DOI: 10.1111/cgf.70353

EUROGRAPHICS 2026 / B. Masia and J. Thies
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 45 (2026), Number 2

ROBO-SABER: Generating and Simulating Virtual Reality Players

N. H. Kim[1], J. M. Liu[2], J. Lehtinen[1,3], P. Hämäläinen[1], J. F. O'Brien[2], and X. B. Peng[3,4]

[1]Aalto University, Finland [2]University of California, Berkeley, United States [3]NVIDIA [4]Simon Fraser University, Canada

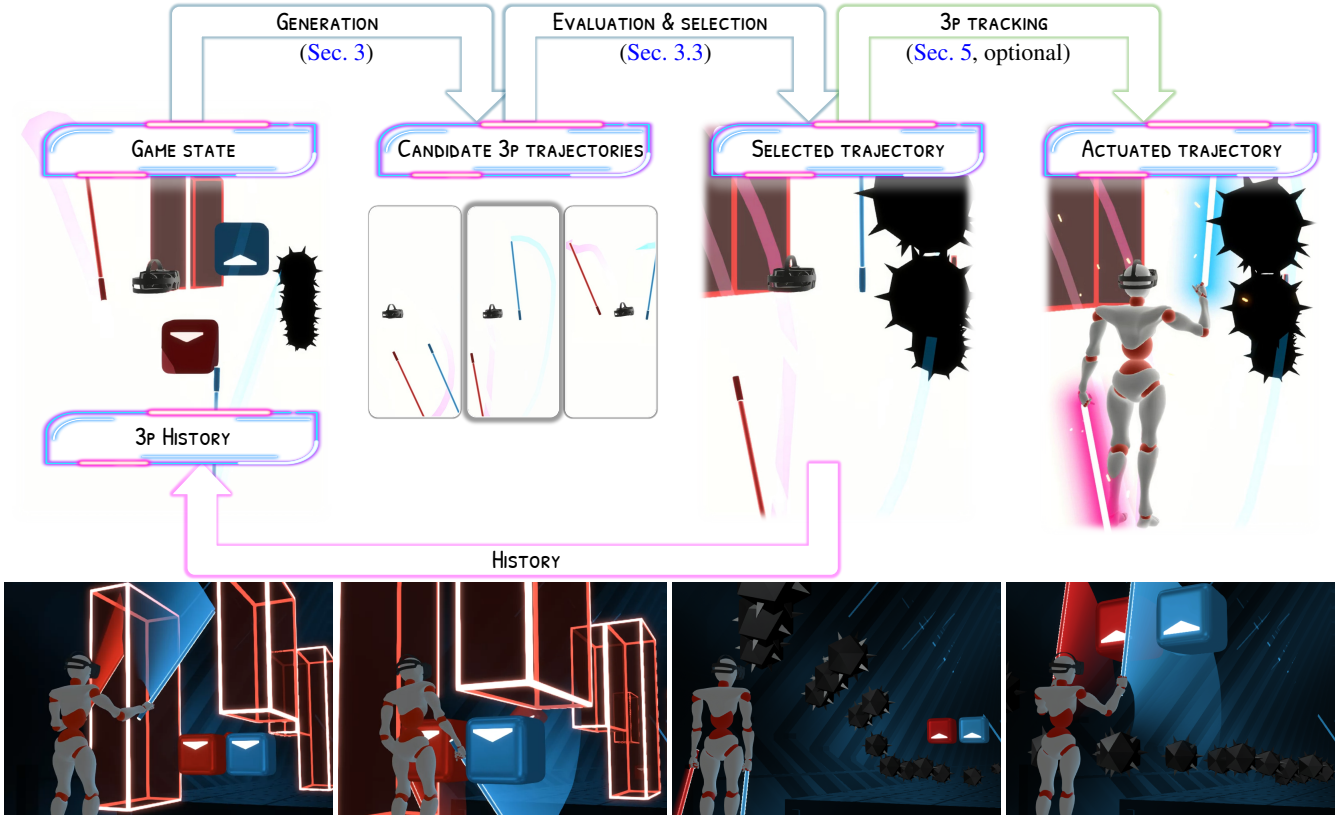

Figure 1: ***Robo-Saber*** *is a player model for VR games, demonstrated in simulated* Beat Saber *playtesting.* ***Top:*** *System overview. An autoregressive generative model samples candidate three-point (3p) trajectories for the headset and handhelds, conditioned on the game state. The best 3p trajectory is then selected based on simulating the game forward. Optionally, a physics-based tracking policy performs whole-body movements based on the 3p trajectory.* ***Bottom:*** *Examples of Robo-Saber's whole-body gameplay.*

Abstract

We present the first motion generation system for playtesting virtual reality (VR) games. Our player model generates VR headset and handheld controller movements from in-game object arrangements, guided by style exemplars and aligned to maximize simulated gameplay score. We train on the large BOXRR-23 dataset and apply our framework on the popular VR game Beat Saber. *The resulting model* ***Robo-Saber*** *produces skilled gameplay and captures diverse player behaviors, mirroring the skill levels and movement patterns specified by input style exemplars. Robo-Saber demonstrates promise in synthesizing rich gameplay data for predictive applications and enabling a physics-based whole-body VR playtesting agent.*

CCS Concepts

• Human-centered computing → Virtual reality; User models; • Computing methodologies → Simulation by animation; Neural networks; Motion capture; Procedural animation;

1. Introduction

Traditional virtual reality (VR) development can be cumbersome. For example, developers are often required to leave their workstations to test the full-body movements involved in their application. Addressing this problem, simulating human perception and motor control through deep reinforcement learning (DRL) has emerged as a promising approach for automatic testing of VR interaction [IFK*22, FIK*24]. However, traditional DRL agents are ill-suited for representing how different users will move and interact, given the variability in body morphology, skill levels, and abilities within the user population.

Could we learn a player model from real-world data instead? The increasing availability of open-source VR gameplay data (*e.g.*, BOXRR-23, NGW*23) opens up opportunities for supervised learning as an alternative to *tabula rasa* DRL. In this work, we seek to produce a model that learns generalizable gameplay capabilities from such a dataset. Crucially, an ideal model would be calibrated to the diversity in skill levels and behavioral patterns in the training data. Such a player model could then generate comprehensive synthetic gameplay data for novel content, supporting *in silico* user modeling and automated playtesting use cases.

