Robo-Saber: A Full-body Physics-based Virtual Reality Playtesting Agent

NAM HEE KIM, Aalto University, Finland
MAY LIU, University of California, Berkeley, USA
PERTTU HÄMÄLÄINEN, Aalto University, Finland
JAMES F. O'BRIEN, University of California, Berkeley, USA
XUE BIN PENG, Simon Fraser University / NVIDIA, Canada

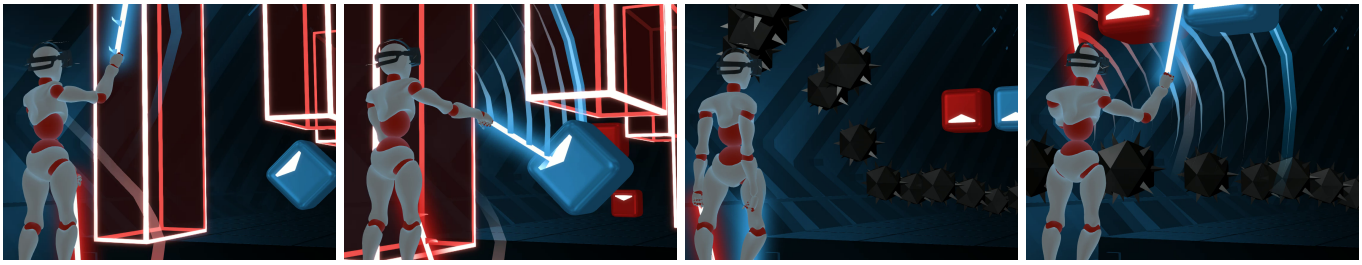

Fig. 1. We present the first full-body physically simulated AI player for VR games, demonstrated in simulated playtesting of *Beat Saber* maps.

We present Robo-Saber, the first physics-based character control system capable of playing *Beat Saber*, a popular VR game that requires complex and spatio-temporally precise full-body movements. Our technical solution combines 1) a custom kinematic generative model for the three-point ($3p$) movements of the VR headset and hand trackers and 2) a $3p$ tracking controller for reconstructing the movements of the player's body in a physically plausible manner. A GPU-accelerated game simulator is incorporated in the inference loop to evaluate and select among multiple candidate $3p$ trajectories, improving generalization. By training our model on the large BOXRR-23 dataset, our system is able to exhibit expert-level gameplay behavior and physically plausible movements that generalize to new game maps. We demonstrate Robo-Saber's utility for user modeling by building an AI-augmented collaborative filtering model for predicting human players' scores on novel *Beat Saber* maps for which no human score data is given. Our results suggest that our framework could assist the designers of novel VR game levels and enhance game curriculum design and personalization for VR experiences beyond *Beat Saber*.

CCS Concepts: • **Computing methodologies** → **Procedural animation**; **Virtual reality**; *Control methods*.

Additional Key Words and Phrases: character animation, virtual reality, reinforcement learning

ACM Reference Format:
Nam Hee Kim, May Liu, Perttu Hämäläinen, James F. O'Brien, and Xue Bin Peng. 2025. Robo-Saber: A Full-body Physics-based Virtual Reality Playtesting Agent. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 14 pages. https://doi.org/XXXXXXX.XXXXXXX

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Simulating human perception and motor control is a common problem in computer animation and robotics that has recently found new applications in *computational user modeling*, e.g., automatic testing of gestural user interfaces and virtual reality (VR) games [Fischer et al. 2024; Ikkala et al. 2022]. The key promise of user modeling is that designers can employ the models instead of (or in addition to) human users to get faster, cheaper, and/or better feedback about their ideas and prototypes. Furthermore, if obtaining such feedback can be automated, designers may also employ computational optimization and design techniques in new domains, the objective functions defined in terms of data produced by the user models.

Until now, user modeling research has tackled problems with limited or no embodiment, e.g., predicting the difficulty of mobile games using an AI player that directly generates touchscreen interaction events without simulating the human hand [Kristensen et al. 2020; Roohi et al. 2021, 2020], or simulating touchscreen typing using a simplified model of finger movements [Shi et al. 2025]. In the rare systems featuring intelligent control of an actual physical or biomechanical simulation model, the focus has been on simulating only a single arm [Cheema et al. 2020; Fischer et al. 2024; Ikkala et al. 2022]. Thus, *building a full-body physically simulated user model for embodied interaction has remained an open challenge*, holding back developers of VR games, for instance.

Towards solving this challenge, we propose and evaluate Robo-Saber, *the first full-body, physically simulated player for a VR game* that requires complex and spatio-temporally precise full-body movements. Our testbed VR game of choice is *Beat Saber* [Beat Games 2019], currently the most popular VR application to date [Wöbbeking 2022]. Due to the popularity of *Beat Saber*, a large dataset comprising human gameplay samples is available: BOXRR-23 [Nair et al. 2023a], making the game a uniquely ideal candidate that allows validating the user modeling results with ground truth human data.

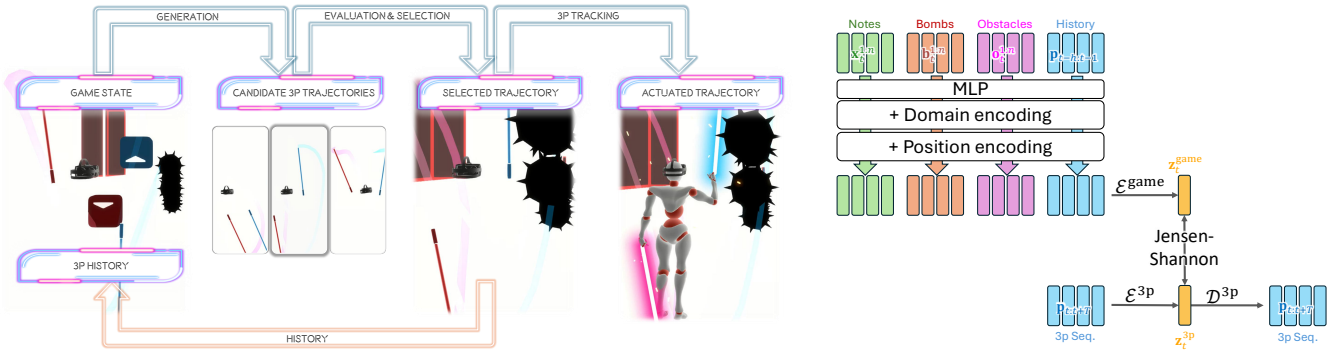

Fig. 2. **Left:** System overview. First, an autoregressive three-point ($3p$) kinematic generative model generates candidate future trajectories for the HMD, LHC, and RHC, conditioned on game state. The best $3p$ trajectory is then selected based on simulating the game forward. Finally, a physics-based $3p$ tracking policy synthesizes full-body movements. **Right:** Our implementation of categorical codebook matching [Starke et al. 2024] features a transformer architecture and a modified loss function based on Jensen-Shannon divergence.

Robo-Saber combines a high-level kinematic planner based on autoregressive motion generation and a low-level physics-based tracking controller, which altogether constitute a full-body, free-standing humanoid control policy that observes game state information and produces game-playing movements. We validate Robo-Saber's output movements by simulating the gameplay and comparing its performance against that of real human players, showing that Robo-Saber generates plausible and skilled game-playing movements. Leveraging Robo-Saber for automated playtesting, we demonstrate the first system for *AI-augmented* personalized score prediction for a VR game, where a player's score for brand-new/never-played maps can be predicted *without involving real human players*.

