Breathing Life Into Biomechanical User Models

Aleksi Ikkala
Aalto University
Finland

Florian Fischer
University of Bayreuth
Germany

Markus Klar
University of Bayreuth
Germany

Miroslav Bachinski*
University of Bayreuth
Germany

Arthur Fleig
University of Bayreuth
Germany

Andrew Howes
University of Birmingham
United Kingdom

Perttu Hämäläinen
Aalto University
Finland

Jörg Müller
University of Bayreuth
Germany

Roderick Murray-Smith
University of Glasgow
Scotland

Antti Oulasvirta
Aalto University
Finland

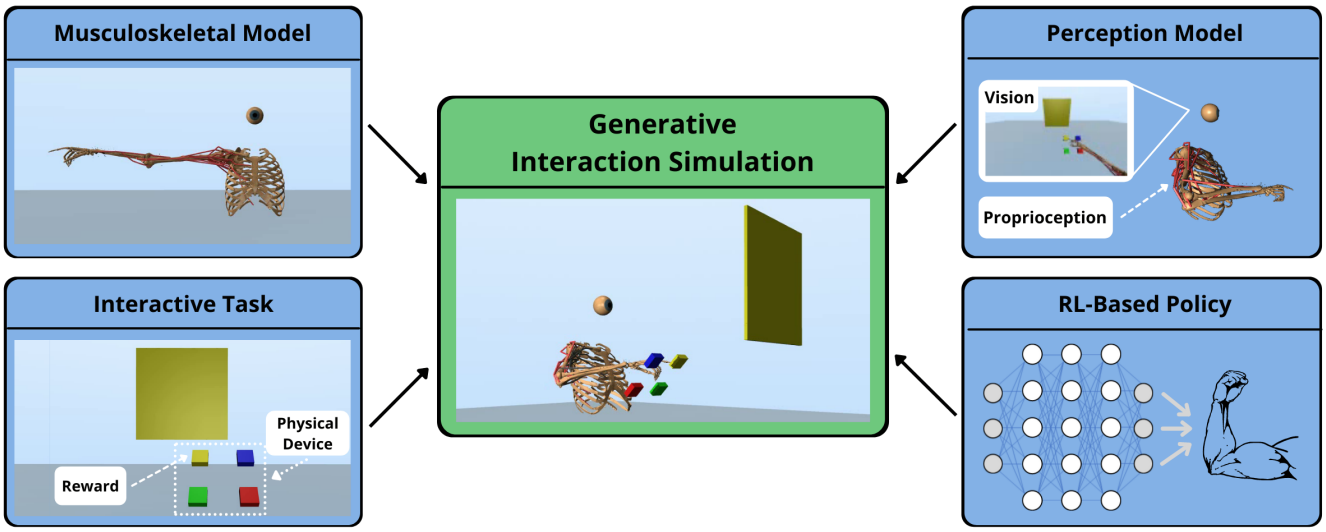


Figure 1: We present an approach for generative simulation of interaction with perceptually controlled biomechanical models interacting with physical devices. The users are modelled with a combination of muscle-actuated biomechanical models and perception models, and we use deep reinforcement learning to learn control policies by maximizing task-specific rewards. As a showcase, we apply a state-of-the-art upper body model to four HCI tasks of increasing difficulty: pointing, tracking, choice reaction, and parking a remote control car via joystick.

ABSTRACT

Forward biomechanical simulation in HCI holds great promise as a tool for evaluation, design, and engineering of user interfaces. Although reinforcement learning (RL) has been used to simulate biomechanics in interaction, prior work has relied on unrealistic assumptions about the control problem involved, which limits the plausibility of emerging policies. These assumptions include direct torque actuation as opposed to muscle-based control; direct, privileged access to the external environment, instead of imperfect sensory observations; and lack of interaction with physical input devices. In this paper, we present a new approach for learning muscle-actuated control policies based on perceptual feedback in interaction tasks with physical input devices. This allows modelling of more realistic interaction tasks with cognitively plausible visuomotor control. We show that our simulated user model successfully learns a variety of tasks representing different interaction methods, and that the model exhibits characteristic movement regularities observed in studies of pointing. We provide an open-source implementation which can be extended with further biomechanical models, perception models, and interactive environments.

*Also with University of Bergen.

UIST '22, October 29-November 2, 2022, Bend, OR, USA

ACM ISBN 978-1-4503-9320-1/22/10.
https://doi.org/10.1145/3526113.3545689

CCS CONCEPTS

• **Human-centered computing → Human computer interaction (HCI)**; **User models**; *Pointing*; **Systems and tools for interaction design**.

KEYWORDS

biomechanical models, simulation models, deep reinforcement learning

ACM Reference Format:
Aleksi Ikkala, Florian Fischer, Markus Klar, Miroslav Bachinski, Arthur Fleig, Andrew Howes, Perttu Hämäläinen, Jörg Müller, Roderick Murray-Smith, and Antti Oulasvirta. 2022. Breathing Life Into Biomechanical User Models. In *The 35th Annual ACM Symposium on User Interface Software and Technology (UIST '22), October 29-November 2, 2022, Bend, OR, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3526113.3545689

1 INTRODUCTION

Biomechanical models studied in HCI can be compared to "crash test dummies" [4, 5]. They are inactive agents, with each movement following a predefined motion plan; they do not have agency, and they cannot close the loop and explore their environments themselves. To simulate an interaction, a researcher may either manually define a set of actions, or collect motion capture data that is fit to the model using inverse simulation. What is missing is a *generative* form of simulation that would learn *realistic interaction policies* on its own through *forward simulations* in a given interactive task and without requiring prespecified motions. The ability to build a generative model that matches user behaviour is a strong test of whether we understand an interactive system. "Breathing life" into crash test dummies would open the door to wider use of these models, such as quickly evaluating prototypes before committing to an expensive experimental study. It would permit us to rapidly explore a more diverse set of user behaviours than would be feasible in experimental studies on humans, and such models could also enable parametric optimization of user interfaces. Biomechanical modeling is highly relevant for HCI research, because, with the exception of BCIs, all interfaces require physical effort from the user. Among others, physical effort is a critical consideration in the design of AR/VR interfaces, interactive surfaces, and haptic and tangible interfaces. Until recently, evaluating physical effort required running an empirical study (e.g., using NASA-TLX). However, with the advent of open access models, HCI research has recently turned to biomechanical simulations for studying and modeling interfaces as well as for developing novel interaction techniques (e.g., [4, 5, 10, 17, 29, 48]).

What has been missing is an integrative approach that would allow using reinforcement learning (RL) to learn human-like interaction policies in a designer-specified interactive task, where the task components (goals, user model, physical environment) can by flexibly changed (Figure 1). Instead, the approaches developed so far in HCI have been limited to a particular task (e.g., pointing) or a discrete combination of primitive tasks, to a particular model (e.g., upper body model), or to a particular physical environment. Moreover, with the exceptions that we discuss below, all biomechanical user models prior to this work have relied on unrealistic assumptions regarding their force actuation, ability to observe their environment, and interaction with input devices, all of which limit the realism of emerging motion patterns. First, those models operate with torque actuated joints, second, they operate without adequately perceiving their surroundings, and third, they avoid simulation of physical interaction with input devices by investigating contact-free interfaces, such as mid-air pointing (e.g., [10]). Torque actuation can be problematic because it allows for movements that are not achievable with muscles, and RL approaches will exploit these more efficient, but unrealistic, movements. Lack of visual perception can be problematic as it allows agents to, e.g., exactly know the position of a target even if it is occluded or outside the field of view. Without simulating physical interactions with input devices, the majority of interactions with computers cannot be simulated, as pure mid-air pointing is still a niche interaction technique. What is needed is a muscle-actuated user model that is able to receive and process perceptual observations of its surroundings – including vision, audition, haptics, proprioception, and so on – and physically interact with input devices. This coincides with the emerging RL challenge of learning control policies via perceptual inputs [43]. By combining perception models, musculoskeletal models, and physically simulated input devices, we can train agents to model and simulate intricate interaction tasks, such as those requiring visuomotor control. A good example of such control is presented in [51], where Nakada et al. introduced a virtual human model and used deep learning to learn reaching and tracking tasks. However, with more powerful physical simulation software, such as MuJoCo [73], we can simulate interaction steps quickly enough to use RL for more flexible problem formalisation and to allow a researcher to guide an agent's learning through reward functions.

The main contribution of this paper is twofold: 1) We integrate perception-based control and muscle-actuated biomechanical models that interact with physical input devices, and 2) we show how our RL-based approach can be leveraged to increase the scope of interactive tasks that can be simulated. The tasks present a variety of different user interfaces that require visuomotor movement control, and a user model is trained to interact with the environment through RL by rewarding desired behaviour. We show that the simulated user successfully learns to complete these interaction tasks, and the simulated movements exhibit human-like movement regularities such as Fitts' Law. This work is publicly available at https://github.com/aikkala/user-in-the-box as an extendable codebase. This implementation allows researchers to model interactive settings flexibly by changing assumptions about the musculoskeletal model, the user's task, and the perception models. It could be used in the future to e.g. aid in developing and evaluating user interfaces.

2 RELATED WORK

2.1 Biomechanical Modelling and Simulation

Biomechanical models and computer-based simulations were introduced more than four decades ago [2]. However, for a long time they were oversimplified and limited to computation of mechanical loads in static postures or using simple link-segment models [76]. Increases in computational power in the last two decades have enabled more physiologically-accurate musculoskeletal models [14, 15]. A sizable collection of such models exists now, many of which are

publicly available.[1] Depending on the use case, one might opt for a model that describes the functionality of a single limb [55, 63], or go for a more comprehensive full-body model [22, 59, 74]. In these models, the force generation mechanism may rely on motors actuating individual joints, or in a more realistic use case, on muscle-tendon units [23, 45].