For producing this model, the following VR-specific computational challenges remain: First, many VR scenarios require interaction with 3D objects in richly configurable environments, making it challenging to effectively support the vast space of object configurations. Second, VR gameplay frequently involves extended-length motions, lasting several minutes or sometimes hours, requiring models to generate long, robust motion sequences that remain aligned with gameplay objectives throughout. Third, capturing behavioral diversity and output multimodality requires specific design choices that enable the representation of individual styles or behavior modes. Altogether, these challenges converge on a central problem: **generative motion planning**–learning to generate robust, well-aligned, and diverse motion plans for a vast space of input scenarios from real-world movement data.

In pursuing the above challenges, we develop and evaluate **Robo-Saber**, a novel gameplay agent for the VR game *Beat Saber* built on a conditional motion generation model. *Beat Saber* is especially well-suited to evaluating our framework: The VR game is an established benchmark across computational and non-computational research domains, widely popular, and offers extensive open-source data (see Sec. A in the Appendix). We train Robo-Saber on *Beat Saber* gameplay recordings from the large BOXRR-23 dataset [NGW*23]. Conditioned on in-game object states, Robo-Saber generates diverse three-point (positions and orientations of the headset and two handheld controllers, denoted as $3p$) motion plans aligned with gameplay objectives. The model utilizes *contextual gameplay exemplars* (*i.e.*, player-specific examples of gameplay movements paired with in-game object states) to distinguish between players based on individual characteristics such as skill level and movement patterns. Deployed autoregressively, Robo-Saber produces minutes-long gameplay trajectories, enabling automated testing of game content (*maps* in *Beat Saber*).

Robo-Saber is capable of skilled *Beat Saber* gameplay, closely competing with elite-level human players via generated kinematic trajectories. We demonstrate Robo-Saber's ability to simulate gameplay consistent with the human reference players' skill levels and movement patterns represented by their exemplars. We build on this result and leverage Robo-Saber for a player modeling use case: the *personalized score prediction* (PSP) scenario, *i.e.*, predicting scores for player-map combinations absent from the training data. Our results suggest that personalized gameplay can be simulated, enabling one to predict player performance on brand-new game content from a few gameplay examples. Furthermore, we find that Robo-Saber can synthetically augment training data for a downstream score-prediction model, thereby achieving impressive accuracy. Finally, we demonstrate that Robo-Saber readily interfaces with physics-based tracking. We examine the effect of tracking on player modeling performance and identify areas of opportunity, contributing towards building a physics-based whole-body VR player model.

Our technical contributions are summarized as follows. First, we introduce improvements to Categorical Codebook Matching (CCM, SSH*24): Our Transformer-based [VSP*17] encoder models support the use of *contextual exemplars* as conditioning signals. We also replace the mean squared error (MSE) matching loss in CCM with a logit-space Jensen-Shannon divergence (JSD) loss (Sec. 3). Furthermore, we develop and evaluate a rejection sampling scheme for motion plans sampled from our model (Sec. 3.3). We use kinematic gameplay performance evaluated via a custom GPU-accelerated game simulation, which greatly improves generalization compared to deterministic rollout (Sec. Q2 and Fig. 4). Applying these techniques to the popular benchmark VR game *Beat Saber*, we produce *the first style-conditioned generative VR player model* that lends itself to predictive applications and physics-based whole-body gameplay synthesis.

Project page. Our source code, lay-friendly summary, and supplementary videos can be found on our project page: `https://robo-saber.github.io`

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. We build on generative techniques for solving controllable and interactive motion synthesis (*e.g.*, HKS17, ZSKS18, LZCVDP20, TRG*23, SWJ*24), as well as more recent research successfully combining generative modeling with physics simulation for expressive physics-based character control (*e.g.*, XTS*23, TRC*24, HTZ*25, XSYP25). A key application domain of motion generation techniques is virtual reality (VR). For VR, motion generation often incorporates three-point ($3p$) pose data, as most consumer-grade VR equipment allows tracking the positions and orientations of the 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.*, YLHX22, DKP*23, SSH*24, BBM*25), *i.e.*, learning a $3p$-to-full-body correspondence from various motion 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 requires conditioning on game input. Towards this end, the BOXRR-23 dataset [NGW*23] provides $3p$ trajectories for well-known VR games, including

Beat Saber. In this work, we uniquely leverage the vast amount of *Beat Saber*-specific $3p$ pose data from BOXRR-23 and the open-source custom map database BeatSaver [Tea21], aligning game and pose data to train a conditional motion generation model for producing $3p$ movements directly from game states. Our incorporation of spatio-temporal goals follows the recent success of generative models for virtual instrument performance (*e.g.*, WXS*24, JZDP25, CZS*), extending the idea beyond symbolic music to VR gameplay. By integrating with a physics-based humanoid tracking controller (Sec. 5), our work also lays the foundation for simulating whole-body gameplay movements.

Stylized motion generation. Generating motions according to specific styles has long been of interest to character animation and generative modeling research. Early work focused on transferring style between two or more motions using nearest neighbor queries [XWCH15], convolutional autoencoder-based motion manifolds [HSK16, HHKK17], or spectral methods [YM16]. Subsequently, generative architectures for stylized motion synthesis have advanced, ranging from variational autoencoders [DHS*19] to flow models [WCHW21] and denoising diffusion models [ANBH23, AZL23, ZXJ*24, RGS*24, SGT*25, KJS*25]. A key challenge of the stylization problem is representing the elusive notion of style, which is sometimes manually annotated (*e.g.*, SCNW19, AWL*20) or often learned as latent codes from exemplar motions. These representations have been used as conditioning signals in synthesizing stylized motions over long horizons (*e.g.*, WCHW21, JPL*25, XLP*25). Similar to prior work [WCHW21, TZCvdP22], our work uses exemplar motions for autoregressive motion generation while learning a set of latent codes with a Transformer-based style encoder. Orthogonal to most recent stylized conditional generation work that aligns output motions with input text prompts (*e.g.*, ZXJ*24, JPL*25), our work instead aims to align output motions with gameplay objectives. Moreover, our framework operates with *contextual* exemplars, which provide movement signals as well as task information.

Computational user modeling. Constructing faithful models of user behavior is a long-pursued goal in human-computer interaction that promises to enhance our understanding of human behavior and expedite interactive system design. Modern approaches frame interaction as Markov decision processes, using deep reinforcement learning (DRL) to discover simulated rational behavior [OJH22]. DRL-based user models have been successfully applied across a wide range of complexity levels: from puzzle games [RRT*20] to embodied scenarios including touchscreen typing [JAU*21], gestural interaction [CFLN*20], muscle-and-tendon arm models [IFK*22], and VR interaction [FIK*24]. The increasing availability of real behavior data has enabled imitation learning approaches that calibrate to behavioral diversity (*e.g.*, dWLC22). Although DRL remains the dominant paradigm, we believe there is untapped potential in recent advances in data-driven approaches that could leverage massive real-world data. Thus, we seek to produce user models via supervised learning (*e.g.*, GEP*18) and extend promising results in generative models for human motion synthesis more broadly. A key promise in applying generative techniques to control and planning is *reward alignment* (*e.g.*, SSH*24, HTZ*25, TLH*25), which mirrors a human user's ability to evaluate and refine plans; our use of score-based selection applies this aspect to simulate gameplay.