We summarize our contributions as follows:

- **The first full-body physics-based AI playtesting framework for VR games.** Our framework incorporates kinematic $3p$ motion generation and full-body physics-based tracking. In so doing, we produce full-body gameplay movements that empirically demonstrate expert-level performance.
- **Demonstration of *in silico* VR user modeling application: personalized score prediction.** We apply our framework to enable *AI-augmented* user modeling, demonstrated in our personalized score prediction (PSP) model based on collaborative filtering for *Beat Saber* (Sec. 6 and Fig. 6). The prediction performance is further enhanced by our system's output performance diversity owing to physical constraints and emulated limits to planning ability. Our success with PSP shows promise for downstream applications such as content curation and game curriculum design.
- **Improving generalization with game simulation in sampling.** Akin to the sampling-based model predictive control (MPC) framework (e.g., [Hämäläinen et al. 2014]), our Gumbel-Softmax VAE system samples and selects among multiple candidate $3p$ trajectory samples, enabled by our custom gameplay simulator that leverages a vectorized collision detection algorithm running on GPU (Sec. 4.2). This reward-based candidate trajectory is decoupled from the generative model training, enhancing the flexibility of our framework. We demonstrate that a simple reward-based selection greatly improves generalization (Sec. 5 and Fig. 4).

2 Related Work

Generative models for motion and VR. The availability of annotated human motion data has enabled impressive progress in kinematic motion generation approaches. Previously, conditional generative techniques have been applied to solve controllable and interactive motion synthesis (e.g., [Holden et al. 2017; Ling et al. 2020; Shi et al. 2024; Tevet et al. 2023; Zhang et al. 2018]). A key consumer of motion generation techniques is virtual reality (VR), whose applications are actively being researched and developed. For VR, motion generation often incorporates three-point ($3p$) pose data, as most consumer-grade VR equipment allows tracking the movement of a headset and two handheld controllers. Much of the prior work targets the task of synthesizing full-body pose to be displayed in VR (e.g., [Barquero et al. 2025; Du et al. 2023; Starke et al. 2024; Ye et al. 2022]), which can learn $3p$-to-full-body correspondence from even non-VR-specific datasets. However, as movements in VR games are responses to specific tasks (e.g., swinging arms to combat an enemy), modeling $3p$ movements as intentional behaviors require conditioning on game input. Towards this end, the BOXRR-23 dataset [Nair et al. 2023a] provides $3p$ trajectories for well-known VR games including *Beat Saber*. In this work, we exploit the vast availability of *Beat Saber*-specific $3p$ pose data from BOXRR-23 and an open-source custom map database BeatSaver [2021], aligning game and pose data to formulate a conditional generation task to produce full-body movements from game states. To the best of our knowledge, our work is the first attempt at this conditional generation problem.

Physics-based tracking controllers. Physics-based character animation has become a predominant paradigm for natural and plausible motion synthesis. While difficult to learn from scratch, data-driven methods such as imitation learning (e.g., [Peng et al. 2018, 2022, 2021]) have shown great promise in learning to control humanoids from demonstrations. Using demonstration data also enables tracking controllers for sparse pose inputs (e.g., [Reda et al. 2022; Winkler et al. 2022]), which can synthesize physically based movements from

VR controller motion input. Recent advances have shifted focus toward creating more generalist tracking policies (e.g., [Luo et al. 2023a,b]) for building simulated character controllers that can track a variety of target poses. We build on Perpetual Humanoid Control (PHC, Luo et al. [2023a]) which is trained in an adaptive curriculum over the AMASS [Mahmood et al. 2019] dataset; namely, PHC's 3p tracking module is used to track the generative model's 3p motion output with full-body actuations, such that the physically simulated character performs a corresponding motion. As described in later sections, we adopt PHC while fine-tuning the 3p tracking policy with custom, *Beat Saber*-playing full-body motion capture data.

User modeling. Computational user models have gradually progressed from simple decision making such as puzzle games [Gudmundsson et al. 2018; Roohi et al. 2020] to embodied interactions [Cheema et al. 2020; Ikkala et al. 2022; Jokinen et al. 2021] and modeling the effects of fatigue [Cheema et al. 2020, 2023]. Recently, Fischer et al. [2024] presented Sim2VR, a user model for VR interaction tasks, albeit only focusing on modeling a single arm. We test Robo-Saber in game score prediction, which is closely related to difficulty prediction, a common user modeling task [Gudmundsson et al. 2018; Kristensen et al. 2022; Poromaa 2017; Roohi et al. 2021, 2020]. Kristensen et al. [2022] demonstrated that *personalized* difficulty prediction can be implemented using factorization machines for a given game level and player, based on the performance of other players on the level in question. We extend their approach, utilizing AI players to allow score predictions for levels that no human player has yet tested.

3 Preliminaries

3.1 Beat Saber: a VR Rhythm Game

In *Beat Saber*, the player swings hand-held virtual sabers to cut each *colored note* with the correct hand and in the correct direction, while avoiding *bomb notes* with the sabers and *obstacles* with the head. Often referred to as an exergame, *Beat Saber* has been shown to have health benefits similar to traditional physical exercise [Thai 2021], as well as measurable cognitive benefits [Grosprêtre et al. 2023]. *Beat Saber* has also been proposed as a form of neurological music therapy for Parkinson's disease [Ruhf 2020].

Beat Saber's overwhelming popularity is partly owed to its culture of user-created content, comprising over 100,000 custom *maps* authored by online creators referred to as *mappers* [Nair et al. 2023b]. Mappers have creative freedom to choreograph interesting and complex patterns that encourage a variety of movements from flowing dance-like motions to rapid striking motions. While each map is manually assigned an overall difficulty rating among Easy, Normal, Hard, Expert, or Expert+, in practice, human play is required to assess a map's actual difficulty. This is particularly so for maps intended for advanced players as these maps often creatively break guidelines. Moreover, the experience of difficulty levels is often subjective: a player might find a particular map easier or harder than how an average player would experience it, based on their particular gameplay history and playstyle. These pain points lead to difficulties in curating the vast game content, despite *Beat Saber* being a skill-intensive game that could be served better with recommendations according to a more objective criterion. In this work, we propose the use of a playtesting agent in conjunction with collaborative filtering to perform personalized score prediction, showing promise towards this direction.

3.2 Three-Point (3p) Tracking

A typical consumer-grade VR controller comprises three-point (3p) pose input, i.e., three rigid body entities, each with 6 degrees of freedom, corresponding to the head-mounted display (HMD), the left-hand controller (LHC), and the right-hand controller (RHC). More formally, a 3p motion segment of T frames is denoted as $\mathbf{p}_{t:t+T}$, each $\mathbf{p}_t$ consisting of pose features, i.e., position and orientation, for each of HMD, LHC, and RHC.

Recent work involves the use of a reinforcement learning-based goal-conditioned control policy treating 3p poses as reaching targets (e.g., [Reda et al. 2023; Winkler et al. 2022]). For a physically simulated humanoid agent, the goal-conditioned controller π samples actuations for its full body by taking a target 3p pose $\mathbf{p}_t$ as its goal:

$$\mathbf{a}_t \sim \pi\left(\cdot \mid \mathbf{s}_t, \mathbf{p}_t\right) \tag{1}$$

where $\mathbf{a}_t$ is the action directing the agent's movement and $\mathbf{s}_t$ is the agent's state at timestep t.

3.3 Conditional Autoregressive Motion Generation

Motion generation can be formulated as an autoregressive process, whereby a learned model maps preceding poses to the distribution of future poses, hence modeling valid pose transitions according to the training data. In our case, at each timestep, our objective is to generate a multiple timesteps' worth of pose output, given the information about game objects and the history of pose. Formally, suppose we have h frames for the input *history*: $\mathbf{p}_{t-h:t}$ and T frames for the output *chunk*: $\mathbf{p}_{t:t+T}$. At timestep t, the generative model samples the chunk as

$$\hat{\mathbf{p}}_{t:t+T} \sim p\left(\mathbf{p}_{t:t+T} \mid \mathbf{p}_{t-h:t}, \mathbf{x}_t\right) \tag{2}$$

where $\hat{\mathbf{p}}_{t:t+T}$ denotes the sampled chunk and $\mathbf{x}_t$ denotes the observed game state at timestep t. $\hat{\mathbf{p}}_{t:t+T}$ can then be used to play the game. In autoregressive generation, the last h frames of the generated output are taken as input for the next step's generation.