The main use of biomechanical models at the moment is via inverse biomechanical simulation using the OpenSim ecosystem [65]. The inverse simulation methods, namely inverse kinematics, inverse dynamics, and static optimisation or computed muscle control, allow the estimation of mechanical loads within the human musculoskeletal system, and neural controls of the muscles given motion tracking data of a given user's movement as input [15]. Such estimated variables are increasingly valuable and important in the areas of medicine, rehabilitation, and sports. On the other hand, there exists a stream of computer graphics research on biomechanical simulation and control that emphasizes visual fidelity and simulation speed rather than validation for purposes such as medical or ergonomics research [37, 38, 51, 62, 67, 69]. Forward simulation methods were also developed within the OpenSim software, however, besides their use as a component of computed muscle control, they were rarely used as standalone. They require muscle controls as inputs, which are extremely complex to measure experimentally for all required muscles, and are typically used with controls computed by inverse simulation. Standalone forward simulation has only become more useful when applied in combination with computational controllers [16, 36].

Controlling the force output of a model's actuators in forward simulations to perform a desired movement is a difficult optimisation problem. In order to find the appropriate control signals one must be able to run simulations quickly — which is often infeasible for complex models with possibly dozens of degrees-of-freedoms and a high number of (muscle) actuators. This is especially problematic for RL approaches, where finding good solutions often require millions of simulation steps. This introduces a challenge for biomechanical simulation software, which typically has not been designed for such use cases. In our work, we convert models validated by biomechanics researchers into a faster simulator [28]. An alternative would be to use a simplified simulation model and apply machine learning to predict the omitted details such as state-dependent joint actuation torque limits and muscle-based energy expenditure [30]; however, this requires generating training data using a realistic simulator, and the learned prediction model is inherently less accurate and general than the simulator itself. Previous work has also built models themselves for a faster simulator [37]. The approaches to solve this optimisation problem range from well-understood classical optimal control methods (see Section 2.2) to cutting-edge methods such as deep RL (see Section 2.3).

2.2 Classical Optimal Control Methods for HCI

Mathematically, reinforcement learning (RL) can be interpreted as a method to solve optimal control problems. Optimal control is the optimization of a cost or objective function subject to some system dynamics, such as the biomechanical model of a human and the dynamics of an interactive system. The parameter to optimize is called the control or input signal and is usually a function of time, such as muscle excitations [68]. In addition to RL, human-computer interaction has also been interpreted and simulated with classical (optimal) control methods [50]. If the cost function is quadratic and the system dynamics are linear, e.g., as in [19, Ch. 7], then the go-to method to compute the optimal control is via the linear-quadratic regulator (LQR). (Gaussian) noise, e.g., in the observation or control of the system [19, Ch. 8], can be handled by the linear-quadratic-Gaussian (LQG) regulator, an extension of LQR. More recently, event-driven Intermittent Control (IC) has been introduced as a framework to better explain relevant aspects of human movement [44]. For other cost functions and nonlinear system dynamics in particular (e.g., a state-of-the-art biomechanical model), a viable approach is to use Model Predictive Control (MPC) [33, 58]. A major difference of these approaches to the RL-based approach of our paper is that RL computes an entire policy, which maps arbitrary observations to actions and can be used to generate approximately optimal movements quickly, while classical optimal control methods compute individual optimal movements.

2.3 Reinforcement Learning -Based User Modelling

RL algorithms solve sequential decision making problems where at every timestep, an agent observes the current state, takes an action, and receives an action- and state-dependent scalar reward. The goal is to select actions that maximize expected utility defined as the sum of future rewards. RL provides a suitable framework for modelling human behaviour in a flexible way: one only needs to define the states, actions, and rewards and then RL computes the optimal policy [12, 13]. In some cases, the reward function and other parameters can be inferred from human data [6, 32]. A more detailed RL problem formulation along with our definitions for states, actions, and rewards are discussed in Section 3.1. When the reward function and state–action space, including their key limitations, are similar to a human's, increasingly human-like behavior has been shown to emerge through learning [52]. Applications in HCI include models of typing, menu selection, multitasking, and visual decision-making [52]. However, no application in HCI so far has looked at perceptual control of a biomechanical model.

Using the assumptions of signal-dependent control noise and movement time minimization, Fischer et al. [18] have shown that an RL agent can learn to generate human-like movements with a torque-actuated state-of-the-art model of the upper extremity. The generated movements were in accordance with well-established phenomena such as Fitts' Law [20] and the Two-Thirds Power Law [35]. RL-based simulation may also provide valuable information for predicting usability- and ergonomics-related criteria and to aid in interface design. For instance, Cheema et al. trained a (simplified) torque-actuated biomechanical arm model in a mid-air pointing task, and used the model to predict fatigue of real human subjects performing the task [10]. Leino et al. [39] used RL to learn policies for keystroke-level models, and used this to optimize button arrangements. In addition to learning control policies for embodied agents, RL can be used to model users performing

[1]For instance, OpenSim models https://simtk-confluence.stanford.edu:8443/display/OpenSim/Musculoskeletal+Models

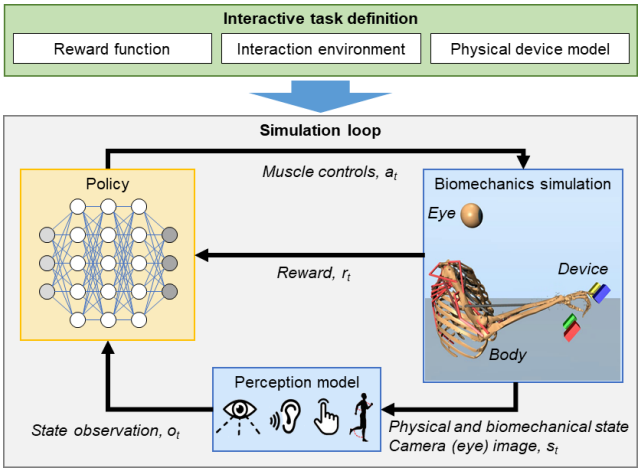


Figure 2: Our approach: The researcher specifies properties of an interactivate task (green box), including the reward function that guides the simulated user's learning process, the interaction environment, and the physical devices the user interacts with. The simulated user is defined through biomechanical and perception models, and is controlled by an RL policy based on observations from the perception models.

interactive tasks with symbolic actions. For instance, in [40, 66] RL agents interact with websites using vision and low-level mouse and keyboard actions. However, these methods rely on human demonstrations to learn policies. Recently, game companies have started applying RL-based user modelling in simulation-based game testing [34, 60, 61].

Advances in deep learning during the past decade have made it possible to scale up RL-based models. Deep RL has been used to learn control policies in increasingly complex state–action spaces — such as torque-actuated humanoids [7, 18, 53, 77] and musculoskeletal systems [37, 51], as well as eye-hand coordination in typing [31]. Hetzel et al. [25] presented an RL agent for simulating joint-controlled movement of hands in typing, however lamenting that while muscle control would have been preferable, they were not able to train a model that had muscles. Moveover, their control problem was not perceptual like ours; their agent state was a vector describing joint kinematics and the position of next target key. Nakada et al. [51] demonstrated how deep artificial neural networks (ANNs) can be leveraged to enable visuomotor control of biomechanical simulation for tasks like target tracking. However, rather than RL, they used a supervised learning approach that utilised a task- and stimuli-specific training process that is not suitable for flexibly modelling a variety of interaction tasks.

3 OVERVIEW OF APPROACH

We present User-in-the-Box (UitB), an extendable open-source implementation for biomechanical user modelling in interactive tasks in the MuJoCo [73] simulator.[2] UitB enables flexible modelling of muscle-actuated, perceptually controlled biomechanical user models with deep RL. It allows flexibly changing the physical model of the interactive device via 3D models that can be implemented in MuJoCo, or imported from another modelling software, e.g. Unity (Figure 2). The implementation can be extended with additional biomechanical models, for instance, by converting them from OpenSim [28], perception models, and interactive tasks. The user model learns to interact with the environment through a task-specific reward function.

To use UitB, one begins by defining an interactive task. This includes defining the reward function, which guides the agent's learning process, the interaction environment, and the physical devices that the user interacts with. Then, one creates the user model by choosing existing biomechanics and perception models, or implementing new ones. Models of perception can be formed by defining transducing functions from e.g. the output of an egocentrically placed camera, or other sensors available in the MuJoCo simulator. Once the interactive task and the user model are defined, the RL agent is trained using deep RL, and the simulated user's performance can be evaluated.[3] This approach forces the simulated user to adhere to important assumptions about low-level perception and movement characteristics inherent in human physiology while retaining enough flexibility to adapt the user model, through learning, to different interactive tasks. Learning control policies with RL requires an efficient forward physics simulator; in this work we use MuJoCo as a simulation environment, but ultimately this approach is simulator-agnostic.

In this section, we will first frame the problem of user modelling in interactive tasks as an RL problem and then discuss what aspects one should consider when creating an interactive task or a user model.

3.1 Modelling an Interactive Task with RL

The computational core of our framework is reinforcement learning. To utilize RL, an interactive task needs to be modelled as a Markov decision process (MDP) [70], or more generally, a partially observable MDP (POMDP). As illustrated in Figure 2, this means that at every timestep t, the agent takes an action a_t based on a state observation o_t. This results in receiving a scalar reward r_t and a new observation o_{t+1}, from which a new action a_{t+1} is performed.

We utilize a discounted RL objective with a stochastic policy and episodic learning. This means that we optimize a policy $\pi(a_t|o_t)$ to maximize the expected return $\mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$, where the actions are sampled from the policy, $a_t \sim \pi(a_t|o_t)$. The discount factor $\gamma \in [0, 1)$ controls how much earlier rewards are preferred to distant rewards. Learning progresses through episodes, where the simulation is returned to an initial state at $t = 0$ and actions are simulated up to the time limit T.

3.1.1 Reward Function. In UitB, a key modelling aim is to select a reward function that represents whatever the user tries to achieve and values in interaction. For instance, in a pointing task, human subjects would be asked to point to a target; in an RL setting, this can be implemented by rewarding behaviour where the fingertip is brought on top of the target in minimum time and with minimum

[2] available at https://github.com/aikkala/user-in-the-box

[3] We use term *simulated user* to refer to the performance of the user model during simulation, and term *agent* — originating from the AI and RL literature — when referring to RL training or evaluation.

effort. The effort term is denoted as δ_t, and it is assumed to be a part of the reward r_t the user receives. UITB allows flexibly defining the reward function by reference to measures available in the simulation. The reward function is task-specific, and while designing it can be non-trivial, the reward functions of our four simulation tasks presented in Section 4 should provide a useful starting point.