3. Method

Our approach uses a generative motion planning model to produce $3p$ gameplay trajectories. **Fig. 1 overviews the complete pipeline; Fig. 2 details the generative model architecture and notation.**

3.1. Overview and Problem Formulation

Our objective is to develop a generative model that produces realistic VR gameplay motions conditioned on in-game object configurations. Specifically, we model the three-point ($3p$) poses of the VR headset and handheld controllers as they respond to dynamic game states. The key challenge is not only to generate motions that align with gameplay objectives, but also to capture the diversity of player behaviors and styles present in real human gameplay.

To address this, we formulate the task as learning the following conditional distribution:

$$\hat{\mathbf{p}}_{t:t+T} \sim p\left(\mathbf{p}_{t:t+T} \mid \mathbf{p}_{t-h:t}, \mathbf{x}_t^{\text{game}}, \mathbf{x}^{:N_{\text{ref}}}\right) \tag{1}$$

where $\hat{\mathbf{p}}_{t:t+T}$ denotes the generated future poses, $\mathbf{p}_{t-h:t}$ represents the length-h pose history, and $\mathbf{x}_t^{\text{game}}$ captures the current in-game object state (*e.g.*, incoming notes, bombs, obstacles). To capture player-specific characteristics, we condition on $\mathbf{x}^{:N_{\text{ref}}}$–short gameplay segments from the same player that serve as style exemplars. Detailed notation for these components is introduced in Sec. 3.4.

Our system follows a generate-simulate-select pipeline (Fig. 1): First, we use a reference-conditioned Categorical Codebook Matching model (CCM, Sec. 3.2) to generate multiple candidate $3p$ motion trajectories. Second, we evaluate these candidates using TorchSaber, a custom GPU-accelerated game simulator (Sec. 3.3). Finally, we select the trajectory that achieves the highest simulated gameplay performance. This model can be deployed autoregressively to produce gameplay sequences of arbitrary length.

Contextual exemplars. While using exemplar motion as a style signal is well-explored in prior work (*e.g.*, XWCH15, HSK16, WCHW21, GMZ*24, RGS*24, SGT*25, KJS*25), we extend the idea in two ways. First, each style reference in $\mathbf{x}^{:N_{\text{ref}}}$ includes not only the player's motion but also the game state $\mathbf{x}_t^{\text{game}}$ during that motion—allowing the model to learn how players respond to different gameplay situations rather than just their unconditional movement patterns. Second, we condition on multiple reference examples (N_{ref}) simultaneously to capture consistent behavioral patterns across varied scenarios. Our experiments (Sec. 4) further validate the use of contextual exemplars.

3.2. Reference-Conditioned Categorical Codebook Matching

Please refer to Fig. 2 for the visualization of our codebook matching process. To model the conditional distribution in Eq. (1), we implement Categorical Codebook Matching (CCM, SSH*24). We use a Gumbel-Softmax variational auto-encoder (GS-VAE) to encode each individual $3p$ trajectory, such that its embedding corresponds to the logits of a categorical distribution. To support conditioning

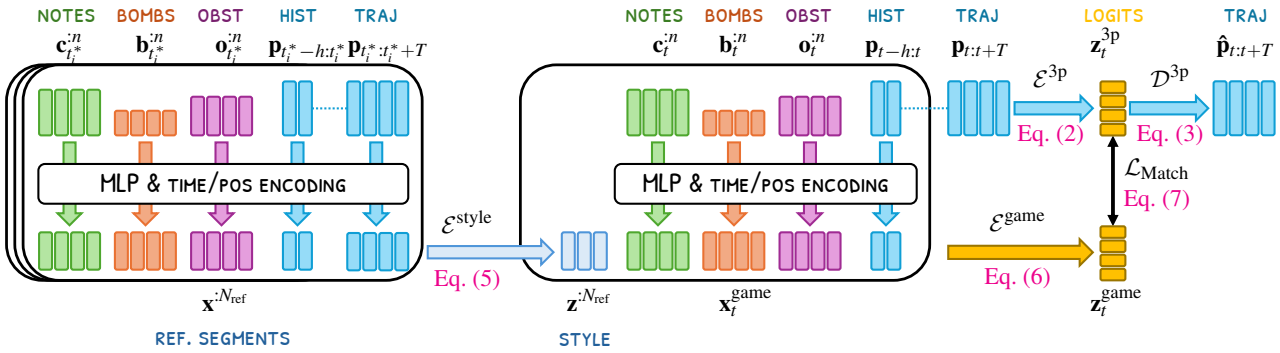

Figure 2: *Overview of our generative model architecture, extending Categorical Codebook Matching (CCM, SSH*24, Sec. 3) with Transformer encoder models $\mathcal{E}^{style}$ and $\mathcal{E}^{game}$, as well as a modified loss function based on Jensen-Shannon divergence.*

by reference, we extend SSH*24's CCM encoder block, introducing a style encoder $\mathcal{E}^{\text{style}}$ that embeds style reference examples; the resulting encoder block consisting of two Transformer-based models ($\mathcal{E}^{\text{style}}$ and $\mathcal{E}^{\text{game}}$) is trained to predict the logits from the corresponding input conditions, *i.e.*, $\mathbf{p}_{t-h:t}$, $\mathbf{x}_t^{\text{game}}$, and $\mathbf{x}^{:N_{\text{ref}}}$.