4 Method

As illustrated in Fig. 2, we combine a kinematic motion generation model and a physics-based 3p tracking controller to implement a physically simulated character capable of playing *Beat Saber* with full-body movements. The kinematic model receives game state features as input, such as incoming colored notes, bombs, and obstacles, and then generates a movement plan for the 3p body parts in an autoregressive manner. Building on [Starke et al. 2024]'s use of a Gumbel-Softmax VAE, the model allows for conditional generation of multiple candidate 3p movement plans. The movement plans are then evaluated via gameplay simulation, after which the plan producing the highest score is selected. The selected output is given to the physics-based controller, which actuates a full-bodied physically-simulated humanoid character to follow the target kinematic plan. The physics-based controller is constructed by fine-tuning the 3p

variant of PHC [Luo et al. 2023a] on a small custom dataset of full-body *Beat Saber* gameplay we captured with an expert player, to augment the 3p motion data available in BOXRR-23.

4.1 Categorical Codebook Matching

As illustrated on the right in Fig. 2, we tackle our conditional 3p motion generation process by producing a latent embedding sequence based on the inputs, and then predicting the logits of the corresponding categorical distribution as the target, as an implementation of categorical codebook matching (CCM, Starke et al. [2024]).

More formally, the 3p movement chunk $\mathbf{p}_{t:t+T}$ of length T is mapped to the logits $\mathbf{z}_t^{\text{3p}}$. The logits are reshaped into that of a joint categorical distribution consisting of C channels of D categories. With Gumbel-Softmax sampling, we produce from this distribution one-hot sequences of length C, each row corresponding to an integer in $\{1, 2, \cdots, D\}$. With straight-through estimation (STE), the one-hot samples retain their gradients through the reparameterized Gumbel sampling procedure [Jang et al. 2017]. The resulting one-hot samples are then decoded back to reconstruct the input 3p movement chunk $\mathbf{p}_{t:t+T}$. Hence, the Gumbel-Softmax VAE's auto-encoding loss is simply the mean squared-error (MSE) loss between the original and reconstructed input, i.e.:

$$\mathcal{L}_{\text{Recon}} = \frac{1}{T}\left\|\mathbf{p}_{t:t+T} - \hat{\mathbf{p}}_{t:t+T}\right\|^2 \tag{3}$$

To match the input control signal–the game observation in our case–to the 3p movement chunks in data, we train a separate transformer encoder block that maps the history of the character's movements and the current game observation to a separate categorical probability distribution. Fig. 2 illustrates how the input sequences of notes, bombs, obstacles, and 3p poses are embedded for the encoder, first using multi-layer perceptrons (MLPs) and then adding domain and positional encodings.

Codebook Matching via Jensen-Shannon Divergence Similar to [Starke et al. 2024], we encourage the similarity of the two categorical distributions $\mathbf{z}_t^{\text{3p}}$ and $\mathbf{z}_t^{\text{game}}$. During inference, this allows generating 3p output sequences conditioned on the game state by connecting the 3p decoder $\mathcal{D}^{\text{3p}}$ to the game state encoder $\mathcal{E}^{\text{game}}$, i.e., omitting the $\mathbf{z}_t^{\text{3p}}$ and sampling the $\mathcal{D}^{\text{3p}}$ input using $\mathbf{z}_t^{\text{game}}$.

While Starke et al. [2024] minimize L2-distances between the $C \times D$ one-hot samples coming from the two distributions, we instead minimize the Jensen-Shannon divergence (JSD) between the distributions. Although this alternative loss introduces a loss weight hyperparameter to training, we note that this approach is more grounded in principle for matching two categorical distributions. While using the L2-distances can perform well empirically, we observe that using JSD results in a comparable performance; see Appendix C and Fig. 12 for quantitative comparisons. The JSD-based matching loss is computed according to:

$$\mathcal{L}_{\text{Match}} = D_{\text{KL}}\left(\mathbf{z}_t^{\text{3p}} \middle\| P\right) + D_{\text{KL}}\left(\mathbf{z}_t^{\text{game}} \middle\| P\right) \tag{4}$$

where D_{KL} is Kullback-Leibler divergence between two distributions and $P = \frac{1}{2}\left(\mathbf{z}_t^{\text{3p}} + \mathbf{z}_t^{\text{game}}\right)$. Then, the final loss function is simply the weighted sum of the two loss terms:

$$\mathcal{L} = \mathcal{L}_{\text{Recon}} + \lambda_{\text{Match}} \cdot \mathcal{L}_{\text{Match}} \tag{5}$$

We find λ_{Match} =1e-4 to be effective for our experiments.

Encoder details. The 3p pose input $\mathbf{p} \in \mathbb{R}^{27}$ is represented by the concatenation of the global xyz-coordinate as well as the 6-dimensional orientation [Zhou et al. 2019] of the three body parts (i.e., HMD, LHC, and RHC). The features for each colored note $\mathbf{c}$ are simply the integer values corresponding to the position indices, note colors, and cut direction (following the *Beat Saber* Modding Group–BSMG–format), paired with the float-valued TTA in seconds, i.e., the difference between the current timestamp and the absolute timestamp where the note appears in the map. The absolute timestamp is computed using the accompanying soundtrack's tempo in beats per minute and the note's beat index. Each bomb $\mathbf{b}$ and obstacle $\mathbf{o}$ similarly represented by the position indices and its TTA following the BSMG format. Each sequence can contain up to n notes over the upcoming s seconds of *lookahead*. The colored notes $\mathbf{c}_t^{1:n}$, the bombs $\mathbf{b}_t^{1:n}$, and the obstacles $\mathbf{o}_t^{1:n}$, as well as the history of previous h generated poses $\mathbf{p}_{t-h:t-1}$, are projected to the d-dimensional input space $\mathbb{R}^d$ for the transformer to produce a latent sequence of $3n + h$ vectors. The note/obstacle sequences are sorted by their order of appearance in game, and each vector receives a positional encoding based on the number of seconds until the note/obstacle arrives at the origin (time-to-arrival, or *TTA*). The pose history receives positional encoding based on its index. We use sinusoidal position encoding for both. Additionally, to help separate notes, bombs, obstacles, and history from one another, we add another layer of higher-frequency sinusoidal encoding, which we refer to as *domain encoding*.

While the input latent sequence always has $3n + h$ vectors, the attention weights are activated only for the vectors corresponding to the objects that appear within the model's lookahead s, i.e., the TTA in seconds at which the object "appears" in the model's purview. In Sec. 6, the lookahead is manipulated to simulate players with different motion planning and anticipation capabilities. Unused vectors are masked with zero.

4.2 Candidate Trajectory Selection

A key benefit of the CVAE framework is the inference-time sampling capability; for each conditional input, the user of the VAE can sample from the corresponding latent distribution to produce multiple viable candidate outputs. Moreover, having encoded long-horizon movement plans, the candidate outputs can be evaluated in a non-greedy, planning-like manner. For example. in [Starke et al. 2024], the candidate movement chunks corresponding to an input signal were evaluated and then used in a receding horizon fashion; only the first of the T frames of final movement output was used before autoregressively computing from the next input in sequence. In our case, we leverage this by evaluating a large number of candidate 3p movement chunks by simulating the game forward for each chunk. Given the predicted logits $\mathbf{z}_t^{\text{3p}}$, we sample N_{traj} *candidate* 3p trajectories:

$$\left(\hat{\mathbf{p}}_{t:t+T}{}^{i}\right)_{i=1,2,\cdots,N_{\text{traj}}} \sim \mathcal{D}\left(\text{GumbelSoftmax}\left(\mathbf{z}_t^{\text{3p}}\right)\right) \tag{6}$$

The candidate trajectories are evaluated based on the input game observation, which contain sufficient information for simulating the game forward for T frames. The highest-scoring trajectory ${\hat{\mathbf{p}}_{t:t+T}}^{i^*}$ is then chosen as the final output, where:

$$i^* = \underset{i=1,2,\cdots,N_{\text{traj}}}{\arg\max} \ \text{Evaluate}\left({\hat{\mathbf{p}}_{t:t+T}}^{i}, \mathbf{c}_t^{1:n}, \mathbf{b}_t^{1:n}, \mathbf{o}_t^{1:n}\right) \tag{7}$$

Unlike [Starke et al. 2024], we use the entire T-frame prediction without intermediate re-planning; we find the motion quality for the $3p$ trajectories to be better overall for tracking when the output chunks are coherent and continuous, albeit possibly suboptimal in terms of gameplay.