3.1.2 Observations. To allow multisensory perception, we define our observations as a tuple $o_t = (\Omega_t, \xi_t)$, where $\Omega_t = (\omega_t^1, \omega_t^2, ...)$ is a tuple of outputs from different perception models, such as vision or proprioception, with ω_t^i being the output of i:th perception model. For the sake of clarity, we will refer to the outputs of proprioception and vision models, which are used in all of the tasks, as ω_t^P and ω_t^V, respectively. In some of the tasks we also model tactile feedback through force sensors: this tactile perception is denoted by ω_t^T. Furthermore, we introduce a stateful information observation ξ_t as a part of o_t. This quantity may contain information related to the task, e.g., how much time is left until an episode terminates. In tasks where the agent needs to, for instance, infer movement direction we may also include perception model outputs from previous timesteps in the observation, such that $o_t = (\Omega_t, \xi_t, \Omega_{t-1}, \xi_{t-1}, ..., \Omega_{t-k}, \xi_{t-k})$, where k denotes how many previous observations are included.

3.1.3 Actions. Our neural network policy does not directly output a vector of muscle controls a_t. Instead, we use *relative muscle control*, sampling relative action vectors a'_t from a Gaussian policy as

$$a'_t \sim \pi(a'_t|o_t) = \mathcal{N}(\mu_\theta(o_t), \mathrm{diag}(\sigma^2)), \tag{1}$$

where θ are the parameters of the policy neural network, and μ and σ^2 are mean and variance vectors, respectively. The σ^2 controls the exploration/exploitation tradeoff.[4] The final muscle controls are computed as

$$a_t = clip_0^1(m_{t-1} + clip_{-1}^1(a'_t)), \tag{2}$$

where m_{t-1} are the model's internal muscle activation states for the previous step, which are included in the proprioceptive observations ω^P, and $clip_x^y$ denotes a clipping operation to range $[x, y]$. Essentially, the policy is controlling whether muscle excitation for the next step should be higher or lower than internal muscle activation in the previous step, i.e., whether the agent should increase or decrease force output of a specific muscle actuator. In particular, applying zero control results in constant internal muscle activation, which lets the body converge towards a "steady-state" posture. According to our experience, this type of control leads to a significant speed-up in the training of policies for muscle-actuated models. This is in contrast to using an *absolute muscle control*, where a policy would output the muscle excitation signals directly in range $[0, 1]$.

3.1.4 Algorithm Choice. Depending on the POMDP formulation, one typically has multiple alternative RL algorithms to choose from. In this paper, we utilize the Stable Baselines 3 library's [57] implementation of Proximal Policy Optimization (PPO [64]), which works for both discrete and continuous states and actions, and is the most popular RL algorithm in both recent HCI works [10, 31] and high-quality human movement control papers in the computer animation literature [7, 53, 77]. Note that as per Stable Baselines 3 defaults, our standard deviations σ are optimized together with the policy network parameters θ, starting from a high value to allow thorough initial exploration of the state and action spaces.

The majority of RL algorithms, including PPO, are designed for fully observable MDPs, where the observations o_t are replaced by states s_t. The assumption is that s_t contains all the information for choosing the optimal actions. In many real-world tasks, this is not the case, but standard RL methods can still be applied, either by using a recurrent policy network [75] that can learn to infer the true state from a sequence of observations, or engineering each observation to include enough information for the inference. For instance, if a game-playing agent only observes a single frame of pixels at a time, the observation is only informative of object positions; inferring velocities can be enabled by concatenating two or more consecutive frames as the observation [47].

3.2 Interactive Task Definition

The interactive task definition includes the reward function, and models of the environment and related interaction devices.

Even when not modeling a human, the reward function is often the most difficult part of an RL problem to specify. When modeling a human, the reward function must not only be cognitively plausible but it must also facilitate efficient learning. In general interaction tasks are easier to formulate with sparse rewards, but this makes it more difficult to solve the RL optimisation problem. For instance, one could give reward only when an interaction is completed satisfactorily, like once an agent has put its fingertip inside a target in a pointing task. In practice, however, the RL problem is often made easier by using reward shaping (e.g. reward is a function of distance between fingertip and target in a pointing task), early termination (terminate an episode early if the agent is in some sense moving further from a goal), or curriculum learning (start the learning process with a simplified version of the problem).

The interaction environment is defined as a MuJoCo model that contains one or multiple interactive devices. The physical devices can be modelled as a set of MuJoCo primitives, or one could import a 3D model of a device into MuJoCo as a triangulated mesh. The device may contain physically moving parts, which need to be modelled with joints, or it may include dynamically changing content, like the color or size of an interaction device. However, some aspects of interaction may be difficult — though not impossible — to model in MuJoCo, such as the exact type of friction between contacts, or a touch screen with dynamic content.

3.3 User Model

With *user model* we refer to the combination of a biomechanical model, any set of perception models, and a control policy; all the components that are required to model and simulate a user.

The user model can be flexibly defined based on how the simulated user needs to interact with its environment. For instance, when simulating a mid-air pointing task, one mainly needs to model the movement of arm and shoulder, and vision system. The perception models can be implemented depending on the level of realism required in the modelling. For instance, the vision system could be modelled with one RGB-D camera, or two RGB cameras with overlapping fields of view. The RL policy represents a cognitive

[4] In the future, it would be interesting to add signal-dependent control noise [24] as an additional source of variance.

model that receives perceptions from the environment, and decides how to control the muscle actuators of the biomechanical model.

The perception models receive full and perfect knowledge of the simulation's physical state, biomechanical state included. The role of these models is to bound information that cannot be expected to be available to a user, and hence the user receives a transformed observation of the environment. For example, in a pointing task the user would not know the exact coordinates where a target is located, but instead has to infer the target location from visual observations. Furthermore, humans' perceptions of the world are rarely perfect; this noise modelling could be included in the perception models. However, the perception models used in our simulations are relatively simple and noise-free. One could extend the user model with more intricate perception models, for instance, e.g., by implementing foveal and peripheral vision and adding eye movements.

The user model may also include an effort term or other types of fatigue modelling to drive the agent behave in a more human-like fashion. As mentioned in Section 3.1.1, the effort term is a part of the reward the agent receives by interacting with the environment. The exact form of a suitable effort term is not known, and, as Berret et al. [8] showed in an arm movement modelling setting, the most appropriate cost function may be a combination of several different functions. In our simulations we use a neural effort term [8], which we introduce in Section 4.1. However, the UITB implementation allows these to be easily changed to investigate the effects of different effort terms.

4 SIMULATION STUDIES

In the following subsections, we show applications in a diverse and challenging set of interaction tasks (Figure 3). We first provide details of the biomechanical model used to simulate movement dynamics, and the perception models that allow the simulated user to observe its environment. Then we describe the interaction environment for the standard HCI task of mid-air pointing and analyse the simulated movements. We show that the simulated pointing movement complies with predictive models of human movement such as Fitts' Law and the Minimum Jerk model to a sufficient degree for this approach to be a valuable tool in evaluating behaviour in interactive tasks. Finally, we show that our simulated user successfully learns to perform three additional HCI tasks of varying difficulty: target tracking, choice reaction, and parking a remote control car via joystick.

4.1 Model Implementation and Training

We use *MoBL ARMS model* [63], a state-of-the-art muscle-actuated upper extremity model, originally created in OpenSim [15], to model arm movements in a set of interaction tasks. This model includes seven degrees-of-freedom (DoF) to represent the movements of shoulder, elbow, and wrist. In contrast to point-mass or linked-segment models, which are widely used to simulate user behavior [24, 71, 72], the MoBL ARMS model includes translational and rotational coupling between body segments, physiological joint axis orientations, and joint angle limits. The model is actuated by 50 Hill-type muscle-tendon actuators [45].

We converted the model to MuJoCo using the *O2MConverter* [28], which creates an approximate replica of the original OpenSim model in MuJoCo. MuJoCo allows for much faster forward simulations of the interactive environment, while including more sophisticated contact dynamics. Some of the limitations of the converted model are that MuJoCo tendons are inelastic, and tendon paths are limited to fixed sites as opposed to having dynamically moving and conditional pathpoints or wrapping objects as in OpenSim. Excluding these limitations, the MuJoCo model is anatomically as accurate; and MuJoCo and OpenSim use the same muscle activation dynamics and exhibit comparable force-length-velocity-curves.

In order to decrease the state and action space dimensionality of the RL optimisation problems, we disabled two wrist DoFs (wrist flexion and deviation), which were not instrumental in the interaction tasks. Furthermore, we disabled 24 muscle actuators that mainly actuated finger movements, which were deemed unnecessary, as there were no DoFs in the model's fingers. Therefore, 5 DoFs and 26 muscle actuators remained to represent the kinematics and dynamics of the arm. In order to ensure that our simulation can only achieve reasonable body postures, we modified the equality constraint that couples elevation angle and shoulder rotation in the MuJoCo model by adding an additional dependency on shoulder elevation (details are given in Suppl. Mat. S2). The fingers of the model were modified such that index finger is extended, while the rest are flexed.

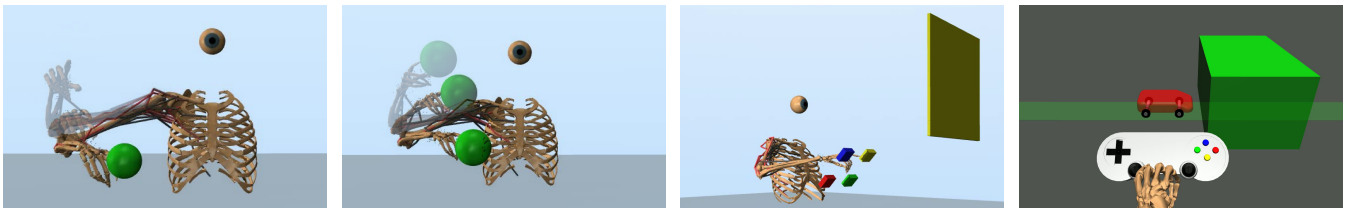
The proprioceptive observations ω^P contain joint angles, velocities, and accelerations for the five DoFs, and muscle internal activation states m for the 26 muscle actuators. As all considered tasks require precise movement of the end-effector, we also included Cartesian coordinates of the tip of the index finger in the proprioceptive observations. The joint angles and muscle internal activations are normalised to range [-1, 1].

