyboard decoder and the user data D_{ma2} from Dudley et al. [11]. Initial testing is simulated by using data from a single user to find hyperparameters that minimise the average CER_{corr} using Bayesian optimisation. This set is then evaluated on the data for the other 23 real users. This process is repeated for each of the 24 users.

As indicated above, our model achieves an average CER_{corr} of 5.77% on D_{ma2}, while the parameters found using real user data achieved an average CER_{corr} of 5.63%. The fact that our model performs nearly as well as real users is particularly impressive when considering how poorly the random baseline from Sect. 4.2.1 performed . Additionally, in a pairwise comparison our model has a 38.3% percent chance of outperforming any single user. This is particularly impressive, considering that this implies that (at least in terms of our evaluation) our synthetic model is more indicative of the behaviour of the average real user than 38.3% of actual real users.

From the benchmark user study [11] we know that collecting the data that we used during our evaluation took real users approximately two hours. Even if we assume that a researcher would be willing to invest this time to optimise a certain parameter, there is still a significant chance that our model will find better parameters.

Further, our model may be used to optimise parameters that realistically cannot be numerically optimised through human testing. In most cases, human testing can only be used to numerically optimise parameters that can be altered after the data is already captured (for example, the parameters of the keyboard decoder from our evaluation). In other cases (for example, altering the keyboard size) a human participant would have to type a large number of sentences for each level of the size parameter to be evaluated, which quickly becomes infeasible. Since our simulation can generate data for approximately 100 stimuli per minute, even on a laptop with an Intel i7-6700HQ CPU, fresh training data can easily be generated at each evaluation step.

5 Discussion

In this paper we have shown that a reinforcement learning agent controlling a biomechanical model of the human arm and trained on a simple cost function can approximate unseen user data with high accuracy. Importantly, the resulting model does not explicitly approximate human behaviour and it is not trained on any data from real users, or tuned on any evaluation metrics, such as entry or error rate. Instead, reinforcement learning allowed the system to learn a model that shows emergent behaviour aligned with human behaviour. It does so not by explicitly modelling how humans may behave, but through optimising its behaviour under similar underlying constraints as a human typist, including the skeletal structure and muscle model.

5.1 Implications

Advances in human-computer interfaces are often driven by stronger predictive methods. Touchscreen input already relies heavily on methods that predict the most likely user action (e.g. [18]). Considering the vastly more complex interactions that need to be well-supported in virtual and augmented reality, such as bimanual tracked movement of both hands and the individual fingers, predictive computational models are likely to form an important building block towards virtual and augmented reality user interface design that is more aligned with engineering science principles.

Many promising fluent, robust and flexible interaction techniques in virtual and augmented reality, such as mid-air virtual keyboard typing, will rely on data-hungry statistical models, such as deep neural networks. However, representative data from VR and AR headset users is typically limited. The generative RL model introduced in this paper acts as an inexpensive source of representative surrogate user data suitable for training a rich computational model and thereby paves the way towards developing such models in the future.

Another use of the RL model for mid-air virtual keyboard typing is as an interactive design tool for a use-case where little or no user data may be available. For example, our RL model can be used to carry out sensitivity analysis and alert designers to hyperparameter choices that should be more carefully considered during the design of a system. Further, as previously demonstrated, our RL model can readily support designers in finding suitable hyperparameters for a keyboard auto-correction decoder for two-finger mid-air virtual keyboard typing.

5.2 Limitations and Future Work

The most notable limitation of the RL model in this paper is that it uses two right arm models to simulate bimanual typing. This limits the model's ability to approximate the complex bimanual interleaving that real users exhibit [13]. However, the simplified approach is justified since adding a second arm would double the number of degrees of freedom and require the agent to dexterously coordinate both hands. A possible solution to account for this increase in difficulty would be to use imitation learning on real user data to warm-start the reinforcement learning [39]. However, such approaches rely on real user data, which reduces their availability for novel interactive systems, such as virtual and augmented reality, which is why we do not consider them in this work and leave such investigations for future work.