TorchSaber: A simplified, GPU-accelerated Beat Saber simulator. As the real game remains closed source and thus difficult to interface with, the evaluation routine in Eq. 7 utilizes a custom simplified *Beat Saber* gameplay simulator we call *TorchSaber (TS)*. TS implements vectorized collision detection and distance computation on PyTorch, so that all vector operations leverage massively parallel GPU compute. With TS, we can simulate and evaluate candidate trajectories, guiding the $3p$ motion generation towards better gameplay and more continuous output. Moreover, we utilize TS as a simplified metric for user modeling experiments, as Robo-Saber's performance can be readily evaluated and compared to that of real human players. As *Beat Saber*'s scoring criteria are intricate, we simply compute the swing angle score for every "good cut", i.e., collision events with correct directions and colors. The maximum score of 1 per colored note is achieved if the correct saber's orientations 24 frames previous to and after the good cut are at least -100 degrees and 60 degrees, respectively. We compute the average swing angle score and label it the TS score to serve as a heuristic for gameplay performance for user modeling experiments. For our validation set comprising 4 thousand samples, TS score evaluated on real player $3p$ movement data exhibit a moderate Pearson correlation of 0.710 with respect to the ground truth game score values.

Reward function. Based on TorchSaber's simulation results, we assign a reward value r_i to each candidate output ${\hat{\mathbf{p}}_{t:t+T}}^{i}$, as defined below; Appendix B discusses the reward terms in more detail.

$$r_i = r_{\text{TS}} - \lambda_{\text{Bomb}} \cdot r_{\text{Bomb}} + \lambda_{\text{Obstacle}} \cdot r_{\text{Obstacle}} \tag{8}$$

- $r_{\text{TS}} \in [0, 1]$ is the TorchSaber score as defined above, calculated for the generated poses and colored notes within range.
- $r_{\text{Bomb}} \in [0, 1]$ is the bomb collision penalty, computed as the ratio between the numbers of bomb hits and appearing bombs.
- $r_{\text{Obstacle}} \in [0, 1]$ is the obstacle distance bonus, computed as the minimum distance between any appearing obstacles' yz-bounds and the head yz-positions. When the head collides with an obstacle, the value is set to 0.
- The λ_* values are weights corresponding to each term; see Table 1 for the values used in our experiments.

4.3 Tracking Controller

We build on a popular tracking controller for a physically simulated humanoid from Perpetual Humanoid Control (PHC, Luo et al. [2023a]). Built within Isaac Gym [Makoviychuk et al. 2021], PHC produces full-body joint actuations that lead the robot's head and hands to be close in both position and rotation to the input $3p$ pose. We first tested the $3p$ tracking variant of PHC [Luo et al. 2023a] without modification. We found it to be unable to maintain the character's balance with *Beat Saber*-specific movements, and so we fine-tuned the model on custom mocap sequences. An experienced *Beat Saber* player performed 16 play sequences in total while wearing an inertial mocap suit together with a VR headset and controllers, covering Normal, Hard, Expert, and Expert+ difficulty levels. For the PHC finetuning, we produced a 50-50 mixture of old training data and new training data to prevent the pretrained controller from forgetting its existing tracking abilities.

Taking full-body reference motions as training input, the $3p$ tracker is trained to jointly optimize the tracking and imitation rewards: the former encourages the physically resulting $3p$ poses to match those of the reference, while the latter encourages the agent to produce full-body motions similar to those in the training data. The fidelity of $3p$ tracking and the human-likeness of the movement are at a tradeoff; as our system aims to produce valid gameplay movements foremost, we use the default reward weights, which prioritize the $3p$ tracking performance.

4.4 Data and Implementation Details

We use *Beat Saber*-specific $3p$ replay data from BOXRR-23 [Nair et al. 2023a]. The $3p$ replay data is paired with the corresponding map's BSMG data [BSMG 2019] downloaded from the open-source database BeatSaver [2021]. As a quality control measure, we remove every data point that is missing BSMG entry or score record from consideration. As *Beat Saber* features many *mods* and *game modes*, which change the behavior of the game significantly, we simplify our task by filtering out data points that have mods or non-standard game modes. Additionally, corrupt data points are removed, after which 3, 079, 180 replay sequences remain. We normalize all players' $3p$ xyz-coordinates to match the height of the PHC character, which is $1.5044m$ tall. The sabers are attached to the PHC character's hands, aligned with the direction of the fingers. We use transformer-based architecture for the auto-encoder. Instead of processing the entire T-frame chunks at once, we subsample the frames at a stride to reduce from 60 frames per second (FPS) to 15, following conventional practice in human motion generation. At inference time, the keyframes are predicted and then expanded back to 60 FPS using linear (and spherical) interpolation.

5 Experiment 1: Game Performance

Qualitative evaluation. As shown in the supplementary video, Robo-Saber generates plausible behavior when deployed on new songs not included in the training dataset. Figs. 7–9 show image sequences demonstrating how Robo-Saber moves in response to the observed colored notes, bombs, and obstacles. As our system leverages Gumbel-Softmax VAE that encodes long-horizon motion sequences, a diverse set of trajectories can be generated as a response to the same observation (Fig. 10). Notably, movements involving large full-body motions emerge in situations that require drastic ducking and squatting, but otherwise the physically simulated player displays efficient side-swaying movements with the lower-body mainly balancing upright, reflecting what a real expert player would do.

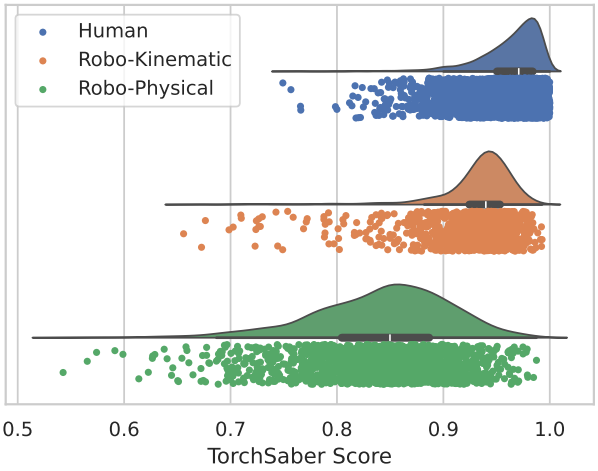

Fig. 3. Comparison of Robo-Saber to human players on held-out maps. The kinematic 3*p* trajectories score similar to humans, while the physical trajectories score lower as expected. As explained in Sec. 6, the physical limitations are actually beneficial for user modeling.

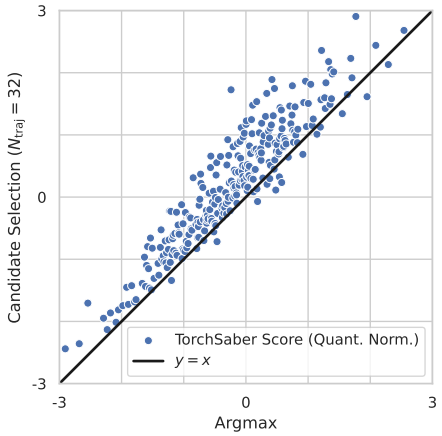

Fig. 4. Scatterplot showing that candidate trajectory selection (y-axis) improves generalization compared to using deterministic, argmax (x-axis) selection for Gumbel-Softmax VAE. Each point compares the two methods for inference for a same unseen map.