The visual observation ω^V is rendered from an RGB-D camera with a resolution of 120×80 pixels (Figure 4 shows an example of a visual observation). The "eye camera" was placed 20 cm above the torso, approximately where one's head would be located. For simplicity, we decided to fixate the camera position, resulting in a constant field of view in the same direction. In each of the tasks we use either one or multiple color channels with or without depth channel, depending on what kind of information the agent requires to successfully learn the task. In some tasks we also include prior visual observations to allow the agent to infer movement velocity. The image data is normalised to range [-1, 1] for each channel.

Furthermore, in choice reaction and parking tasks we have included tactile observation ω^T of the fingertip. This force sensor lets the simulated user know how much force it is exerting through contacts. The force value is a non-negative scalar.

The policy network π contains a convolutional neural network to encode the high-dimensional visual observations into lower dimensional representations. The other observations are vectors which are concatenated and passed through a separate encoder, before being concatenated with the encoded visual observations. This representation is then passed through two fully connected layers to produce the mean vectors μ_θ, which are then used to sample relative action vectors a'_t cf. (1). Network architecture details are given in Suppl. Mat. S1.

We chose to use a *neural effort* term [8] to constrain unnecessary movements, as a similar term is often used in RL when learning



(a) Pointing (b) Tracking (c) Choice Reaction (d) Parking a Remote Control Car

Figure 3: Four interactive tasks with differing perceptual-motor requirements. The figures show the MoBL ARMS model, and the RGB-D camera that serves as a visual system.

control policies. At each timestep t we compute the effort term

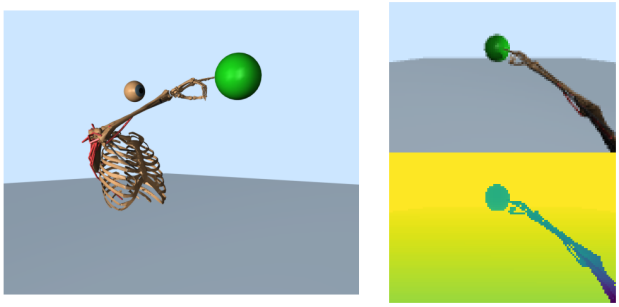
$$\delta_t = \sum_{i=1}^{N} a_{t,i}^2, \tag{3}$$

which represents neural strain from controlling motor neurons of the $N = 26$ muscle actuators, cf. (2). As mentioned earlier, the effort term δ_t is part of the reward r_t, i.e. it is subtracted from the proposed reward functions.

The simulation timestep in MuJoCo is set to 2 milliseconds, and actions are sampled from the policy with a frequency of 20 Hz during training, and 100 Hz during evaluation. According to our observations, this mismatch of action frequency sampling between training and evaluation has minimal effect on the results. Training with lower sampling frequency makes the training faster and mitigates the credit assignment problem [46], while evaluation with higher action frequency is required for some of the movement analysis.

4.2 Case Study: Pointing

Pointing is one of the most intensely studied interactive tasks in HCI. In a pointing task, users are asked to move a physical or virtual end-effector towards some object, e.g., a target sphere of given size. In this case study we demonstrate perception-based muscle control in a setup that corresponds to the well-known ISO pointing task variants.



(a) (b)

Figure 4: (a) A scene during a pointing task from an external camera. (b) The same scene rendered from the RGB-D camera, RGB image on top and depth image on bottom. Both images are 120x80 pixels.

In our model of this task, the end-effector corresponds to the tip of the index finger and the target is a penetrable sphere of varying radius located in front of the simulated user. The radius and location of the target are sampled randomly during training: radius from a continuous interval [5, 15] cm, and location from a 2D plane of size 60 cm × 60 cm. The origin of the plane is located 55 cm in front of the agent's shoulder and 10 cm to the right, in order to make targets easily reachable in all areas of the plane. The agent must keep its fingertip inside a target for 500 milliseconds to successfully "hit" the target. A new target location is sampled when a target is hit, or after four seconds if the agent fails to hit the target. The new target location is sampled with rejection sampling such that the distance between two consecutive targets is typically more than 30 cm. However, a new tentative location is sampled maximum ten times, so in rare cases the distance may be less than 30 cm. A total of fourteen targets are spawned during one episode of training. The location of the target plane and dwell time of 500 milliseconds were chosen to try and match the experimental conditions of a reciprocal ISO pointing task presented in [33], which allows us to compare our simulations to their human data, specifically to user U1. In order to make the comparisons more fair, the MoBL ARMS model was anatomically scaled to better match the anatomical dimensions of user U1 (only in this task).

At time t, the simulated user observes $o_t = (\omega_t^P, \omega_t^V, \xi_t)$. The visual observation ω_t^V contains only depth information, as color information is not necessary for completing this task. The stateful information ξ_t comprises of two quantities: how many targets are left in the episode, and how many milliseconds the fingertip has been inside a target. Both quantities are normalised to range [-1, 1], and either is not necessary to learn the task, but do speed up the training process. Following the same rationale as for typical experimental instructions, we want the simulated user to complete the task of hitting 14 targets as quickly as possible. Thus, the reward function is a mixture of two components: a negative reward, shaped by the distance between the agent's fingertip and target, issued as long as the target has not been reached, and a positive bonus for hitting the target. The reward function is

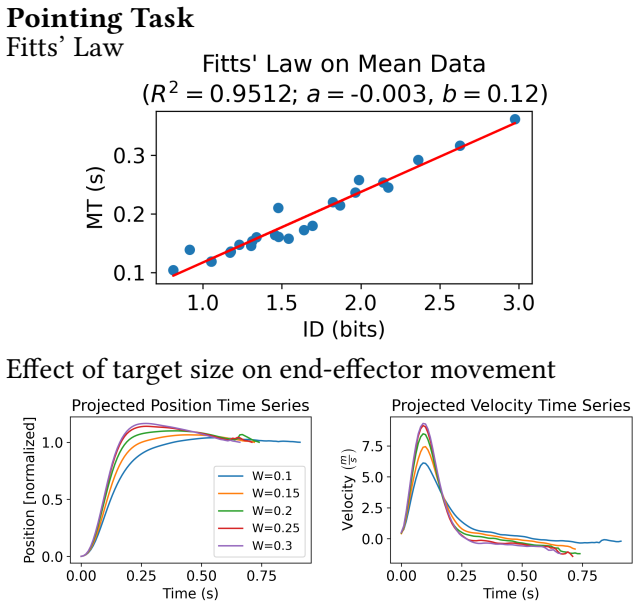


Figure 5: Simulated end-effector trajectories reproduce the Fitts' Law. That is, there is a clear effect of target size W on average peak velocity and total movement duration. Overshoot (i.e., projected position > 1) increases with target size.

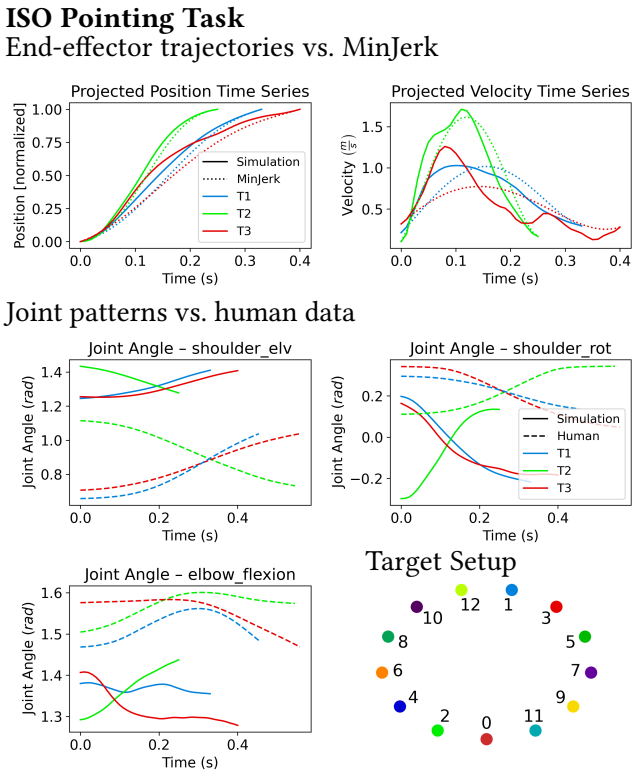


Figure 6: In our simulation of the ISO cyclical pointing task, targets are reached with a single ballistic movement (solid lines; upper plots). The projected position and velocity time series are close to the Minimum Jerk trajectories (dashed lines; upper plots). The joint angles of both shoulder and elbow resulting from our simulation (solid lines; lower plots) exhibit the same patterns as observed in the user study (dashed lines; lower plots). As expected, movement direction has a strong effect on qualitative behavior (representative movements to targets 1, 2, and 3 are shown, respectively).

$$r_t = \begin{cases} 8 - \delta_t & \text{if target is hit} \\ 0 - \delta_t & \text{if fingertip is inside target} \\ (e^{-d_t * 10} - 1)/10 - \delta_t & \text{otherwise,} \end{cases} \quad (4)$$

where d_t is the distance between fingertip and target surface and δ_t is the effort term (3) at time t.

4.2.1 Performance metrics. We collected a dataset of movements by running the policy for 100 episodes with randomly sampled target radii and locations. We denote the time between two sampled targets as one trial. During one episode the agent is presented with 14 targets, therefore we have a total of 1,400 trials in the dataset. The simulated user hit 1,395 targets (a success rate of 99.64%), and on average it took the agent 690 ms to finish one trial.