In this paper, we demonstrate the validity of the RL model by comparing the generated behaviours with existing user data. Our proposal to use the model for investigating the influence of other design parameters (e.g. key size, layout) still requires validation with comparable real user data. Unfortunately, collecting new user data at a large scale was not possible during this project due to the COVID-19 pandemic. Nevertheless, since the keyboard geometry is not explicitly considered in the model, we expect our approach to generalise well to a wide range of keyboard geometries and even to other target acquisition tasks. The generalisability of the reinforcement learning-based approach is exemplified by the fact that we were able to use the exact same RL model architecture, training regime and hyperparameters to type in both the mid-air and surface-aligned conditions.

It is likely that the accuracy of our model, particularly in terms of lower-level parameters, such as the 3D trajectory of the fingertip, could be improved by using a more complex biomechanical model. In particular, future work includes comparing different muscle models, or using the 3CC-r [34] model introduced in Sect. 3.2 and explicitly consider the fatigue predicted by that model in the cost function.

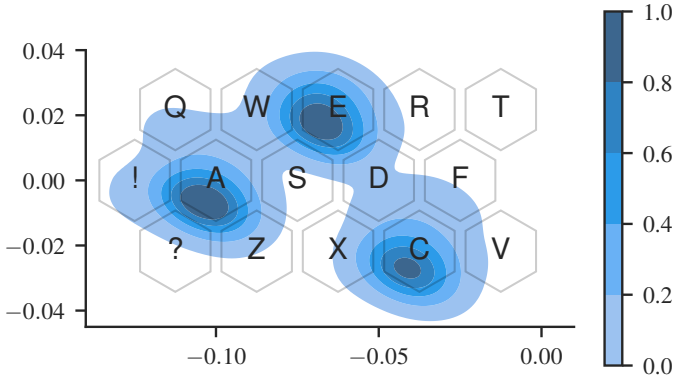

Fig. 10. Our model can generate highly variable traces: kernel density estimates of our model typing the word "ace" 500 times. All labels are in meters. The estimator bandwidth was computed using Scott's Rule [46].

The ability to generate large amounts of training data is only valuable if the model is sufficiently complex. In contrast to simple 2D user models [15], noise introduced into our RL model is mapped through the highly complex dynamics of the arm and throughout the episode to determine the trace captured for a specific stimulus. Therefore, even for the same stimulus phrase, our RL model can generate traces that are distinct in non-trivial ways (see Fig. 10). However, as evident in Fig. 6, our RL model does not currently capture the full variance of human behaviour. To further increase the variance of the generated data, the RL model can be extended with domain randomisation—a well-established practice in RL [38]. For example, skeletal and reward parameters can be randomised to train a large collection of individual agents. This ensemble can then be used to generate arbitrary amounts of valuable training data for training of, for example, a large recurrent neural network-based keyboard decoder for mid-air text entry. Previous work in reinforcement learning [38] suggests that the addition of domain randomisation during training can also significantly improve the generalisability of the model, making it more robust towards large changes in keyboard layout and geometry. We will investigate these aspects in future work.

6 Conclusions

Accurate modelling of user behaviour has the potential to substantially improve the quality of user interaction in virtual and augmented reality. All models of user interaction have to strike a careful balance between accuracy and complexity for them to be relevant beyond the scope of highly specialised research. At one extreme, a high-level mathematical regularity, such as Fitts' law, is easy to apply and research, and as a consequence has achieved widespread use. More complex handcrafted models are often ignored by practitioners and the research community because of their high complexity and relatively low value given the substantial effort required to tune an accurate model. Reinforcement learning-based models of interaction have the potential to capture much of the accuracy of these more complex models, while retaining most of the ease of use and generality of simple mathematical regularities. In particular, we believe RL models are particularly promising for modelling hand tracked user interaction in virtual and augmented reality.

In summary, in this paper we have demonstrated that reinforcement learning can approximate user behaviour in a complex interaction task that is ubiquitous in virtual and augmented reality: two-finger typing on a mid-air or surface-aligned virtual keyboard. Specifically, we have shown that our RL model is capable of replicating high-level typing behaviour of human users. Further, we have demonstrated the utility of this approach by showing that it may be used to augment or replace human testing during the validation and development of virtual keyboards by assisting with hyperparameter selection for an

auto-correcting virtual keyboard decoder. We hope this work will inspire further efforts into creating complex yet easy-to-use RL models of user interaction that can accelerate the development of more robust and fluent interaction techniques for virtual and augmented reality.