Quantitative evaluation. To evaluate Robo-Saboer's playing capability quantitatively, we use TS scores, normalized so that the maximum score for each map is 1. For validation data, we hold out the top 300 most played maps, as well as maps that share a large pool of players (100 total) among BOXRR-23, resulting in a validation set of 400 maps. We compare Robo-Saber's performance with the real players of these maps. For this, the default lookahead of $s = 2$ seconds and number of candidate trajectories of $N_{\text{traj}} = 64$ are used. For difficult or unorthodox gameplay sequences requiring unique movements, the physics-based agent may accrue significant tracking error, sometimes to the extent that it falls to the ground irrecoverably. To ensure that evaluation is not affected by such irrecoverable failures, we implement a simulation reset when falling is detected.

Kinematic-only vs. physically based play. As illustrated in Fig. 3, our kinematic 3*p* generator produces scores similar to or slightly below the real human play sequences on the validation set, suggesting that the model is capable of high performance gameplay. Here, one should note that with BOXRR-23 data, *we are comparing against expert rather than average human performance*. The human scores are heavily skewed towards the higher end, due to a number of factors: poor plays typically result in failing the level with no scores reported, players generally select songs they are able to perform well on, experienced players are more likely to install the extension that submits scores online and makes them available to BOXRR-23, and the recorded score for each player is their best reported score for that map. Unlike human scores, Robo-Saber's scores are *zero-shot*, as the agent does not refine its gameplay on the validation maps.

As expected, Fig. 3 shows that introducing physical constraints results in lower scores, as tracking with the full body would degrade and alter any physically infeasible 3*p* movements. As discussed in Sec. 6, this physically-based degredation provides highly useful signals for user modeling. The gameplay performance is still overall respectable, as shown in the supplementary video.

Candidate trajectory selection improves generalization. Selecting the most likely trajectory at each step (i.e., using *argmax* actions) may lead to suboptimal behaviors, as the model may not always produce correct logits at every timestep. Our system instead generates N_{traj} samples and selects the best after evaluation, akin to black-box model predictive control (MPC) frameworks, which leverage a simulator for producing and pruning candidate motion plans before committing to one. In Fig. 4, we observe that using rollouts with candidate trajectory selection (y-axis) reliably outperforms using argmax rollouts (x-axis), showing better generalization with sampling and evaluation via gameplay simulation. This is consistent with the findings of Starke et al. [2024] who also highlighted the effect of inference-time candidate selection–their use case, however, deals with the continuity of the generated full-body motion, while ours optimizes gameplay performance.

6 Experiment 2: Personalized Score Prediction

Beat Saber suffers from poor difficulty labeling and the lack of personalized curation despite an overwhelming amount of content. Towards this challenge, we demonstrate how automated playtesting with Robo-Saber might help develop a recommender system by solving the *personalized score prediction* (PSP) problem, i.e., predicting a given map's score for a given human player.

Method. Following Kristensen et al. [2022], we employ factorization machines (FMs, Rendle [2010]) for PSP. However, while Kristensen et al. [2022] only used human player data, we augment the FM training data with Robo-Saber scores. This allows *personalized score predictions for novel maps* for which no human scores are available.

FMs learn embedding vectors for each player and map such that the dot product of the player and map embeddings predicts the

score for the player-map pair. Importantly, *FM does not require exactly human-like data from Robo-Saber*–it only assumes that the differences between Robo-Saber players at least partially model the differences between human players. To ensure this, we utilize a *diverse population of different Robo-Saber variants*, created by varying the following:

- Lookahead time: Being able to anticipate and plan movements in advance is a key aspect of human motor performance. We artificially manipulate this capability by limiting how far into the future the model can perceive the upcoming objects. We use lookahead times 2.0, 1.5, 1.0, and 0.5 seconds; see Fig. 11 for visualizations of the diversity of trajectories resulting from this. Note that we only train Robo-Saber once and manipulate the lookahead time during inference.
- Physicality: For each lookahead time, we include a Robo-Saber player both with and without the full-body physics simulation. The full-body physics simulation is also done with 2 variations of hand masses, as to emulate movement skill differences (heavier hands make fast movements more difficult). For this, we multiply PHC's default hand mass by 20. We use the corresponding trajectories of the hands and head of the full-body character for computing TS scores.

Various FM variants and extensions have been proposed; we utilize Deep Factorization Machines (DFMs, Guo et al. [2017]) from PyTorch-FM, an open-source reference implementation. We use a simple MLP architecture involving 2 hidden layers of 8 neurons, a dropout rate of 0.50, and embedding size 16.

Data. We train the DFM with the validation set of 341 maps. We only include the human scores for the subset of 100 maps with many human players. For the remaining 241 maps, the DFM training only utilizes the scores of the Robo-Saber variants, and all our results are reported on these maps. Given the aforementioned skew in the data, we employ quantile normalization [Amaratunga and Cabrera 2001; Bolstad et al. 2003] to transform the score distribution into the standard normal, a technique often used in case of unevenly distributed data [Stevens et al. 2002].

Results: Emulated Diversity Improves PSP. We evaluate the PSP success in terms of mean squared error (MSE) across the 241 maps for which the DFM training utilized only the Robo-Saber scores. Fig. 6 illustrates that *adding each source of diversity improves the MSE*. The most diverse configuration, comprising a population of 4 kinematic players and 8 physics-based players, each playing the maps with 5 random seeds to mitigate Robo-Saber's inherent stochasticity, has a drastically lower MSE than the baseline of using a single kinematic player. This allows us to conclude that:

(1) Even our simple emulation of player diversity partially represents the individual differences of real players, and the DFM is able to utilize this to improve the personalized predictions.
(2) Our physically-based full-body 3p-tracking controller is beneficial for user modeling, even though it could be considered detrimental from the point of view of simply maximizing Robo-Saber's playing skill.

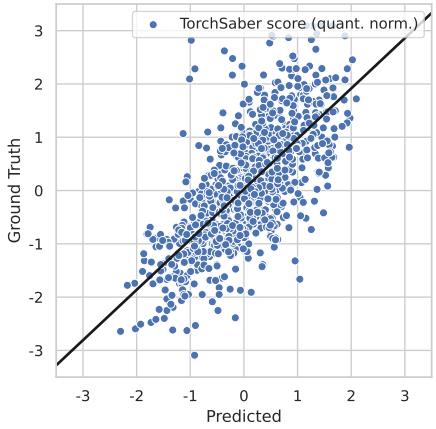

Fig. 5. Scatterplot and correlation of predicted and ground truth quantile-normalized TS scores, using the full DFM configuration. Pearson's $r = 0.702$.

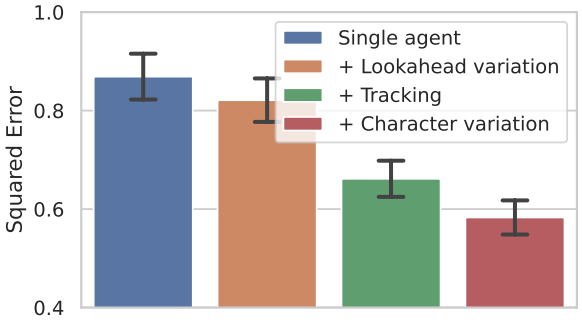

Fig. 6. The effect of behavior diversity among simulated player population on the DFM score prediction error. The increased diversity of the population including full-body physics simulation and its character parameter variation provides better predictions. The bar heights and error caps denote error means and standard errors of the means, respectively.

Fig. 5 visualizes the strong correlation (0.702) of the DFM predictions and ground truth human player scores using the maximally diverse configuration.