4.2.2 Fitts' Law and speed-accuracy trade-off. Fitts' Law is a well-established regularity for pointing and target acquisition tasks, claiming a linear relationship between the difficulty and the average movement time required to reach a given target [20, 42]:

$$\text{MT} = a + b\text{ID} = a + b \log_2 \left(\frac{D}{W} + 1 \right). \quad (5)$$

Here, D and W are the initial distance to target and the size (diameter) of the target sphere, respectively, ID denotes the Index of Difficulty in bits (using the *Shannon Formulation* [41]), and MT is the predicted movement time. In order to verify whether the end-effector trajectories produced by our simulated user follow Fitts' Law, we binned the 1,395 evaluation trials described in Section 4.2.1 into 25 groups, using 5 quantile-based partitions of each distance and target width. For each group, we computed the average movement time per trial, and used the resulting combinations of ID and MT to identify the model parameters $a = -0.003$ and $b = 0.12$ via linear regression. As can be seen in the upper plot of Figure 5, our simulation trajectories (blue dots) are consistent with the linear relationship predicted by Fitts' Law (red line), explaining more than 95% of the between-group variance ($R^2 = 0.9512$).

To analyze the specific effect of target size on the simulation trajectories, we simulated five movements between the same two target locations for five different target sizes W. This was repeated for eight pairs of targets, resulting in a total of 40 movements per target size. To isolate the effect of target size from those of confounding variables such as movement direction or different trials, we projected each end-effector trajectory onto the respective direct path between initial and target position and then computed the *mean projection* for each target size. This was done by computing the average projected position and velocity of all 40 movements at

each timestep, and concatenating the resulting position and velocity time series. As shown in the lower plots of Figure 5, there is a clear effect of target size on the mean projections, both in terms of end-effector position and velocity. As target size decreases, movements become slower, resulting in both a lower average peak velocity and a larger average movement time (9.31 m/s and 660 ms for 30 cm width vs. 6.12 m/s and 900 ms for 10 cm width). Also note that there is a clear tendency to overshoot for large target sizes. This shows that the speed-accuracy trade-off typically observed in aimed movements towards spatially constrained targets (and which is also consistent with Fitts' Law) is inherent to our simulation. Albeit the confidence intervals of these mean projections overlap (not shown in the figure), the effect is consistently seen across different movements.

4.2.3 Minimum Jerk. One of the best-known models to describe the kinematics of human aimed movements is the Minimum Jerk (MinJerk) model proposed by Flash and Hogan [21]. This model assumes that humans aim to generate smooth end-effector trajectories, which is equivalent to minimizing the change in end-effector acceleration over time, denoted as *jerk*. While the MinJerk model does not make any predictions of the underlying human body and interaction dynamics, cannot account for corrective submovements, and requires movement duration as well as initial and terminal positions, velocities, and accelerations to be known in advance, it has been successfully used for modelling perturbed reaching [26] and word-gesture keyboard typing [56].

We replicated the experimental setup of a previous user study [33], where 13 target spheres were equidistantly arranged according to the ISO 9241-9 ergonomics standard (see target setup in Figure 6). In the original experiment, all ISO targets were always visible with the active target highlighted, whereas in our version of the task only the active target was visible. As opposed to the previous task where targets were randomly sampled, now the target radius was fixed to 5 cm, and the target location was chosen according to the ISO protocol. We used the same policy as previously (trained with random targets) to control the agent in this task, as ISO pointing effectively is a subset of the more general pointing task. In Figure 6, projected simulation trajectories for three representative movements – from T0 to T1, T1 to T2, and T2 to T3 – are shown for the surge phase, i.e., since the former target was hit and until the latter target was first reached. Both the projected position and velocity time series (solid lines in upper plots) match the corresponding minimum jerk trajectories (dotted lines in upper plots) visually well, suggesting that the targets are reached with a single ballistic movement. The simulation trajectories exhibit the symmetric, bell-shaped velocity profiles during the surge phase that are characteristic of mid-air pointing [49]. More quantitative comparisons between our method and MinJerk would not be particularly meaningful due to the different assumptions and goals described above.

4.2.4 Comparison to Human Data. To identify whether body postures of our simulations are comparable to those of humans, we computed the joint angles that best explain the movements of user U1 observed in the ISO task user study in [33] via *Inverse Kinematics*, and compared them to the joint angles inferred from our simulation. Note that the trajectories which we compare, e.g. starting from T0 and ending in T1, begin when the simulated agent (or user U1) has hit T0, that is, the agent's (user U1's) fingertip is inside said target. Similarly the trajectory ends when the agent's (user U1's) fingertip is inside T1, and hence the maximum distance between the agent's and the user's initial and final fingertip positions is less than 10 cm (twice the target radius). These comparisons aim to provide qualitative evidence of the simulated agent's movements with respect to actual human movements. As can be seen in the lower plots of Figure 6, the general patterns of each shoulder elevation, shoulder rotation, and elbow flexion match considerably well between our simulation (solid lines) and the human reference (dashed lines). In particular, the body postures required to reach high targets (T1, T3; blue and red lines) are clearly different from those required for low targets (T2; green lines), which our simulation captures well. The largest differences between simulation and human data occur in terms of used joint ranges and movement duration. Both of these were expected, as we did not explicitly set our simulation to the initial body posture of the respective user, and we did not optimize the neural effort cost such that the absolute movement times would match human data.

4.3 Demonstrations: Tracking, Choice Reaction, and Parking a Remote Control Car

Here we further demonstrate that our approach is suitable to modelling and simulating a wide range of interactions that include perception and physical contact. We provide a description of each task, followed by relevant performance metrics to show that the simulated user learns

- complex muscle-actuated visuomotor control in an emergent fashion, simply based on task-specific rewards,
- to utilize prior observations to anticipate movement (tracking and parking tasks),
- to discriminate between different responses, and choose a correct response based on observed stimuli (choice reaction task),
- to control objects that have non-trivial (sixth order) dynamics (parking task using a joystick).

4.3.1 Tracking. In the tracking task the agent's objective is to follow a moving target with its fingertip as closely as possible. The environment here is very similar to the one used in pointing task: the target is confined to a 2D plane of size 60 cm × 60 cm in front of the agent, but the target is not static and target radius is fixed to 5 cm. The target follows a trajectory that is a mixture of five sine waves with varying amplitudes, frequencies, and phases. The amplitudes are uniformly sampled from interval [1, 5], and frequencies from [0, 0.5], and episode length is fixed to 10 seconds. We used a curriculum learning approach for this task, where the targets are initially fixed for the first 15 million training steps, and between 15 million and 40 million training steps the frequencies are sampled from range [0, f_{max}], where f_{max} linearly increases from 0 to 0.5.

To allow the simulated user to anticipate the target's movements, we included a past visual observation as input to the policy. The observation is then $o_t = (\omega_t^P, \omega_t^V, \omega_{t-k}^V)$, where k is chosen such that the past observation is 100 milliseconds prior to the current

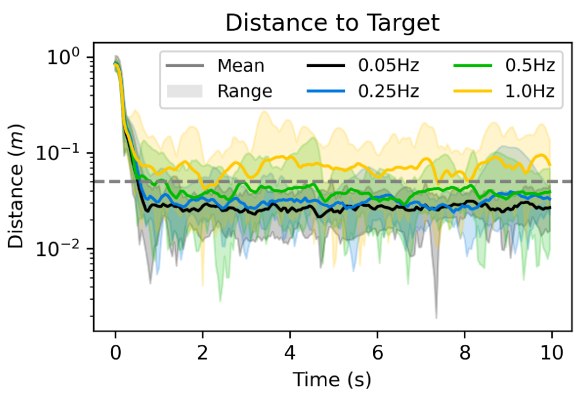


Figure 7: Distance between the agent's fingertip and target center as a function of time. After a fast initial movement towards the target, the agent is able to keep track of the target, even if the fingertip is temporarily outside the target (values above the horizontal dashed line). Each of the considered frequencies that define the movement pattern of the target is given in a different color.

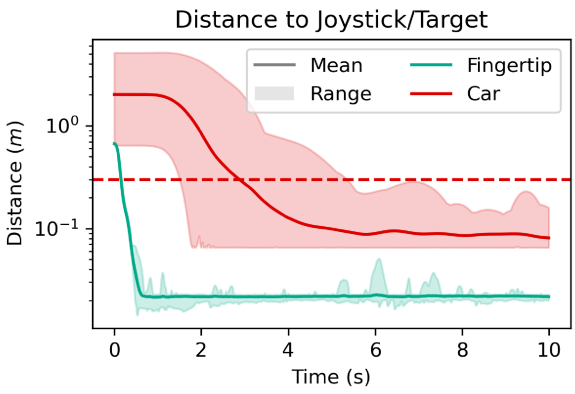


Figure 8: Distance between the agent's fingertip and the joystick (teal), as well as distance between the controlled car and target (red) as a function of time. After moving the hand towards the joystick, which takes approx. 800 milliseconds, the joystick is used to steer the car inside the target box (values below the horizontal dashed line, showing the 30 cm target size constantly used during evaluation).

one. As in the pointing task, the visual observation ω^V contains only depth information. The reward function for this task is simply

$$r_t = -d_t - \delta_t, \tag{6}$$

where d_t is the distance between fingertip and the target surface at time t, and δ_t is the effort term (3).

Figure 7 shows the distance between fingertip and target origin as a function of time on a logarithmic scale. Initially the average distance is large, as the agent's arm is besides its torso in the starting position. The distance drops quickly as the agent starts to track the target, and stays close to the target for the rest of the episode. To analyze the effect of the target speed, we considered four different frequencies f_{max} (0.05, 0.25, 0.5, and 1 Hz), with 10 episodes created for each frequency. The solid lines in Figure 7 correspond to the respective average values, the filled areas to the complete ranges. While slower targets are clearly easier to track, the agent is able to keep the fingertip mainly inside the target up to frequencies of 0.5 Hz. For 1 Hz movements, which the agent has not seen during training, the fingertip is short behind the target most of the time, resulting in a consistently small offset.

4.3.2 Choice Reaction. In a choice reaction task a participant is presented with several responses, and is required to choose between those responses when observing a stimulus. In our simulated version of this task (see Figure 3(c)), the agent is presented with four different colored buttons, and a screen to show the stimulus – all in field of view of the agent. The training procedure is similar to the pointing task: the agent has four seconds to press a button and receive a positive reward. When a button has been pressed with suitable force, or four seconds have passed, the screen changes color and indicates which button should be pressed next. The agent is presented with a stimulus and expected to choose a response ten times in one episode.