More formally, the length-T pose sequence $\mathbf{p}_{t:t+T}$ is mapped to the logits (Sec. 3.4 overviews feature notation and representations in more detail):

$$\mathbf{z}_t^{\text{3p}} = \mathcal{E}^{\text{3p}}\left(\mathbf{p}_{t:t+T}\right). \tag{2}$$

The logits are reshaped into those of a joint categorical distribution consisting of C channels, each with 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, the one-hot samples retain their gradients through the reparameterized Gumbel sampling procedure [JGP17]. The resulting one-hot samples are then decoded into

$$\hat{\mathbf{p}}_{t:t+T} = \mathcal{D}^{\text{3p}}\left(\text{GumbelSoftmax}\left(\mathbf{z}_t^{\text{3p}}\right)\right). \tag{3}$$

Hence, the GS-VAE's auto-encoding loss is simply the mean squared error (MSE) loss between the original and reconstructed input, *i.e.*:

$$\mathcal{L}_{\text{Recon}} = \frac{1}{T}\left|\left|\mathbf{p}_{t:t+T} - \hat{\mathbf{p}}_{t:t+T}\right|\right|^2 \tag{4}$$

Our GS-VAE departs from the multi-layer perceptron (MLP) architecture of SSH*24 and instead utilizes a Transformer architecture [VSP*17] to better leverage the sequential nature of the $3p$ pose sequence. The details of our Transformer-based GS-VAE are discussed in Sec. C in the Appendix.

To match our conditioning signals with a corresponding $\hat{\mathbf{p}}_{t:t+T}$, we train an encoder block consisting of the style encoder $\mathcal{E}^{\text{style}}$ and the game segment encoder $\mathcal{E}^{\text{game}}$. $\mathcal{E}^{\text{style}}$ first embeds the reference examples into

$$\mathbf{z}^{:N_{\text{ref}}} = \mathcal{E}^{\text{style}}\left(\mathbf{x}^{:N_{\text{ref}}}\right). \tag{5}$$

$\mathcal{E}^{\text{game}}$ predicts the logits

$$\mathbf{z}_t^{\text{game}} = \mathcal{E}^{\text{game}}\left(\mathbf{p}_{t-h:t}, \mathbf{x}_t^{\text{game}}, \mathbf{z}^{:N_{\text{ref}}}\right). \tag{6}$$

Taking into account the varying number of objects captured in purview, as well as their continuous timing values (when each object reaches the player), we also employ a Transformer architecture for both $\mathcal{E}^{\text{style}}$ and $\mathcal{E}^{\text{game}}$. That is, all of the objects and poses in $\mathbf{p}_{t-h:t}$, $\mathbf{x}_t^{\text{game}}$, and $\mathbf{x}^{:N_{\text{ref}}}$ are projected to the d-dimensional input space via an MLP prepared for each domain (among notes, bombs, obstacles, poses).

The resulting embeddings then receive position encoding; importantly, position encoding for game objects is based on their timing, which informs their temporal arrangements (as opposed to indices, which only inform their order of appearance). Pose sequences, however, receive position encoding based on index per usual, as the time between poses is constant. We register an all-zero sentinel token [DCLT19, DBK*20] concatenated to the end of the projected sequences and feedforward its corresponding final embedding through an MLP head to produce $\mathbf{z}^{:N_{\text{ref}}}$ and $\mathbf{z}_t^{\text{game}}$.

Codebook Matching via Jensen-Shannon divergence. Per the usual CCM formulation [SSH*24], we encourage the similarity of the two categorical distributions $\mathbf{z}_t^{\text{3p}}$ and $\mathbf{z}_t^{\text{game}}$. A sufficient match between the two enables the conditional generation in Eq. (1) by connecting $\mathcal{E}^{\text{game}}$ to $\mathcal{D}^{\text{3p}}$ (*i.e.*, omitting $\mathcal{E}^{\text{3p}}$ at inference).

While SSH*24 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. The JSD-based matching loss is computed according to:

$$\mathcal{L}_{\text{Match}} = D_{\text{KL}}\left(\mathbf{z}_t^{\text{3p}} \middle|\middle| P\right) + D_{\text{KL}}\left(\mathbf{z}_t^{\text{game}} \middle|\middle| P\right) \tag{7}$$

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:

$$\mathcal{L} = \mathcal{L}_{\text{Recon}} + \lambda_{\text{Match}} \cdot \mathcal{L}_{\text{Match}} \tag{8}$$

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

3.3. Inference: Candidate Trajectory Selection

Incorporating rejection sampling in generative motion planning has been explored more recently, *e.g.*, in SSH*24, IRCvdP25. Leveraging the variational capabilities, trained generative models can typically produce multiple viable candidate outputs, among which a suitable one is selected. While evaluating generated motion plans can be generally non-trivial, often calling for learned proxies (*e.g.*, IRCvdP25), gameplay is fortunately well-suited for simulation. Consequently, we take the approach of *combining game simulation with GS-VAE's sampling procedure* for generative motion planning. Given the predicted logits $\mathbf{z}_t^{\text{game}}$, we sample N_{traj} *candidate* $3p$ trajectories:

$$\hat{\mathbf{p}}_{t:t+T}^{:N_{\text{traj}}} \sim \mathcal{D}^{3\text{p}}\left(\text{GumbelSoftmax}\left(\mathbf{z}_t^{\text{game}}\right)\right) \tag{9}$$

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

$$j^* = \underset{j=1,2,\cdots,N_{\text{traj}}}{\arg\max}\ \text{Evaluate}\left(\hat{\mathbf{p}}_{t:t+T}^{j}, \mathbf{x}_t^{\text{game}}\right) \tag{10}$$

Unlike SSH*24, we use the entire T-frame prediction without intermediate re-planning; we find the motion quality for the $3p$ trajectories to be better overall when the output chunks are coherent and continuous.

TorchSaber: A GPU-accelerated Beat Saber simulator. As *Beat Saber* remains a proprietary, closed-source game, evaluating scores for generated trajectories inside the real game would require interacting with the compiled game or building an additional integration layer. We instead develop and use TorchSaber, a GPU-accelerated *Beat Saber* simulator, as an alternative (details in Sec. D in the Appendix). TS produces simplified, normalized proxy scores that omit parts of the official scoring (*e.g.*, combos and some cut-angle terms), while still correlating strongly with real scores; evaluating the held-out human play data, we obtain Pearson's r=0.856 between TS and recoded *Beat Saber* scores.

Reward function. Based on TorchSaber's simulation results, we assign a reward value r_j to each candidate output $\hat{\mathbf{p}}_{t:t+T}^{j}$, as defined below (Sec. E of the Appendix provides a detailed discussion of the reward terms):

$$r = r_{\text{TS}} - \lambda_{\text{Bomb}} \cdot r_{\text{Bomb}} + \lambda_{\text{Obstacle}} \cdot r_{\text{Obstacle}} \tag{11}$$

- $r_{\text{TS}} \in [0,1]$ is the TS score, calculated for the $3p$ poses and colored notes.
- $r_{\text{Bomb}} \in [0,1]$ is the bomb penalty, computed as the ratio between the number of bomb hits and the number of 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 becomes negative.
- The λ_* values are weights corresponding to each term; see Table 1 of the Appendix for the values used in our experiments.