7 Conclusion

We have presented Robo-Saber, a fully embodied, physically simulated VR user model that exhibits realistic and skilled Beat Saber gameplay behavior and can be applied to personalized prediction of the scores of novel maps. Our results suggest that *Beat Saber* map designers could use Robo-Saber at least for initial playtesting to predict and visualize player movements and scores. Given suitable 3p example data and a modest amount of full-body motion capture examples, our approach should also generalize to VR scenarios beyond *Beat Saber*, should appropriate data become available.

Limitations and Future Work. In potential follow-up work, we propose to use quantities from physics simulation to improve the modeling of player diversity and estimating variables beyond game score. For instance, the cumulative fatigue modeling of Cheema et al. [2020, 2023] might be useful for VR exergames such as *Beat Saber*. A simulation model with sufficiently realistic anatomy could be used for evaluating the safety of evoked player movements. We are also interested in modeling variations in body proportions, weight, and strength, which our current physics-based controller cannot handle.

While Robo-Saber enables completely automated playtesting, its performance is not calibrated to match the human player population's characteristics. However, this limitation is mitigated by the artificial diversity of the agents we induce (e.g., with and without physics) combined with the DFM-based approach.

References

Dhammika Amaratunga and Javier Cabrera. 2001. Analysis of data from viral DNA microchips. *J. Amer. Statist. Assoc.* 96, 456 (2001), 1161–1170.

German Barquero, Nadine Bertsch, Manojkumar Marramreddy, Carlos Chacón, Filippo Arcadu, Ferran Rigual, Nicky Sijia He, Cristina Palmero, Sergio Escalera, Yuting Ye, et al. 2025. From Sparse Signal to Smooth Motion: Real-Time Motion Generation with Rolling Prediction Models. *arXiv preprint arXiv:2504.05265* (2025).

Beat Saber Modding Group. 2019. Beat Saber Modding Group Wiki. Web page. https://bsmg.wiki/

Benjamin M Bolstad, Rafael A Irizarry, Magnus Åstrand, and Terence P. Speed. 2003. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics* 19, 2 (2003), 185–193.

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. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

Noshaba Cheema, Rui Xu, Nam Hee Kim, Perttu Hämäläinen, Vladislav Golyanik, Marc Habermann, Christian Theobalt, and Philipp Slusallek. 2023. Discovering Fatigued Movements for Virtual Character Animation. In *SIGGRAPH Asia 2023 Conference Papers*. 1–12.

Atticus Cull and Vivek Nair. 2023. SimSaber: Python-based Beat Saber replay simulator and scoring validator. Software. https://github.com/MetaGuard/SimSaber

Yuming Du, Robin Kips, Albert Pumarola, Sebastian Starke, Ali Thabet, and Artsiom Sanakoyeu. 2023. Avatars grow legs: Generating smooth human motion from sparse tracking inputs with diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 481–490.

Florian Fischer, Aleksi Ikkala, Markus Klar, Arthur Fleig, Miroslav Bachinski, Roderick Murray-Smith, Perttu Hämäläinen, Antti Oulasvirta, and Jörg Müller. 2024. SIM2VR: Towards Automated Biomechanical Testing in VR. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 1–15.

Beat Games. 2019. Beat Saber. Video game.

Sidney Grosprêtre, Philémon Marcel-Millet, Pauline Eon, and Bettina Wollesen. 2023. How Exergaming with Virtual Reality Enhances Specific Cognitive and Visuo-Motor Abilities: An Explorative Study. (April 2023), e13278. https://doi.org/10.1111/cogs.13278

Stefan Freyr Gudmundsson, Philipp Eisen, Erik Poromaa, Alex Nodet, Sami Purmonen, Bartlomiej Kozakowski, Richard Meurling, and Lele Cao. 2018. Human-like playtesting with deep learning. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 1–8.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

Perttu Hämäläinen, Sebastian Eriksson, Esa Tanskanen, Ville Kyrki, and Jaakko Lehtinen. 2014. Online motion synthesis using sequential monte carlo. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–12.

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–13.

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 *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–14.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rkE3y85ee

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.

Jeppe Theiss Kristensen, Christian Guckelsberger, Paolo Burelli, and Perttu Hämäläinen. 2022. Personalized game difficulty prediction using factorization machines. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–13.

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.

Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel Van De Panne. 2020. Character controllers using motion vaes. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 40–1.

Zhengyi Luo, Jinkun Cao, Kris Kitani, Weipeng Xu, et al. 2023a. Perpetual humanoid control for real-time simulated avatars. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 10895–10904.

Zhengyi Luo, Jinkun Cao, Josh Merel, Alexander Winkler, Jing Huang, Kris Kitani, and Weipeng Xu. 2023b. Universal humanoid motion representations for physics-based control. *arXiv preprint arXiv:2310.04582* (2023).

Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. 2019. AMASS: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF international conference on computer vision*. 5442–5451.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. 2021. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470* (2021).

Vivek Nair, Wenbo Guo, Rui Wang, James F O'Brien, Louis Rosenberg, and Dawn Song. 2023a. Berkeley open extended reality recordings 2023 (boxrr-23): 4.7 million motion capture recordings from 105,852 extended reality device users. *arXiv preprint arXiv:2310.00430* (2023).

Vivek Nair, Viktor Radulov, and James F. O'Brien. 2023b. Results of the 2023 Census of Beat Saber Users: Virtual Reality Gaming Population Insights and Factors Affecting Virtual Reality E-Sports Performance. (May 2023), 1–19. https://doi.org/10.48550/arXiv.2305.14320

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 Transactions On Graphics (TOG)* 37, 4 (2018), 1–14.

Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. 2022. Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Transactions On Graphics (TOG)* 41, 4 (2022), 1–17.

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. 2021. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (ToG)* 40, 4 (2021), 1–20.

Erik Ragnar Poromaa. 2017. Crushing candy crush: predicting human success rate in a mobile game using Monte-Carlo tree search. https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1093469&dswid=2874

Daniele Reda, Hung Yu Ling, and Michiel Van De Panne. 2022. Learning to brachiate via simplified model imitation. In *ACM SIGGRAPH 2022 Conference Proceedings*. 1–9.

Daniele Reda, Jungdam Won, Yuting Ye, Michiel van de Panne, and Alexander Winkler. 2023. Physics-based Motion Retargeting from Sparse Inputs. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6, 3 (2023), 1–19.

Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J Guibas. 2021. Humor: 3d human motion model for robust pose estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*. 11488–11499.

Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International conference on data mining*. IEEE, 995–1000.

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.

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.

Kaitlyn Danielle Ruhf. 2020. *Physically Active Virtual Reality and Parkinson's Disease: A Pilot Study*. Ph. D. Dissertation. Wake Forest University.

Danqing Shi, Yujun Zhu, Francisco Erivaldo Fernandes Junior, Shumin Zhai, and Antti Oulasvirta. 2025. Simulating Errors in Touchscreen Typing. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–13.

Yi Shi, Jingbo Wang, Xuekun Jiang, Bingkun Lin, Bo Dai, and Xue Bin Peng. 2024. Interactive character control with auto-regressive motion diffusion models. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–14.

Sebastian Starke, Paul Starke, Nicky He, Taku Komura, and Yuting Ye. 2024. Categorical Codebook Matching for Embodied Character Controllers. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–14.

James Stevens et al. 2002. *Applied multivariate statistics for the social sciences*. Vol. 4. Lawrence Erlbaum Associates Mahwah, NJ.

BeatSaver Team. 2021. BeatSaver. Web page. https://beatsaver.com/

Guy Tevet, Sigal Raab, Brian Gordon, Yoni Shafir, Daniel Cohen-or, and Amit Haim Bermano. 2023. Human Motion Diffusion Model. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=SJ1kSyO2jwu

Tuong Thai. 2021. *The Influence Of Exergaming On Heart Rate, Perceived Exertion, Motivation To Exercise, And Time Spent Exercising*. Ph. D. Dissertation. Salem University.