In this task the simulated user receives an observation $o_t = (\omega_t^P, \omega_t^V, \omega_t^T, \xi_t)$. The stateful information ξ_t contains one quantity: the number of targets left in the episode, normalised to range $[-1, 1]$. The reward function is akin to (4) used in the pointing task:

$$r_t = \begin{cases} 8 - \delta_t & \text{if target button is pressed} \\ (e^{-d_t * 10} - 1)/10 - \delta_t & \text{otherwise,} \end{cases} \tag{7}$$

where d_t is the distance between the fingertip and the center of the target button at time t, and δ_t is the effort term (3).

While the simulated user learns to press the appropriate buttons successfully, it does so in a rather quick manner. The average time to finish the episode is 3.94 seconds with a standard deviation of 0.91 seconds (averaged over 100 episodes). Only in four trials out of 1000 the agent was unable to respond within four seconds of observing the stimulus. While this behaviour is in the realm of possibilities for a human, it is likely faster than an average human subject would perform. One explanation for this could be the somewhat unrealistic visual model that neither models selective attention nor peripheral vision. Instead of having to alternate focus between a button and the screen, the simulated user is able to perceive all objects at the same time. It is also possible that, since the simulated user and the buttons are fixed in space, the user learns the locations of the buttons based on proprioceptive observations instead of visual observations.

4.3.3 Parking a Remote Control Car. As another interaction task, we trained the agent to steer a remote control car using a joystick (see Figure 3(d)). The goal of this interaction task is to park the car inside the green box. The initial positions of the car and the target are sampled from a green line in front of, and fully visible to,

the agent. The car moves only in one dimension, along the green line, and its acceleration/deceleration is controlled by tilting the left joystick of the gamepad forward or backward. The length of an episode is fixed to ten seconds. Note that this task differs from previous tasks in terms of difficulty twofold. First, it requires fine muscle control since the joystick and required movements are relatively small. Second, in addition to controlling the biomechanical model, the agent has to learn the second-order dynamics of the car resulting in a higher complexity (sixth-order in total).

The observation for this task is $o_t = (\omega_t^P, \omega_t^V, \omega_{t-k}^V, \omega_t^T)$, where k is chosen such that the past visual observation ω_{t-k}^V is 100 milliseconds prior to the current one. To speed up the training, the visual observation contains only the red color channel. The tactile perception ω_t^T contains force reading of contact between fingertip and the joystick to aid the agent in estimating how much the joystick needs to be tilted to move the car.

The reward function is

$$r_t = D(fingertip, joystick) + D(car, target)/10 + B_{joystick,fingertip} + B_{car,target} - \delta_t, \quad (8)$$

where $D(x, y) = e^{-d_t(x,y)*3} - 1$ is a function of distance d_t between x and y at time t, $B_{x,y}$ are bonus terms, and δ_t is the effort term (3). The bonus $B_{fingertip,joystick} = 0.8$ is given only once per episode, for the first time when the fingertip touches the joystick. The bonus $B_{car,target} = 8$ is granted if the car is inside the target with a velocity less than 0.1 m/s.

We evaluated the agent's performance over 50 episodes. Figure 8 shows the distances between agent's fingertip and joystick, and car and target, as functions of time.[5] The figure shows that it takes less than three seconds, on average, to move the car inside the target (values below red dashed line), and in all 50 episodes the car is successfully parked inside the target by the end of the episode.

5 SUMMARY AND DISCUSSION

We implemented perception-based muscle-actuated biomechanical models in MuJoCo, and demonstrated how an RL approach can be leveraged to simulate human-like behaviour in different interaction tasks with physically simulated input devices. The user model successfully learned to complete a variety of interaction tasks, while also producing movements that comply with predictive models of human movement, such as Fitts' Law. Such models could be used to evaluate user interfaces *in silico*, before or instead of, running evaluation studies with human subjects. Also, use of simulation may enable a more rigorous way to ensure diversity is taken into account during design. The models are available in the release of the User-in-the-Box open-source implementation, available at https://github.com/aikkala/user-in-the-box.

5.1 Reward–Model Interactions

RL provides flexibility in formulating an interaction task as an optimisation problem, but it does require experience with RL to develop an appropriate reward function, and its formulation will affect the final model outcome. As this is a new approach to generating representative models of human behaviour for HCI, there is not yet a mature workflow for refining the reward function design for a given task. Our open-source implementation introduces a new problem domain for researching the effect of a utility function, i.e., the reward function, on model behaviour. For instance, based on our observations, the scaling of a reward (not including the effort term) did not often incur major differences in the simulated user's behaviour, while e.g. the (non-)linearity of a reward function did.

In our simulation studies we employed well-known strategies for making the optimisation problem easier: early termination, reward shaping, and curriculum learning. We used a form of early termination in pointing and choice reaction tasks, where a new target was spawned every 4 seconds; reward shaping in all of the reward functions to guide the agent towards desired behaviour; and curriculum learning in the tracking task, where the agent first learns to point towards a fixed target, and eventually the target begins to move. Arzate Cruz and Igarashi's survey [1] reviews reward function design for interactive applications.

Our primary concern in this paper was to define reward functions for which policies could be learned efficiently. The reward functions in our tasks consisted of two components, a distance component and an effort component. The former guided the agent towards desired body postures and is *task-specific*, although the idea of using some sort of distance reward can be applied to many tasks. The effort term included in the reward, on the other hand, is a *task-agnostic* component that served to steer the agent to interact with the environment in a specific way, i.e., with minimum effort. A third component, time, comes into play implicitly via negative rewards. With negative rewards the agent is incentivized to finish a task quickly, if the episode length is not fixed (for example, the pointing and choice reaction tasks). If one uses only positive rewards the task completion may be unnecessarily prolonged, as there is no incentive to finish the episode promptly, especially if rewards far in the future are not heavily discounted using a low discount factor γ. To reiterate, all our reward functions share negative distance and effort components, while the positive bonus terms are connected to milestones or completion of the task.

The complexity of the task being modelled plays a significant role in finding an efficient reward function. In the pointing, tracking, and choice reaction tasks the agent learned a good control policy robustly without search for an exact scaling or parameterisation of the reward function. However, in the parking task it took us multiple iterations to find a reward function that produced a successful policy. Further study is required to find best practices for efficient reward function design.

Further, although the learned policy captures human behaviour in a number of ways (see Section 4), it is not known how well these reward functions model human subjective utility functions. Indeed, we have not tried to 'fit' the parameters of the reward function to human data. We anticipate that future work will need to take on this challenge. Future work should be inspired by what is known about human subjective utility. For example, it is known that people are sensitive to externally imposed speed/accuracy trade-offs [27, 78] but that people vary in *how* sensitive they are, with some preferring to be more accurate and others preferring to be fast.

[5] Note that the distance between car and target center cannot fall below a certain threshold, as it is always measured from the most distant wheel.

5.2 Hierarchical Controllers

Another frontier concerns computational efficiency. The required number of training steps until convergence in the first three tasks was typically 40-80 million steps, which required 24-48 hours to train with 10 parallel workers and a GPU. The agent in the last task (parking a remote control car) was trained for 130 million steps, which required 94 hours. In each of the simulation tasks we train the agent from scratch, which means that the agent must always learn again how to move its arm. We believe that training time could be significantly decreased by using a hierarchical RL approach, where the optimisation problem is made easier by first training a separate task-agnostic low-level controller to control movements of the agent, and then training task-specific higher level policies to solve the actual RL problem [3, 37, 54].

5.3 Increasing Realism

We see multiple ways to increase the realism of biomechanical and perception models for HCI research. While advancing significantly in the last decade, modern biomechanical models are developed as mechanical systems involving multiple assumptions and simplifications in comparison to the natural human body: e.g. (in order of decreasing severity) an activation optimality assumption that excludes muscle co-contraction, static muscle states that ignore fatigue effects, solid movement mechanics ignoring soft tissues, generic weight distribution based solely on rigid segment properties, continuous excitation signal instead of motor unit size-based control, or simplified hinge-based joint mechanics instead of slide & roll movement with complex 3D transformation. These simplifications can lead to the unnaturalness of generated movements, particularly ones involving fine motor control or under fatigued muscles, and deviations in predicting injury risks or fatigue. Considering the above simplifications, the modern models can simulate with reasonable accuracy most movements, except fine motorics involving co-contraction of opposite muscles, such as writing. Our biomechanical model, although more sophisticated than previous HCI models, is only a representation of the upper torso with shoulder and arm movement. There are no wrist or finger movements included. The perceptual observations were based on rather rudimentary models: the visual system was represented by a low-resolution RGB-D camera with a constant field of view, tactile perception was based on a single force sensor, and none of the perception models included noise modelling. Furthermore, MuJoCo as the chosen physics simulator might not be suitable to model some interaction devices, such as a touch screen with dynamic content. However, model development is typically allocated limited resources, and it is necessary to stop development once one has a model that works well enough for a given task. In this situation, it is important to document the qualities of the model for others to build on in future. A practical challenge is that a biomechanical model which initially appears to predict the human data well may have inaccuracies which first become apparent when used for optimisation in the RL process, as the inaccuracies are 'exploited', leading to unnatural behaviour.

While movements simulated in the pointing task shared many characteristics found in human data, the simulated movement differed in some aspects. For instance, the joint ranges of our simulated user were different from the joint ranges obtained with inverse kinematics of a human user, and the movement speed of the simulated user was slightly faster. However, it is unclear what the best choice of distance measure is between simulation states and observations of human poses, and how accurate replications of human data need to be for practical applications such as user interface evaluations. This is likely to vary based on use case.

6 CONCLUSION

We believe that perceptual control of biomechanically plausible human models is the key to more extensive use of simulations in the field of HCI. Perception has such a significant role in interaction that oversimplifying it in models may have stalled progress in studies of motor control in HCI. Linking perception and muscle-based control is necessary for understanding both low-level phenomena in HCI, such as bimanual control and eye-hand coordination when using input devices, but also higher-level phenomena, such as the emergence of fatigue-avoiding strategies in AR/VR applications.