ACKNOWLEDGMENTS

This work was carried out when Lorenz Hetzel was a visiting student at the University of Cambridge. The authors would like to thank Otmar Hilliges for his advice and support. John Dudley and Per Ola Kristensson were supported by EPSRC (grant EP/S027432/1).

REFERENCES

[1] B. I. Ahmad, J. K. Murphy, P. M. Langdon, S. J. Godsill, R. Hardy, and L. Skrypchuk. Intent Inference for Hand Pointing Gesture-Based Interactions in Vehicles. *IEEE Trans Cybern*, 46(4):878–889, Apr. 2016.

[2] B. Baker, I. Kanitscheider, T. M. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch. Emergent Tool Use From Multi-Agent Autocurricula. *CoRR*, abs/1909.07528, 2019. _eprint: 1909.07528.

[3] D. A. Bowman, C. J. Rhoton, and M. S. Pinho. Text Input Techniques for Immersive Virtual Environments: An Empirical Comparison. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 46(26):2154–2158, 2002. _eprint: https://doi.org/10.1177/154193120204602611. doi: 10.1177/154193120204602611

[4] E. Brochu, M. Hoffman, and N. de Freitas. Hedging Strategies for Bayesian Optimization. Technical Report arXiv:1009.5419, Sept. 2010.

[5] P. Burns. Performance measurement via random portfolios. *Available at SSRN 630123*, 2004.

[6] N. Cheema, L. A. Frey-Law, K. Naderi, J. Lehtinen, P. Slusallek, and P. Hämäläinen. Predicting Mid-Air Interaction Movements and Fatigue Using Deep Reinforcement Learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–13. Association for Computing Machinery, New York, NY, USA, 2020.

[7] C. Chevalier and D. Ginsbourger. Fast Computation of the Multi-points Expected Improvement with Applications in Batch Selection. Oct. 2012.

[8] A. Clare, N. Motson, and S. Thomas. An evaluation of alternative equity indices-part 1: Heuristic and optimised weighting schemes. *Available at SSRN 2242028*, 2013.

[9] L. Colombo, D. Martín De Diego, and M. Zuccalli. Optimal control of underactuated mechanical systems: A geometric approach. *Journal of Mathematical Physics*, 51(8):083519, 2010. _eprint: https://doi.org/10.1063/1.3456158. doi: 10.1063/1.3456158

[10] V. Dhakal, A. M. Feit, P. O. Kristensson, and A. Oulasvirta. Observations on Typing from 136 Million Keystrokes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pp. 1–12. Association for Computing Machinery, New York, NY, USA, 2018. event-place: Montreal QC, Canada. doi: 10.1145/3173574.3174220

[11] J. Dudley, H. Benko, D. Wigdor, and P. O. Kristensson. Performance Envelopes of Virtual Keyboard Text Input Strategies in Virtual Reality. In *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 289–300, 2019. doi: 10.1109/ISMAR.2019.00027

[12] J. J. Dudley, K. Vertanen, and P. O. Kristensson. Fast and Precise Touch-Based Text Entry for Head-Mounted Augmented Reality with Variable Occlusion. *ACM Trans. Comput.-Hum. Interact.*, 25(6), Dec. 2018. Place: New York, NY, USA Publisher: Association for Computing Machinery. doi: 10.1145/3232163

[13] A. M. Feit, D. Weir, and A. Oulasvirta. How We Type: Movement Strategies and Performance in Everyday Typing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pp. 4262–4273. Association for Computing Machinery, New York, NY, USA, 2016.

[14] F. Fischer, M. Bachinski, M. Klar, A. Fleig, and J. Müller. *Reinforcement Learning Control of a Biomechanical Model of the Upper Extremity*. 2020. _eprint: 2011.07105.

[15] F. Fischer, A. Fleig, M. Klar, L. Gruene, and J. Mueller. *An Optimal Control Model of Mouse Pointing Using the LQR*. 2020. _eprint: 2002.11596.

[16] P. I. Frazier. A tutorial on bayesian optimization, 2018.