Alexander Winkler, Jungdam Won, and Yuting Ye. 2022. Questsim: Human motion tracking from sparse sensors with simulated avatars. In *SIGGRAPH Asia 2022 Conference Papers*. 1–8.

Jan Wöbbeking. 2022. Beat saber generated more revenue in 2021 than the next five biggest apps combined. https://mixed-news.com/en/beat-saber-generated-more-revenue-in-2021-than-the-next-five-biggest-apps-combined/

Yongjing Ye, Libin Liu, Lei Hu, and Shihong Xia. 2022. Neural3Points: Learning to Generate Physically Realistic Full-body Motion for Virtual Reality Users. In *Computer Graphics Forum*, Vol. 41. Wiley Online Library, 183–194.

He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. 2018. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–11.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. 2019. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5745–5753.

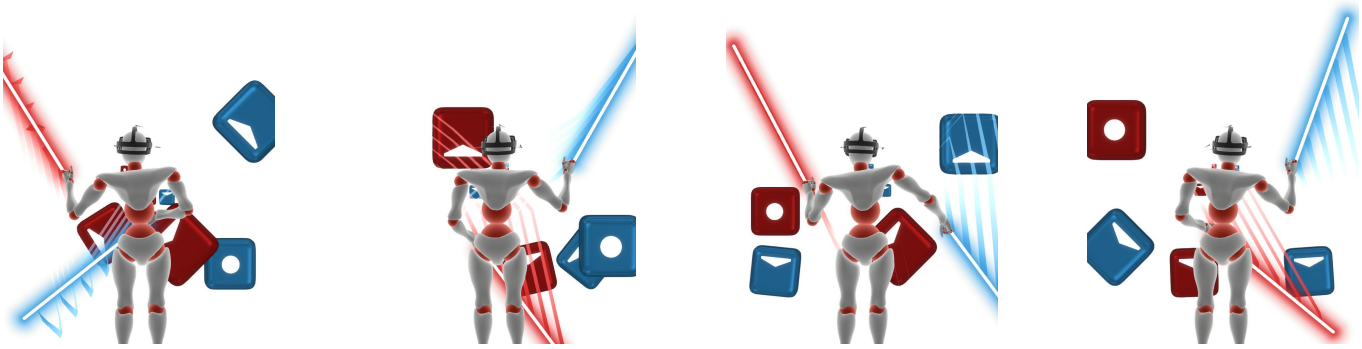

Fig. 7. Robo-Saber plays colored notes. The player is rewarded if either saber cuts a note of the same color in the specified direction. The dotted note (2nd frame) can be hit from any direction.

Fig. 8. Robo-Saber avoids bomb notes. The player is penalized if either of the sabers collides with any of the bomb notes.

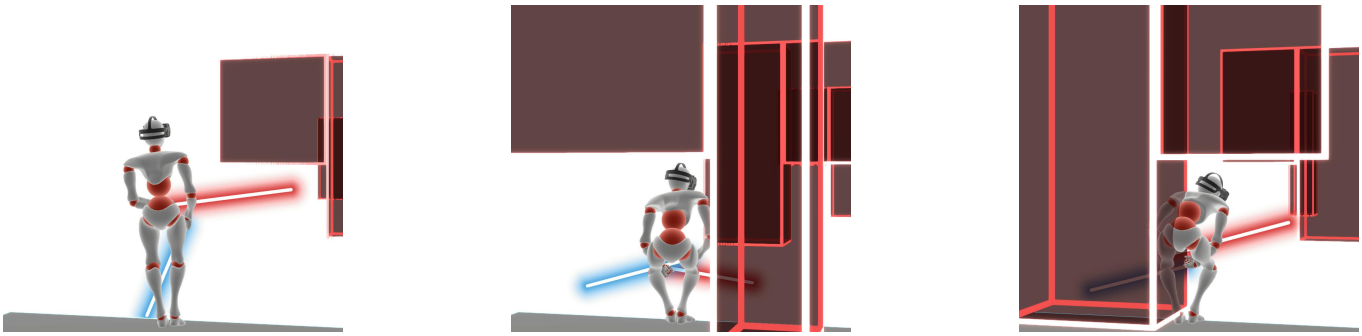

Fig. 9. Robo-Saber performs obstacle avoidance. The player is penalized if the head collides with the obstacle (red boxes).

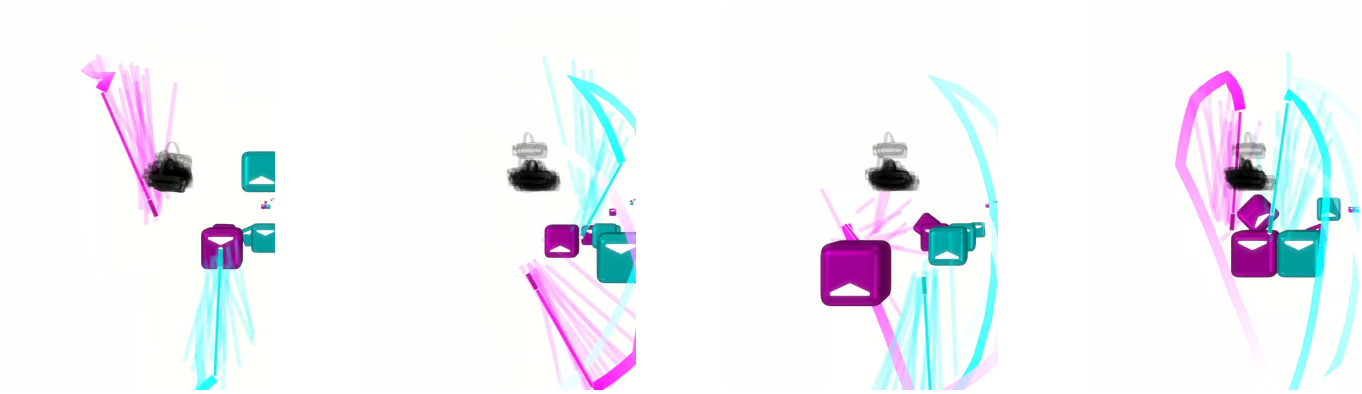

Fig. 10. For the same game input, Robo-Saber's kinematic $3p$ generator samples viable $3p$ trajectory samples (shown as semi-transparent headset and sabers) for the same game input, from which the most optimal one is selected.

Fig. 11. Robo-Saber is deployed with varying **(top)** lookahead seconds (2.0, 1.5, 1.0, 0.5) and **(bottom)** 5 random seeds. Variations in, e.g., saber positioning, playing precision, and idling behaviors can be seen.

A TorchSaber: Implementation Details

TorchSaber is based on a vectorized implementation of collision detection and distance computation. As the name suggests, TorchSaber utilizes PyTorch for GPU-accelerated tensor operations. We use the slab method to detect the intersection between axis-aligned bounding boxes (AABBs) and line segments. We first prepare regularly-spaced keypoints along each saber. Given a gameplay sequence, for each pair of two consecutive frames, we produce line segments by connecting the earlier and later positions of saber keypoints.

In TorchSaber, notes, bombs, and obstacles are modeled as boxes. For notes, we following *Beat Saber*'s collider specification as described in SimSaber [Cull and Nair 2023]. The boxes for the obstacles are scaled according to the duration, width, and height specified in the map data. The sabers are modeled as a 1.2 meters-long line segment. The objects are instantiated in a 3D scene, where each object's x-position (forward) is calculated according to its TTA. The y- and z-position (up) of each object is calculated based on a 4×3 grid. The grid's z-offset is computed based on the player's height. We find $1.05 - \frac{h}{2}$ to be an appropriate offset. We configure the x-offsets of the objects such that they arrive slightly in front of the player. We initially configured the width and height of the grid according to SimSaber's documentation. Then, we fine-tuned these dimensions, as well as the objects' x-offset, in the direction that maximizes the replay scores of real players.