While cognitive models have been at the heart of HCI since its inception [9] and significant progress has been made [11–13, 31, 52], one enduring limitation has been the lack of an *end-to-end* framework for predicting and explaining interaction. In these cognitive approaches, perception and biomechanics are modelled either as black box symbolic input/output functions or with mathematical laws (e.g. Fitts' Law) that do not simulate the processes of embodiment and as a consequence much of real-world interaction is left unexplained. In contrast, the approach to modelling that we have proposed in the current paper embraces perception and biomechanics as a key locus of explanation but, arguably, neglects the role of cognition. Future work should seek to combine the strengths of both approaches. For example, one could seek to explain not only how people use perception and biomechanics to point-and-click, but also how they use such skills to navigate, browse, acquire information, make decisions, and collaborate. RL, and particularly hierarchical RL, offers a framework for such an extension.

Finally, we believe that simulations of the kind discussed in this paper can help the scientific process in HCI research. The formal rigour required in creation of a simulation model, and controlling and documenting the provenance of knowledge and data used to calibrate it, makes clear the importance of many of the often poorly described aspects of context in HCI experiments. A simulation package is also easily shared with other researchers, improving reproducibility via an unambiguous implementation of the current scientific theory, the predictions of which can be validated with observed real-world data. We have made some effort to describe the weaknesses of our model, because aspects of models which at any given stage are poorly justified theoretically, are a poor fit to experimental data, or which are highly sensitive to context can be viewed as prompts to the research community about where they need better theories, more complex models, or more data. This can create improved clarity, and a shared awareness of the open problems and challenges, and can help document progress.

ACKNOWLEDGMENTS

A.I. is funded by the Academy of Finland Flagship programme "Finnish Center for Artificial Intelligence" (FCAI). R.M-S. acknowledges funding support from EPSRC grant EP/R018634/1, *Closed-loop Data Science* and the Academy of Finland via FCAI. A.O. is supported by Academy of Finland project *Human Automata*.

REFERENCES

[1] Christian Arzate Cruz and Takeo Igarashi. 2020. *A Survey on Interactive Reinforcement Learning: Design Principles and Open Challenges.* Association for Computing Machinery, New York, NY, USA, 1195–1209. https://doi.org/10.1145/3357236.3395525

[2] M. A. Ayoub, M. M. Ayoub, and A. G. Walvekar. 1974. A Biomechanical Model for the Upper Extremity using Optimization Techniques. *Human Factors* 16, 6 (1974), 585–594. https://doi.org/10.1177/001872087401600603 arXiv:https://doi.org/10.1177/001872087401600603 PMID: 4442903.

[3] Amin Babadi, Michiel Van de Panne, Caren Liu, and Perttu Hämäläinen. 2021. Learning Task-Agnostic Action Spaces for Movement Optimization. *IEEE Transactions on Visualization and Computer Graphics* (2021), 1–1. https://doi.org/10.1109/TVCG.2021.3100095

[4] Myroslav Bachynskyi, Antti Oulasvirta, Gregorio Palmas, and Tino Weinkauf. 2014. Is Motion Capture-Based Biomechanical Simulation Valid for HCI Studies? Study and Implications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 3215–3224. https://doi.org/10.1145/2556288.2557027

[5] Myroslav Bachynskyi, Gregorio Palmas, Antti Oulasvirta, Jürgen Steimle, and Tino Weinkauf. 2015. Performance and Ergonomics of Touch Surfaces: A Comparative Study Using Biomechanical Simulation. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 1817–1826. https://doi.org/10.1145/2702123.2702607

[6] Nikola Banovic, Tofi Buzali, Fanny Chevalier, Jennifer Mankoff, and Anind K. Dey. 2016. *Modeling and Understanding Human Routine Behavior.* Association for Computing Machinery, New York, NY, USA, 248–260. https://doi.org/10.1145/2858036.2858557

[7] Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. 2019. DReCon: Data-Driven Responsive Control of Physics-Based Characters. *ACM Trans. Graph.* 38, 6, Article 206 (nov 2019), 11 pages. https://doi.org/10.1145/3355089.3356536

[8] Bastien Berret, Enrico Chiovetto, Francesco Nori, and Thierry Pozzo. 2011. Evidence for Composite Cost Functions in Arm Movement Planning: An Inverse Optimal Control Approach. *PLOS Computational Biology* 7, 10 (10 2011), 1–18. https://doi.org/10.1371/journal.pcbi.1002183

[9] Stuart K. Card, Thomas P. Moran, and Allen Newell. 1983. *The psychology of human-computer interaction.* Crc Press.

[10] Noshaba Cheema, Laura A. Frey-Law, Kourosh Naderi, Jaakko Lehtinen, Philipp Slusallek, and Perttu Hämäläinen. 2020. *Predicting Mid-Air Interaction Movements and Fatigue Using Deep Reinforcement Learning.* Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376701

[11] Xiuli Chen, Aditya Acharya, Antti Oulasvirta, and Andrew Howes. 2021. An adaptive model of gaze-based selection. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems.* 1–11.

[12] Xiuli Chen, Gilles Bailly, Duncan P Brumby, Antti Oulasvirta, and Andrew Howes. 2015. The Emergence of Interactive Behaviour: A Model of Rational Menu Search. (2015).

[13] Xiuli Chen, Sandra Dorothee Starke, Chris Baber, and Andrew Howes. 2017. A cognitive model of how people make decisions through interaction with visual displays. In *Proceedings of the 2017 CHI conference on human factors in computing systems.* 1205–1216.

[14] Michael Damsgaard, John Rasmussen, Søren Tørholm Christensen, Egidijus Surma, and Mark de Zee. 2006. Analysis of musculoskeletal systems in the AnyBody Modeling System. *Simulation Modelling Practice and Theory* 14, 8 (2006), 1100–1111. https://doi.org/10.1016/j.simpat.2006.09.001 SIMS 2004.

[15] Scott Delp, Frank Anderson, Allison Arnold, Peter Loan, A. Habib, Chand John, Eran Guendelman, and Darryl Thelen. 2007. OpenSim: Open-Source Software to Create and Analyze Dynamic Simulations of Movement. *Biomedical Engineering, IEEE Transactions on* 54 (12 2007), 1940 – 1950. https://doi.org/10.1109/TBME.2007.901024

[16] Christopher L. Dembia, Nicholas A. Bianco, Antoine Falisse, Jennifer L. Hicks, and Scott L. Delp. 2021. OpenSim Moco: Musculoskeletal optimal control. *PLOS Computational Biology* 16, 12 (12 2021), 1–21. https://doi.org/10.1371/journal.pcbi.1008493

[17] João Marcelo Evangelista Belo, Anna Maria Feit, Tiare Feuchtner, and Kaj Grønbæk. 2021. XRgonomics: Facilitating the Creation of Ergonomic 3D Interfaces. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 290, 11 pages. https://doi.org/10.1145/3411764.3445349

[18] Florian Fischer, Miroslav Bachinski, Markus Klar, Arthur Fleig, and Jörg Müller. 2021. Reinforcement learning control of a biomechanical model of the upper extremity. *Scientific Reports* 11, 1 (2021), 1–15.

[19] Florian Fischer, Arthur Fleig, Markus Klar, and Jörg Müller. 2022. Optimal Feedback Control for Modeling Human-Computer Interaction. *ACM Trans. Comput.-Hum. Interact.* (2022). https://doi.org/10.1145/3524122

[20] Paul M Fitts. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology* 47, 6 (1954), 381.

[21] Tamar Flash and Neville Hogan. 1985. The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of neuroscience* 5, 7 (1985), 1688–1703.

[22] Sam Hamner, Ajay Seth, and Scott Delp. 2010. Muscle contribution to propulsion and support during running. *Journal of biomechanics* 43 (10 2010), 2709–16. https://doi.org/10.1016/j.jbiomech.2010.06.025

[23] Blake Hannaford and Jack Winters. 1990. *Actuator Properties and Movement Control: Biological and Technological Models.* Springer New York, New York, NY, 101–120. https://doi.org/10.1007/978-1-4613-9030-5_7

[24] C. M. Harris and D. M. Wolpert. 1998. Signal-dependent noise determines motor planning. *Nature* 394 (09 1998), 780–4. https://doi.org/10.1038/29528

[25] Lorenz Hetzel, John Dudley, Anna Maria Feit, and Per Ola Kristensson. 2021. Complex Interaction as Emergent Behaviour: Simulating Mid-Air Virtual Keyboard Typing using Reinforcement Learning. *IEEE Transactions on Visualization and Computer Graphics* 27, 11 (2021), 4140–4149. https://doi.org/10.1109/TVCG.2021.3106494

[26] Bruce Hoff and Michael A. Arbib. 1993. Models of Trajectory Formation and Temporal Interaction of Reach and Grasp. *Journal of Motor Behavior* 25, 3 (1993), 175–192. https://doi.org/10.1080/00222895.1993.9942048 arXiv:https://doi.org/10.1080/00222895.1993.9942048 PMID: 12581988.

[27] Andrew Howes, Richard L Lewis, and Alonso Vera. 2009. Rational adaptation under task and processing constraints: implications for testing theories of cognition and action. *Psychological review* 116, 4 (2009), 717.

[28] Aleksi Ikkala and Perttu Hämäläinen. 2022. Converting Biomechanical Models from OpenSim to MuJoCo. In *Converging Clinical and Engineering Research on Neurorehabilitation IV*, Diego Torricelli, Metin Akay, and Jose L. Pons (Eds.). Springer International Publishing, Cham, 277–281.

[29] Hasan Iqbal, Seemab Latif, Yukang Yan, Chun Yu, and Yuanchun Shi. 2021. Reducing arm fatigue in virtual reality by introducing 3D-spatial offset. *IEEE Access* 9 (2021), 64085–64104.

[30] Yifeng Jiang, Tom Van Wouwe, Friedl De Groote, and C Karen Liu. 2019. Synthesis of biologically realistic human motion using joint torque actuation. *ACM Transactions On Graphics (TOG)* 38, 4 (2019), 1–12.

[31] Jussi Jokinen, Aditya Acharya, Mohammad Uzair, Xinhui Jiang, and Antti Oulasvirta. 2021. Touchscreen typing as optimal supervisory control. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems.* 1–14.