3.4. Training Setup and Features

Our system requires a diverse and extensive dataset to generalize properly. For *Beat Saber*, BOXRR-23 [NGW*23] and BeatSaver [Tea21] datasets provide this scale and diversity, featuring millions of replay sequences from hundreds of thousands of players and maps. Given the dataset's characteristics and quality issues, we apply quality control (QC) measures to ensure that our model is trained on correctly aligned, high-quality data (detailed in Sec. B in the Appendix). Our post-QC dataset then consists of 2,357,627 replay sequences from the pool of 85,497 maps and 71,070 players. By aligning the gameplay data of BOXRR-23 with the map data of BeatSaver [Tea21], we produce supervised training samples.

Pose representation. The 27-dimensional $3p$ pose vector $\mathbf{p}$ (at each timestep) is constructed by concatenating the global xyz-coordinates and 6-dimensional orientation [ZBL*19] for each of the three parts: the headset and two handhelds.

In-game object representation. The in-game object configurations follow the Beat Saber Modding Group format [Bea19]. Each colored note $\mathbf{c}$ is represented by grid position indices, note color, and cut direction (dim=4). Each bomb $\mathbf{b}$ and obstacle $\mathbf{o}$ is represented by grid position indices; obstacles additionally include width, height, and depth (dim=2 and 5, respectively). Each object is paired with its absolute timestamp, indicating when it reaches the player.

We use a *lookahead* horizon: only up to n objects within s seconds are considered at timestep t. For each object captured in lookahead, we compute its relative timing, *i.e.*, the difference between the current game timestamp and the object's absolute timestamp. Using subscript t to denote objects in purview at t and superscript ":n" to denote a sequence containing up to n objects, *e.g.*, $\mathbf{c}_t^{:n} = \left(\mathbf{c}_t^1, \mathbf{c}_t^2, \cdots, \mathbf{c}_t^{k_{\mathbf{c}_t}}\right), k_{\mathbf{c}_t} \leq n$, we construct the game state observation:

$$\mathbf{x}_t^{\text{game}} = \left(\mathbf{c}_t^{:n}\ \mathbf{b}_t^{:n}\ \mathbf{o}_t^{:n}\right). \tag{12}$$

We pair $\mathbf{x}_t^{\text{game}}$ with $\mathbf{p}_{t-h:t}$ and $\mathbf{p}_{t:t+T}$.

Style reference representation. A gameplay reference example is similarly constructed as $\mathbf{x}_{t_i^*}^{\text{ref}} = \left(\mathbf{c}_{t_i^*}^{:n}\ \mathbf{b}_{t_i^*}^{:n}\ \mathbf{o}_{t_i^*}^{:n}\ \mathbf{p}_{t_i^*-h:t_i^*}\ \mathbf{p}_{t_i^*:t_i^*+T}\right)$ for some timestep t_i^*, sampled from a random replay sequence i of the same player. Using the superscript ":N_{ref}" to represent a collection of size N_{ref}, we denote a set of gameplay reference examples as:

$$\mathbf{x}^{:N_{\text{ref}}} = \left(\mathbf{x}_{t_1^*}^{\text{ref}}, \mathbf{x}_{t_2^*}^{\text{ref}}, \cdots, \mathbf{x}_{t_{N_{\text{ref}}}^*}^{\text{ref}}\right) \tag{13}$$

with a random time t_i^* for each player-specific reference $i \in \{1,2,\cdots,N_{\text{ref}}\}$.

4. Experiments

Below, we validate Robo-Saber's behavior on held-out validation data, using the TS scores of both Robo-Saber and human players as the main quantitative measure. Importantly, the human $3p$ trajectories are fed into TS to re-compute their scores, so that the scores can be directly compared.

Our primary research question is: *Can Robo-Saber enable automated and personalized VR playtesting?* More specifically:

Q1. Is Robo-Saber capable of general *Beat Saber* gameplay?
Q2. Does incorporating TorchSaber impact generalization?

Q3. Can Robo-Saber reproduce reference skill and style?
Q4. Can Robo-Saber predict personalized scores?

To briefly summarize the results, we find that Robo-Saber is capable of elite-level *Beat Saber* gameplay and the game simulation improves generalization. The model's generated gameplay, emulating held-out players, achieves a strong correlation of $r = 0.789$ with the ground-truth scores on held-out maps. Note that this section focuses on the $3p$ (the headset and handhelds) motion generation, which is suitable for VR player modeling where the player's full-body movement does not have to be inferred or visualized. Extending Robo-Saber with physics-based full-body movement generation is explored in Sec. 5.

Comparing model variants. Note that our experiments compare three Robo-Saber variants: (1) a *reference-agnostic* model that does not utilize $\mathbf{x}^{:N_{\text{ref}}}$ ($N_{\text{ref}} = 0$), (2) a *reference-aware* model with $N_{\text{ref}} = 1$, and (3) a reference-aware model with $N_{\text{ref}} = 5$. For evaluating statistical significance, we report the mean $\pm$ standard error in figure legends, wherever appropriate, and apply Wilcoxon's signed-rank test [Wil92] to compute the p-value.

Holdout data preparation. We prepare holdout data as follows. First, from the post-QC dataset prepared from BOXRR-23, we exclude the most popular 1% of the maps, as measured by the number of unique players who have played them. This prevents the model from overfitting to most typical patterns and enables a stringent test for generalization. Our visual results are rendered on a selected subset of these 847 most popular maps.

From the remaining maps and the player population, we hold out 10% from each, selecting uniformly at random. The holdout data consists of 66,397 player-map pairs between 7,107 players and 8,465 maps. For our quantitative results, we report the performance on this set. Importantly, neither any player nor any map in the holdout appears during training.

Q1. Is Robo-Saber capable of general *Beat Saber* gameplay?

We benchmark Robo-Saber's performance against human players on the held-out test set. To ensure we elicit the best performance from the reference-aware model variant, we sample player-specific $\mathbf{x}^{:N_{\text{ref}}}$ from the top 5% of held-out players. We use the same $\mathbf{x}^{:N_{\text{ref}}}$ for simulating the same player across different maps. While it is possible for a player to have varying skill levels across gameplay records, we simplify by excluding skill level variation from consideration, as players tend to report their best-effort attempt on leaderboards. **Please refer to the supplementary video for qualitative results featuring $3p$ *and* whole-body gameplay animations.**

Robo-Saber vs. humans. As shown in Fig. 3, Robo-Saber's TS scores closely compete with those of human players; the $N_{\text{ref}} = 5$ variant outperforms human players on average at slightly above the 60th percentile. Interestingly, $N_{\text{ref}} = 1$ results in lower overall performance than $N_{\text{ref}} = 0$, perhaps because a single randomly chosen reference segment may fail to adequately represent gameplay skills. For example, the single reference segment may be an idle standing motion, which is insufficient to signal to the model that high-performing samples should be produced. Providing a stronger style-conditioning signal, *i.e.*, via more exemplars, remedies this and produces evidently better performance.