[17] C. Gebhardt, B. Hecox, B. van Opheusden, D. Wigdor, J. Hillis, O. Hilliges, and H. Benko. Learning Cooperative Personalized Policies from Gaze Data. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, UIST '19, pp. 197–208. Association for Computing Machinery, New York, NY, USA, 2019. event-place: New Orleans, LA, USA. doi: 10.1145/3332165.3347933

[18] J. Goodman, G. Venolia, K. Steury, and C. Parker. Language Modeling for Soft Keyboards. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*, IUI '02, pp. 194–195. Association for Computing Machinery, New York, NY, USA, 2002. event-place: San Francisco, California, USA. doi: 10.1145/502716.502753

[19] J. Grubert, L. Witzani, E. Ofek, M. Pahud, M. Kranz, and P. O. Kristensson. Text Entry in Immersive Head-Mounted Display-Based Virtual Reality Using Standard Keyboards. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 159–166, 2018. doi: 10.1109/VR.2018.8446059

[20] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. 2018. _eprint: 1801.01290.

[21] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. Soft Actor-Critic Algorithms and Applications. *CoRR*, abs/1812.05905, 2018. _eprint: 1812.05905.

[22] C. M. Harris and D. M. Wolpert. Signal-dependent noise determines motor planning. *Nature*, 394(6695):780–784, Aug. 1998.

[23] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne. Imitation learning: A survey of learning methods. *ACM Comput. Surv.*, 50(2), Apr. 2017. doi: 10.1145/3054912

[24] A. Ikkala and P. Hämäläinen. *Converting Biomechanical Models from OpenSim to MuJoCo*. 2020. _eprint: 2006.10618.

[25] Y. Jiang, T. Van Wouwe, F. De Groote, and C. K. Liu. Synthesis of Biologically Realistic Human Motion Using Joint Torque Actuation. *ACM Trans. Graph.*, 38(4), July 2019. Place: New York, NY, USA Publisher: Association for Computing Machinery. doi: 10.1145/3306346.3322966

[26] J. J. Knapik, J. E. Wright, R. H. Mawdsley, and J. Braun. Isometric, isotonic, and isokinetic torque variations in four muscle groups through a range of joint motion. *Physical therapy*, 63(6):938–947, 1983. Publisher: Oxford University Press.

[27] P. Knierim, V. Schwind, A. M. Feit, F. Nieuwenhuizen, and N. Henze. Physical Keyboards in Virtual Reality: Analysis of Typing Performance and Effects of Avatar Hands. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pp. 1–9. Association for Computing Machinery, New York, NY, USA, 2018. event-place: Montreal QC, Canada. doi: 10.1145/3173574.3173919

[28] P. O. Kristensson. *Discrete and Continuous Shape Writing for Text Entry and Control*. PhD thesis, Department of Computer and Information Science, Linköping University, Sweden, 2007.

[29] P. O. Kristensson. Five Challenges for Intelligent Text Entry Methods. *AI Magazine*, 30(4):85, Sept. 2009. Section: Articles. doi: 10.1609/aimag.v30i4.2269

[30] P. O. Kristensson. Next-Generation Text Entry. *Computer*, 48(7):84–87, 2015. doi: 10.1109/MC.2015.185

[31] E. Lank, Y.-C. N. Cheng, and J. Ruiz. Endpoint Prediction Using Motion Kinematics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pp. 637–646. Association for Computing Machinery, New York, NY, USA, 2007. event-place: San Jose, California, USA. doi: 10.1145/1240624.1240724

[32] C. J. Lin, B. T. Abreham, and B. H. Woldegiorgis. Kinematics of direct reaching in head-mounted and stereoscopic widescreen virtual environments. *Virtual Reality*, Mar. 2021. doi: 10.1007/s10055-021-00505-6

[33] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989. Publisher: Springer.

[34] J. M. Looft, N. Herkert, and L. Frey-Law. Modification of a three-compartment muscle fatigue model to predict peak torque decline during intermittent tasks. *J Biomech*, 77:16–25, 2018.

[35] K. L. Moore, A. F. Dalley, and A. M. R. Agur. *Clinically oriented anatomy*. Wolters Kluwer/Lippincott Williams & Wilkins Health, Philadelphia, 7th ed ed., 2014.