[32] Antti Kangasrääsiö, Kumaripaba Athukorala, Andrew Howes, Jukka Corander, Samuel Kaski, and Antti Oulasvirta. 2017. Inferring Cognitive Models from Data Using Approximate Bayesian Computation. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 1295–1306. https://doi.org/10.1145/3025453.3025576

[33] Markus Klar, Florian Fischer, Arthur Fleig, Miroslav Bachinski, and Jörg Müller. 2022. Simulating Interaction Movements via Model Predictive Control. https://doi.org/10.48550/ARXIV.2204.09115

[34] Jeppe Theiss Kristensen, Arturo Valdivia, and Paolo Burelli. 2020. Estimating player completion rate in mobile puzzle games using reinforcement learning. In *2020 IEEE Conference on Games (CoG)*. IEEE, 636–639.

[35] Francesco Lacquaniti, Carlo Terzuolo, and Paolo Viviani. 1983. The law relating the kinematic and figural aspects of drawing movements. *Acta Psychologica* 54, 1 (1983), 115 – 130.

[36] Leng-Feng Lee and Brian R Umberger. 2016. Generating optimal control simulations of musculoskeletal movement using OpenSim and MATLAB. *PeerJ* 4 (2016), e1638.

[37] Seunghwan Lee, Moonseok Park, Kyoungmin Lee, and Jehee Lee. 2019. Scalable Muscle-Actuated Human Simulation and Control. *ACM Trans. Graph.* 38, 4, Article 73 (July 2019), 13 pages. https://doi.org/10.1145/3306346.3322972

[38] Sung-Hee Lee and Demetri Terzopoulos. 2006. Heads up! Biomechanical modeling and neuromuscular control of the neck. In *ACM SIGGRAPH 2006 Papers.* 1188–1198.

[39] Katri Leino, Antti Oulasvirta, and Mikko Kurimo. 2019. RL-KLM: Automating keystroke-level modeling with reinforcement learning. In *Proceedings of the 24th International Conference on Intelligent User Interfaces.* 476–480.

[40] Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. 2018. Reinforcement Learning on Web Interfaces using Workflow-Guided Exploration. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=ryTp3f-0-

[41] I. Scott MacKenzie. 1989. A Note on the Information-Theoretic Basis for Fitts' Law. *Journal of Motor Behavior* 21, 3 (1989), 323–330. https://doi.org/10.1080/00222895.1989.10735486 arXiv:https://doi.org/10.1080/00222895.1989.10735486 PMID: 15136269.

[42] I. Scott MacKenzie. 1992. Fitts' law as a research and design tool in human-computer interaction. *Human-computer interaction* 7, 1 (1992), 91–139.

[43] Richard S Marken and Warren Mansell. 2013. Perceptual control as a unifying concept in psychology. *Review of General Psychology* 17, 2 (2013), 190–195.

[44] J. Alberto Álvarez Martín, Henrik Gollee, Jörg Müller, and Roderick Murray-Smith. 2021. Intermittent control as a model of mouse movements. *ACM Transactions on Computer-Human Interaction (TOCHI)* 28, 5 (2021), 1–46.

[45] Matthew Millard, Thomas Uchida, Ajay Seth, and Scott L Delp. 2013. Flexing computational muscle: modeling and simulation of musculotendon dynamics. *Journal of biomechanical engineering* 135, 2 (2013).

[46] Marvin Minsky. 1961. Steps toward artificial intelligence. *Proceedings of the IRE* 49, 1 (1961), 8–30.

[47] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. (12 2013).

[48] Roberto A Montano Murillo, Sriram Subramanian, and Diego Martinez Plasencia. 2017. Erg-O: Ergonomic optimization of immersive virtual environments. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*. 759–771.

[49] Pietro Morasso. 1981. Spatial control of arm movements. *Experimental brain research* 42, 2 (1981), 223–227.

[50] Jörg Müller, Antti Oulasvirta, and Roderick Murray-Smith. 2017. Control theoretic models of pointing. *ACM Transactions on Computer-Human Interaction (TOCHI)* 24, 4 (2017), 1–36.

[51] Masaki Nakada, Tao Zhou, Honglin Chen, Tomer Weiss, and Demetri Terzopoulos. 2018. Deep Learning of Biomimetic Sensorimotor Control for Biomechanical Human Animation. *ACM Trans. Graph.* 37, 4, Article 56 (jul 2018), 15 pages. https://doi.org/10.1145/3197517.3201305

[52] Antti Oulasvirta, Jussi Jokinen, and Andrew Howes. 2022. Computational Rationality as a Theory of Interaction. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 1–14.

[53] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. DeepMimic: Example-guided Deep Reinforcement Learning of Physics-based Character Skills. *ACM Trans. Graph.* 37, 4, Article 143 (July 2018), 14 pages. https://doi.org/10.1145/3197517.3201311

[54] Xue Bin Peng, Glen Berseth, Kangkang Yin, and Michiel Van De Panne. 2017. DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning. *ACM Trans. Graph.* 36, 4, Article 41 (July 2017), 13 pages. https://doi.org/10.1145/3072959.3073602

[55] E Pennestri, R Stefanelli, PP Valentini, and L Vita. 2007. Virtual musculo-skeletal model for the biomechanical analysis of the upper limb. *Journal of biomechanics* 40, 6 (2007), 1350–1361.

[56] Philip Quinn and Shumin Zhai. 2018. Modeling Gesture-Typing Movements. *Human–Computer Interaction* 33, 3 (2018), 234–280. https://doi.org/10.1080/07370024.2016.1215922 arXiv:https://doi.org/10.1080/07370024.2016.1215922

[57] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research* 22, 268 (2021), 1–8. http://jmlr.org/papers/v22/20-1364.html

[58] J.B. Rawlings, D.Q. Mayne, and M.M. Diehl. 2017. *Model Predictive Control: Theory and Design* (2nd ed.). Nob Hill Publishing.

[59] Lei Ren, Richard K Jones, and David Howard. 2007. Predictive modelling of human walking over a complete gait cycle. *Journal of biomechanics* 40, 7 (2007), 1567–1574.

[60] Shaghayegh Roohi, Christian Guckelsberger, Asko Relas, Henri Heiskanen, Jari Takatalo, and Perttu Hämäläinen. 2021. Predicting Game Difficulty and Engagement Using AI Players. *Proceedings of the ACM on Human-Computer Interaction* 5, CHI PLAY (2021), 1–17.

[61] Shaghayegh Roohi, Asko Relas, Jari Takatalo, Henri Heiskanen, and Perttu Hämäläinen. 2020. Predicting game difficulty and churn without players. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*. 585–593.

[62] Prashant Sachdeva, Shinjiro Sueda, Susanne Bradley, Mikhail Fain, and Dinesh K Pai. 2015. Biomechanical simulation and control of hands and tendinous systems. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–10.

[63] Katherine R. Saul, Xiao Hu, Craig M. Goehler, Meghan E. Vidt, Melissa Daly, Anca Velisar, and Wendy M. Murray. 2014. Benchmarking of dynamic simulation predictions in two software platforms using an upper limb musculoskeletal model. *Computer methods in biomechanics and biomedical engineering* 5842, May 2016 (2014), 1–14. https://doi.org/10.1080/10255842.2014.916698

[64] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017). http://dblp.uni-trier.de/db/journals/corr/corr1707.html#SchulmanWDRK17

[65] Ajay Seth, Michael Sherman, Jeffrey A. Reinbolt, and Scott L. Delp. 2011. OpenSim: a musculoskeletal modeling and simulation framework for in silico investigations and exchange. *Procedia IUTAM* 2 (2011), 212–232. https://doi.org/10.1016/j.piutam.2011.04.021 IUTAM Symposium on Human Body Dynamics.

[66] Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. 2017. World of Bits: An Open-Domain Platform for Web-Based Agents. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 3135–3144. https://proceedings.mlr.press/v70/shi17a.html

[67] Weiguang Si, Sung-Hee Lee, Eftychios Sifakis, and Demetri Terzopoulos. 2014. Realistic biomechanical simulation and control of human swimming. *ACM Transactions on Graphics (TOG)* 34, 1 (2014), 1–15.

[68] Eduardo D Sontag. 2013. *Mathematical control theory: deterministic finite dimensional systems*. Vol. 6. Springer Science & Business Media.

[69] Shinjiro Sueda, Andrew Kaufman, and Dinesh K Pai. 2008. Musculotendon simulation for hand animation. In *ACM SIGGRAPH 2008 papers*. 1–8.

[70] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction* (second ed.). The MIT Press. http://incompleteideas.net/book/the-book-2nd.html

[71] Misaki Takeda, Takanori Sato, Hisashi Saito, Hiroshi Iwasaki, Isao Nambu, and Yasuhiro Wada. 2019. Explanation of Fitts' law in Reaching Movement based on Human Arm Dynamics. *Scientific Reports* 9 (12 2019), 19804. https://doi.org/10.1038/s41598-019-56016-7

[72] Hirokazu Tanaka, John W. Krakauer, and Ning Qian. 2006. An Optimization Principle for Determining Movement Duration. *Journal of Neurophysiology* 95, 6 (2006), 3875–3886. https://doi.org/10.1152/jn.00751.2005 arXiv:https://doi.org/10.1152/jn.00751.2005 PMID: 16571740.

[73] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 5026–5033. https://doi.org/10.1109/IROS.2012.6386109

[74] Jack M Wang, Samuel R Hamner, Scott L Delp, and Vladlen Koltun. 2012. Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–11.

[75] Daan Wierstra, Alexander Foerster, Jan Peters, and Juergen Schmidhuber. 2007. Solving deep memory POMDPs with recurrent policy gradients. In *International conference on artificial neural networks*. Springer, 697–706.

[76] DA Winter. 1984. Biomechanics of human movement with applications to the study of human locomotion. *Critical reviews in biomedical engineering* 9, 4 (1984), 287—314. http://europepmc.org/abstract/MED/6368126

[77] Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2020. A Scalable Approach to Control Diverse Behaviors for Physically Simulated Characters. *ACM Trans. Graph.* 39, 4, Article 33 (2020). https://doi.org/10.1145/3386569.3392381

[78] Mingrui Ray Zhang, Shumin Zhai, and Jacob O Wobbrock. 2019. Text entry throughput: Towards unifying speed and accuracy in a single performance metric. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13.