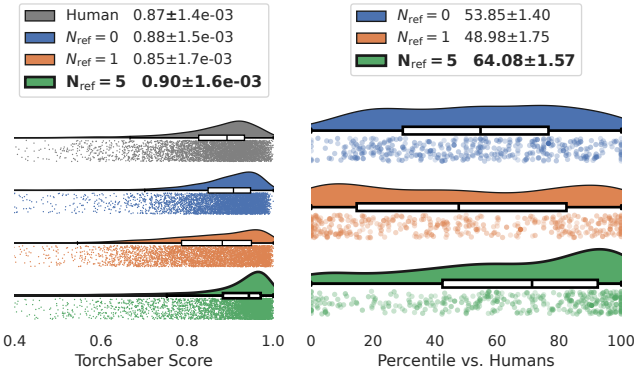

Figure 3: *Comparing Robo-Saber's performance to that of human players. Robo-Saber trajectories are produced by using $N_{ref} = 5$ segments from elite (top 5%) players.* ***Left:*** *Human and Robo-Saber TS score distributions across all held-out maps are shown as raincloud plots.* ***Right:*** *Robo-Saber's performance relative to human players is quantified as score percentiles. To compute the percentiles, for each difficulty level, we select the 100 most-played maps (400 total) and compare Robo-Saber's score against human scores in each map. The number of human scores available in each map is visualized by opaqueness (max=388, min=3) The performance statistics are summarized as mean $\pm$ standard error. The reference-aware model with 5 reference segments from top players ($N_{ref} = 5$) outperforms humans on average.*

Performance across difficulty levels. The top part of Fig. 8 shows the breakdown of percentile distribution across difficulty levels, revealing that Robo-Saber performs better than the 50th percentile against human scores up to the Expert difficulty. However, interpreting performance on Expert+ difficulty should take into account the heavy skew towards elite-level players.

Q2. Does incorporating TorchSaber impact generalization?

Our candidate selection strategy uses rejection sampling, with TS as a simulator to evaluate each candidate. This alignment step impacts how well our model generalizes. Choosing only the trajectory corresponding to the peak of the output logit (using argmax actions) can lead to worse performance, because the model may not always produce correct logits at each stage, especially in previously unseen game states. To address this, our system generates N_{traj} sample trajectories and selects the best one after evaluation (see Fig. 13). This approach closely resembles black-box model predictive control frameworks, which utilize a simulator to propose and refine candidate motion plans before selection. The effectiveness of this approach is evident in Fig. 4, where rollouts with candidate trajectory selection consistently outperform argmax rollouts, indicating stronger generalization. Notably, larger N_{traj} results in better performance in general. Using Wilcoxon's signed-rank test, we obtain $p < 0.005$ comparing the percentiles between argmax

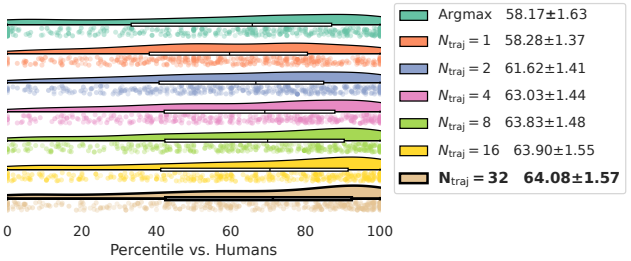

Figure 4: *Sampling-based candidate trajectory selection improves performance compared to using deterministic Argmax selection for GS-VAE during inference. The percentiles are evaluated on the same held-out maps as Fig. 3 with $N_{ref} = 5$ and $\mathbf{x}^{:N_{ref}}$ sampled from elite (top 5%) players. The TS score percentiles are summarized as mean ± standard error for each configuration.*

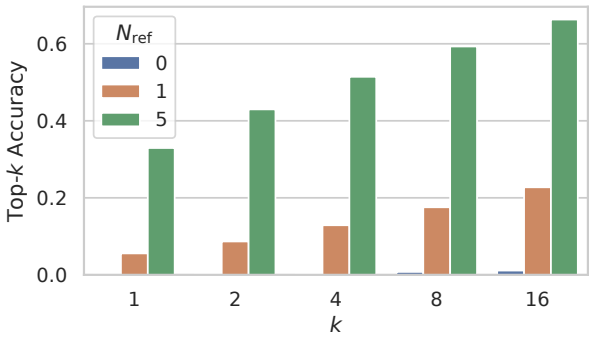

Figure 5: *Quantifying Robo-Saber's ability to produce trajectories consistent with the style reference. The oracle player classifier's top-k accuracy measures how well the generated $3p$ trajectories are recognized. Adding style reference segments clearly improves the recognizability.*

and $N_{traj} = 32$. These results are consistent with the findings of SSH*24, who also examined inference-time candidate selection. While their work focused on the continuity of generated full-body motion, our approach specifically targets optimizing gameplay.

Q3. Can Robo-Saber reproduce reference skill and style?

Our model is tasked with recognizing player-specific characteristics from the input references and incorporating them into gameplay. A well-calibrated model would allow one to simulate a player's gameplay on entirely new game content from using a few reference gameplay segments. In other words, we aim to validate whether the model emulates a player's skill level and movement patterns, replicating the *personality* of that player.