[36] V. Nair and G. E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pp. 807–814. Omnipress, Madison, WI, USA, 2010. event-place: Haifa, Israel.

[37] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang. Solving Rubik's Cube with a Robot Hand. *CoRR*, abs/1910.07113, 2019. _eprint: 1910.07113.

[38] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. W. Pachocki, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning Dexterous In-Hand Manipulation. *CoRR*, abs/1808.00177, 2018. _eprint: 1808.00177.

[39] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. *Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations*. 2018. _eprint: 1709.10087.

[40] M. Richardson, M. Durasoff, and R. Wang. Decoding Surface Touch Typing from Hand-Tracking. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, UIST '20, pp. 686–696. Association for Computing Machinery, New York, NY, USA, 2020. event-place: Virtual Event, USA. doi: 10.1145/3379337.3415816

[41] T. A. Salthouse. Perceptual, cognitive, and motoric aspects of transcription typing. *Psychological bulletin*, 99(3):303, 1986. APA.

[42] K. R. Saul, X. Hu, C. M. Goehler, M. E. Vidt, M. Daly, A. Velisar, and W. M. Murray. Benchmarking of dynamic simulation predictions in two software platforms using an upper limb musculoskeletal model. *Comput Methods Biomech Biomed Engin*, 18(13):1445–1458, 2015.

[43] D. Schneider, A. Otte, T. Gesslein, P. Gagel, B. Kuth, M. S. Damlakhi, O. Dietz, E. Ofek, M. Pahud, P. O. Kristensson, J. Müller, and J. Grubert. ReconViguRation: Reconfiguring Physical Keyboards in Virtual Reality. *IEEE Transactions on Visualization and Computer Graphics*, 25(11):3190–3201, 2019. doi: 10.1109/TVCG.2019.2932239

[44] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust Region Policy Optimization. *CoRR*, abs/1502.05477, 2015. _eprint: 1502.05477.

[45] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347, 2017. _eprint: 1707.06347.

[46] D. W. Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[47] N. A. Tillin, M. T. Pain, and J. P. Folland. Contraction type influences the human ability to use the available torque capacity of skeletal muscle during explosive efforts. *Proceedings of the Royal Society B: Biological Sciences*, 279(1736):2106–2115, 2012. Publisher: The Royal Society.

[48] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

[49] R. J. van Beers, P. Haggard, and D. M. Wolpert. The Role of Execution Noise in Movement Variability. *Journal of Neurophysiology*, 91(2):1050–1063, 2004. _eprint: https://doi.org/10.1152/jn.00652.2003. doi: 10.1152/jn.00652.2003

[50] M. van Otterlo and M. Wiering. Reinforcement Learning and Markov Decision Processes. In M. Wiering and M. van Otterlo, eds., *Reinforcement Learning: State-of-the-Art*, pp. 3–42. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. doi: 10.1007/978-3-642-27645-3_1

[51] P. D. Varcholik, J. J. LaViola, and C. E. Hughes. Establishing a baseline for text entry for a multi-touch virtual keyboard. *International Journal of Human-Computer Studies*, 70(10):657 – 672, 2012. doi: 10.1016/j.ijhcs.2012.05.007

[52] K. Vertanen, H. Memmi, J. Emge, S. Reyal, and P. O. Kristensson. 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*, pp. 659–668, 2015.

[53] C. Wu and Y. Liu. Queuing Network Modeling of Transcription Typing. *ACM Trans. Comput.-Hum. Interact.*, 15(1), May 2008. Place: New York, NY, USA Publisher: Association for Computing Machinery. doi: 10.1145/1352782.1352788

[54] R. Xiao, J. Schwarz, N. Throm, A. D. Wilson, and H. Benko. MRTouch: Adding Touch Input to Head-Mounted Mixed Reality. *IEEE Transactions on Visualization and Computer Graphics*, 24(4):1653–1660, 2018. doi: 10.1109/TVCG.2018.2794222

[55] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie, and Z. Li. DRN: A Deep Reinforcement Learning Framework for News Recommendation. In *Proceedings of the 2018 World Wide Web Conference*, pp. 167–176. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2018.