TorchSaber's score calculation is based on the note-saber, bomb-saber, and head-obstacle collision flags. For note-saber collisions, cut directions, cut velocities, and color correctness are taken into account to determine whether the cut was good. The cut direction is declared good if the dot product between the normalized saber tip velocity and the unit vector pointing to the note direction is greater than 0. For the dotted notes, any cut is declared good. Each good cut receives a score between 0 and 1, based on the "swing score" (Appendix B), similar to *Beat Saber*'s game mechanics. For the sake of simplicity and ease of implementation, we excluded combo multipliers from consideration, which still retained a strong positive correlation between TorchSaber's scores and real *Beat Saber* scores from BOXRR-23.

B Reward Function Details

Here, we define the reward terms that appear in Sec. 4.2. The TorchSaber score r_{TS} is as summarily described in Appendix A: for each collision with correct color and direction, we compute the "swing score" based on the 24 frames before and after the collision.

Given the n_{Notes} colored notes that appear in $\mathbf{c}_t^{1:n}$, paired with $\mathbf{p}_{t:t+T}$, we compute the vector $\mathcal{I}_{1:n_{\text{Notes}}} \in \{0,1\}^{n_{\text{Notes}}}$, which is a boolean mask indicating good collision for each colored note. These masks take into account whether a good collision happens before a bad one, as well as whether the note has been collided before in any previous segments considered.

For each colored note $i \in 1, 2, \cdots, n_{\text{Nootes}}$, the TS score is computed as

$$r_{\text{TS}}^i = \mathcal{I}_i \cdot r_{\text{Swing}}^i \tag{9}$$

where r_{Swing}^i is the swing score for note i:

$$r_{\text{Swing}}^i = 0.5 \cdot (r_{\text{Pre}}^i + r_{\text{Post}}^i). \tag{10}$$

Given the frame t^i where the good cut happens, the pre-cut score r_{Pre}^i and the post-cut score r_{Post}^i are defined as below:

$$r_{\text{Pre}}^i = \text{clip}\left(\max \frac{\theta_{t^i-24:t^i}}{100}, 0, 1\right) \tag{11}$$

$$r_{\text{Post}}^i = \text{clip}\left(\max \frac{\theta_{t^i:t^i+24}}{60}, 0, 1\right) \tag{12}$$

where $\theta_{t^i-24:t^i}$ and $\theta_{t^i:t^i+24}$ denote the angles between note i's forward vector and its corresponding saber, up to 24 frames before and after, respectively. These angles are computed on the plane spanned by the colored note's direction and its forward vector. Then the final TS score r_{TS} is simply the mean across the colored notes, i.e.,

$$r_{\text{TS}} = \underset{i \in \{1,2,\cdots,n_{\text{Notes}}\}}{\text{mean}} r_{\text{TS}}^i \tag{13}$$

For candidate trajectory selection, we compute this score at each generator query step for each candidate. For evaluating the full real and synthetic trajectories, we treat the entire play sequence as a segment to compute r_{TS}.

The collision penalty r_{Bomb} is computed as the ratio between the number of cuts registered between each bomb note and any saber and the number of bombs that appear.

Finally, the head-obstacle distance bonus r_{Obstacle} is computed as the mean perpendicular distance between the yz-position of the head and its nearest yz-bound of the appearing obstacles. The perpendicular distance is computed as the maximum dot product between any normal of the obstacle and the head position in the obstacle's local coordinate system. If the head is determined to be interior to an obstacle, r_{Obstacle} is set to 0.

C Ablation Studies on the Kinematic 3p Generator

Fig. 12 shows learning curves quantifying the kinematic 3p generator's performance with TorchSaber score with respect to 5 held-out maps, evaluated for 5 times.

Effect of Jensen-Shannon divergence Loss. Contrary to [Starke et al. 2024]'s approach of computing matching losses based on the $C \times D$ one-hot sequence samples, we elect to use Jensen-Shannon divergence as the measure of the discrepancy between $\mathbf{z}_t^{\text{3p}}$ and $\mathbf{z}_t^{\text{game}}$. The sample-sample mean squared error (MSE) approach can be viewed as a special case of latent Hausdorff distance loss for assimilating two distributions. However, we test that our Jensen-Shannon divergence (JSD) approach can also be used as a viable alternative grounded in probability. Fig. 12 shows that JSD performs similarly to MSE given the same training budget.

Effect of scheduled sampling. Also deviating from the vanilla recipe in [Starke et al. 2024], we follow MVAE [Ling et al. 2020], HuMoR [Rempe et al. 2021], and A-MDM [Shi et al. 2024] to implement a scheduled sampling strategy to ensure that the autoregressive generative model can familiarize itself with possible modes of error.

As seen in Fig. 12, removing scheduled sampling degrades the learning performance greatly, comporting to the results of many other autoregressive motion generation experiments.

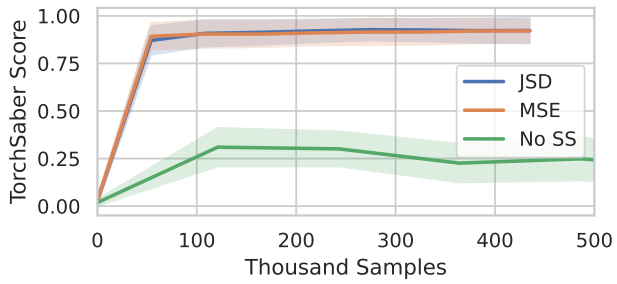

Fig. 12. The effect of ablating major learning components of our system: Jensen-Shannon divergence (JSD) and scheduled sampling (SS), as measured by TorchSaber score for thousands of samples used in training. JSD and MSE are essentially equivalent in empirical performance.

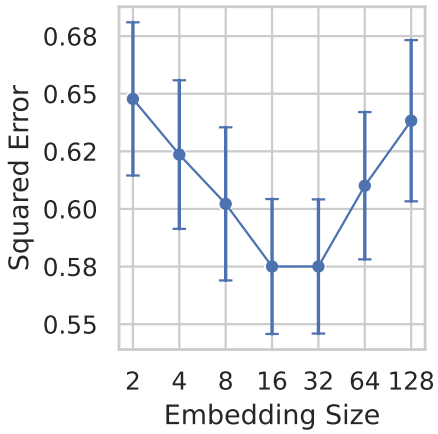

Fig. 13. DFM prediction errors across a range of embedding sizes. We observe that embedding sizes beyond 8 hinders performance due to overfitting. Markers indicate MSEs and error bars indicate standard errors.

Effect of DFM hyperparameters A key DFM hyperparameter is the embedding dimensionality. Fig. 13 shows the personalized prediction error as a function of the embedding dimensionality. With our DFM data including all Robo-Saber variants, increasing the dimensionality beyond 8 yields insignificant gains or even hindrance due to overfitting.

D Table of Hyperparameters

See Table 1.

Table 1. Hyperparameters and their values used for our experiments.

Symbol	Description	Value
T	Number of frames per 3p motion chunk	16
n	Length of latent sequence for each object type	40
s	The default lookahead, in seconds, used in training	2.0
C	The number of channels for categorical codebook matching	128
D	The dimension of each channel for categorical codebook matching	8
h	Number of frames used as 3p history input	2
λ_{Match}	Jensen-Shannon divergence-based matching loss weight	1e-4
λ_{Bomb}	Reward weight for saber-bomb collision penalty	1
$\lambda_{\text{Obstacle}}$	Reward weight for head-obstacle distance penalty	1
-	Interval for interpolating the T frames	4
-	Batch size for DFM training	full batch
-	Learning rate for DFM training	1e-3
-	Optimizer for DFM training	RAdam
-	Number of BOXRR-23 samples in batch for kinematic 3p generator training	64
-	Number of gameplay segments in batch for kinematic 3p generator training	512
-	Learning rate for kinematic 3p generator training	5e-5
-	Number of transformer encoder layers	4
-	Number of attention heads	4
-	Transformer activation function	SiLU