Calibration to skill level. Our results suggest that our model calibrates to the skill level represented by the input $\mathbf{x}^{:N_{ref}}$. For a given player-map, a well-calibrated model should perform similarly to the given player; that is, conditioning on novice examples should yield low scores and expert examples high scores. This is illustrated in Fig. 6: compared to the reference-agnostic baseline ($N_{ref} = 0$),

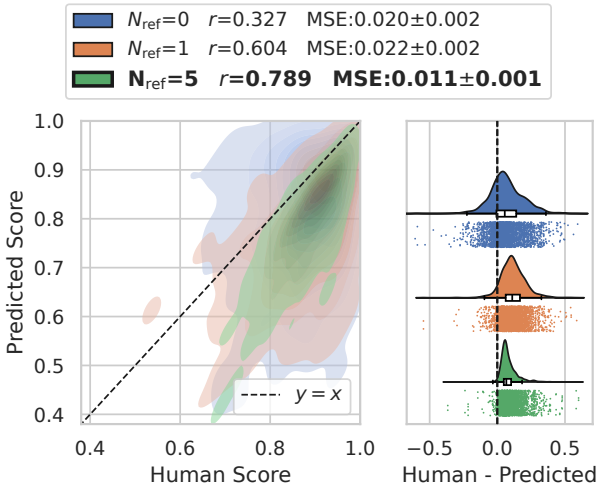

Figure 6: *Quantifying Robo-Saber variants' calibration to the skill levels of human players, measured by the Pearson correlation (r) between Robo-Saber and human players' performance on held-out maps. **Left:** Densities of points comparing robot (predicted) scores against ground truth human scores. The reference-aware Robo-Saber with $N_{ref} = 5$ achieves a strong correlation of $r = 0.789$. **Right:** Distributions of score differences between humans and Robo-Saber variants.*

our reference-aware model variants more accurately approximate the ground truth human performance, resulting in moderate correlations and residuals distributed closer to 0. Unsurprisingly, the signals in $\mathbf{x}^{:N_{ref}}$ get clearer with a larger N_{ref}, resulting in a stronger correlation in TS scores and generally smaller score differences.

Calibration to movement patterns. Individual players' movement patterns contain rich information, such as spatial positioning habits and various biases in movements–NGM*23 and NGO*24, for example, showed that these traits can identify which player has produced the $3p$ trajectory. Inspired by this, we systematically assess whether the generated output resembles the input exemplars. We train an *oracle player classifier*, which predicts held-out player labels from held-out gameplay segments. The oracle's recognition indicates whether an output gameplay sequence resembles that of a specified player and thereby whether the generator utilizes the signals inherent to $\mathbf{x}^{:N_{ref}}$ to produce output consistent with them.

We quantify the output trajectories' recognizability with the oracle's top-k classification accuracy; *i.e.*, we sample $\mathbf{x}^{:N_{ref}}$ for a held-out player, generate a $3p$ trajectory using $\mathbf{x}^{:N_{ref}}$, compute the oracle classifier's logits from the trajectory, and assess whether the top k values in the logits include the category corresponding to the held-out player's ID. Intuitively, a higher top-k accuracy value indicates higher similarity between the model's and the referenced player's movements. The oracle uses a Transformer architecture similar to $\mathcal{E}^{style}$ in Eq. (5); see Sec. C in the Appendix for more details. As shown in Fig. 5, the oracle's top-k accuracy, across different values of k, is significantly higher when we condition Robo-Saber with style information, and the effect increases with N_{ref}.

Qualitative result. As shown qualitatively in Fig. 15 and the supplementary video, the input $\mathbf{x}^{:N_{\text{ref}}}$ can instruct what the generated movements should look like. Fig. 15 shows two generated trajectories on the same held-out map, one conditioned on the reference segments $\mathbf{x}^{:N_{\text{ref}}}$ of an elite-level player and the other on a novice player. The output trajectories reflect visible characteristics of the reference movements, such as swinging speed, anticipatory movements, and the positioning of the handhelds relative to the headset.

Q4. Can Robo-Saber predict personalized scores?

Sec. Q3 establishes a strong correlation between ground-truth human performance and Robo-Saber's player simulation. While directly simulating user behavior is a viable way to predict scores, we seek to leverage the strong correlation by refining the *in silico* player simulation results into signals for predictive models. Thus, we investigate a collaborative filtering approach for personalized score prediction (PSP), inspired by KGBH22, and examine the viability of Robo-Saber as a synthetic data augmentation mechanism for predictive applications.

Q4.1. Setup: PSP via collaborative filtering

Collaborative filtering (CF) enables predicting variables for user-item pairs, such as a target player's difficulty playing a target game level, based on how difficult other players found it [KGBH22]. In our case, we seek to predict TS scores for player-map pairs. As a novel approach to leverage synthetic gameplay data, we augment CF training data with Robo-Saber scores. We use a *diverse population of Robo-Saber agents*, with diversity elicited by the reference-aware $3p$ trajectories representing different players using $N_{\text{ref}} = 5$.

Data for collaborative filtering. For testing, we generate a dataset $\mathbf{N}$, representing new maps that have never been played, each paired with a single unique player in the population. Our aim is to predict the scores for each player-map pair in $\mathbf{N}$ by learning from some existing performance data of the player population. To construct $\mathbf{N}$, we pair 1,000 held-out maps with 1,000 held-out players in a one-to-one match. Then, we construct $\mathbf{R}$, representing the existing population-wise performance data, including a wide pool of player population, including those in $\mathbf{N}$ (the maps do not overlap). $\mathbf{R}$ is produced by removing the maps in $\mathbf{N}$ from the remaining holdout and comprises 11,357 player-map pairs.

Synthetic data augmentation. While KGBH22 only used human player data, we augment the CF training data with Robo-Saber scores, *i.e.*, deploy Robo-Saber on $\mathbf{N}$ and $\mathbf{R}$ to produce additional datapoints for CF–we refer to them as $\hat{\mathbf{N}}$ and $\hat{\mathbf{R}}$. For each player-map pair, we emulate the player as in Sec. Q2, but additionally indicate that the data is synthetic. Consequently, the brand-new maps in $\mathbf{N}$ are now associated with the simulated players' performance, providing a basis for training a score-prediction model. Then, we attempt to predict real players' scores on those brand-new maps.

Q4.2. Results

From the player-map pairs annotated with human TS scores in $\mathbf{R}$ and synthetic TS scores in $\hat{\mathbf{N}}$ and $\hat{\mathbf{R}}$, we learn factorization machines (FMs, Ren10) to perform collaborative filtering. We validate our FM's predictions on the player-map pairs in $\mathbf{N}$. Briefly,

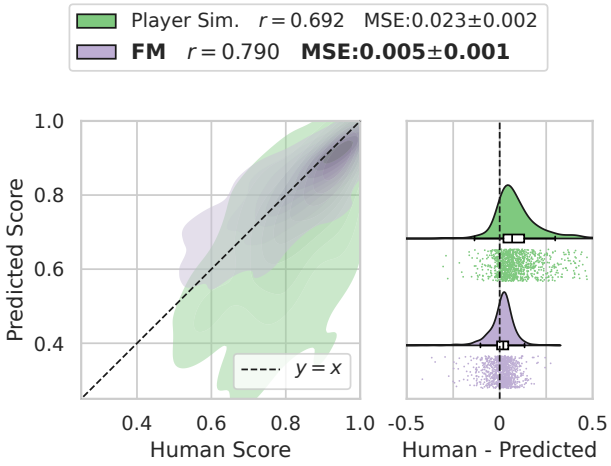

Figure 7: ***Left:*** *2D density plot comparing predicted and ground truth TS scores evaluated on* $\mathbf{N}$*. We compare our FM configuration (violet) with the direct player simulation results (green). A significant reduction in MSE and improvement in Pearson's r is detected.* ***Right:*** *A raincloud plot comparing the residuals. We report the MSE* $\pm$ *standard error in the legend.*

FMs learn player and map embeddings, such that their dot product can predict the score of the player-map pair; Sec. F and Sec. H in the Appendix provides more detailed description of FMs, our customization, and the hyperparameters used in training our FM model. We evaluate the PSP success in terms of mean squared error (MSE) and Pearson's correlation (r) across the real player-map pairs in $\mathbf{N}$. The baseline for our comparison is directly using Robo-Saber's ability to emulate players in $\mathbf{N}$, without using an FM (referred to as *Player Sim.* in Fig. 7).

We report the result in Fig. 7, which indicates that FM's prediction greatly increases the PSP performance in both MSE (0.006 ± 0.001) and r (0.794). Notably, as $\mathbf{N}$ is not identically distributed as the entire held-out population, we observe a discrepancy in the player simulation's performance vs. Sec. Q3. Nonetheless, the degraded prediction performance is remedied fully by learning meaningful FM embeddings. Using Wilcoxon's signed-rank test, we compare the squared errors of player simulation and FM and detect statistical significance ($p \ll 0.00001$).

5. From $3p$ Generation to Full-body Movement

In this section, we examine whether our generative model can help enable a physics-based full-body VR user model and report the performance of our resulting system. Computational user models thus far feature limited or no embodiment, *e.g.*, predicting the difficulty of mobile games using a player model that directly generates touchscreen interaction events without simulating the human hand [RRT*20, RGR*21, KVB20], or simulating touchscreen typing using a simplified model of finger movements [SZFJ*25]. In the rare systems featuring intelligent control of an actual physical or biomechanical simulation model, the focus has been on simulat-

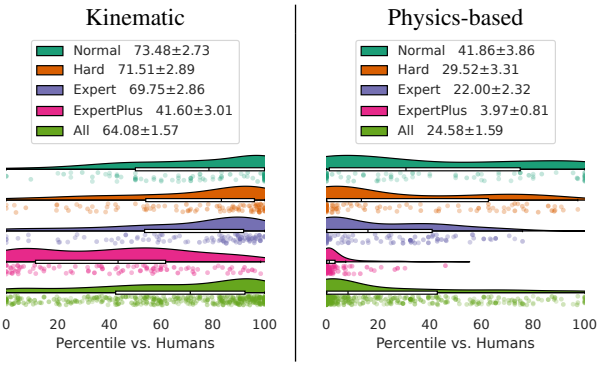

Figure 8: *Evaluating the performances of kinematic **(left)** and physics-based **(right)** versions of Robo-Saber ($N_{ref} = 5$ for both). The distribution of Robo-Saber's TS score percentiles per map is shown across difficulty levels. Physics-based trajectories are produced by tracking the generated 3p trajectories. As expected, absolute performance degrades due to physical/embodied constraints and tracking errors. The kinematic agent achieves respectable percentiles across all difficulty levels. While the physics-based agent achieves percentiles above 40% on average on Normal maps, the performance is poor on Expert and above.*

ing only a single arm [CFLN*20, IFK*22, FIK*24]. Consequently, developing a physically simulated full-body user model for embodied interaction remains an open challenge.

We propose and evaluate a solution that leverages Robo-Saber's 3p motion generation combined with physics-based full-body motion tracking. Namely, we interface our 3p generative model with a popular physics-based humanoid tracking controller from Perpetual Humanoid Control foundation model (PHC, LCK*23). Sec. G in the Appendix describes the details of our physics-based tracking implementation. We then investigate the following questions:

Q5. How performant is physics-based tracking?
Q6. How does physics-based tracking impact PSP?

Q5. How performant is physics-based tracking?

Quantitative results. Tracking with the whole body would naturally degrade and alter 3p movements, due to tracking error and additional constraints introduced by physics simulation; this is an expected consequence of added physical realism rather than an anomaly. The right half of Fig. 8 shows the effect of tracking the 3p trajectories for the left half. Introducing physical constraints indeed results in lower scores across difficulty levels. The physics-based Robo-Saber does not keep up with humans on Expert and Expert+ maps, although it performs reasonably well on Normal and Hard. This is presumably due to the tracking controller being unable to achieve sufficient movement speed and precision for difficult gameplay; we envision further research to enable higher physics-based performance as a promising direction, as noted in Sec. 6.

Qualitative results. As shown in the supplementary video, the full-body gameplay performance remains respectable when deployed on the most popular 1% of maps. Figs. 10–12 show image sequences demonstrating how the robot player moves in response to the observed colored notes, bombs, and obstacles. Unlike in previous physics-based user models, whole-body movements such as swaying and ducking can be observed. As a native feature of our GS-VAE architecture, a diverse set of whole-body trajectories can be generated as a response to the same input (Figs. 13 and 14). While random seed variations on the style-agnostic baseline exhibit some diversity in output, we observe significantly more visible diversity when varying the style references.

Q6. How does physics-based tracking impact PSP?

As previously discussed, physics-based tracking introduces errors and constraints that degrade gameplay performance. Rather than random, however, this degradation is owed to gameplay difficulty; prompted by this, we examine whether the physics-based player model's scores, though much less performant than the kinematic player model, can provide meaningful additional signals for user modeling. To this end, we replicate the CF approach for PSP as above, but we further augment the synthetic Robo-Saber data with scores obtained with physics-based tracking. Given that we need sufficiently performant and consistent scores, we elect to obtain synthetic physics-based data by simulating 3 randomly-chosen expert reference players across all maps in **N** and **R**, instead of simulating the ground truth players as in Sec. Q4. We add the resulting physics-based data and the corresponding kinematic data in FM training, thereby isolating the signals introduced by tracking.

Fig. 9 illustrates evidence that the additional data might contain signals that can improve PSP performance. The curves indicate that the FM model including physics-based data (KP) may underfit at smaller embedding sizes, while accuracy improves at larger ones. Intuitively, learning meaningful embeddings for the maps in **N** is the key objective of our FM, and the additional signals added by physics-based synthetic data may help refine them. However, as shown in the figure, the absolute PSP performance improvement by using additional scores alone is statistically weak; more experimentation may be necessary to ascertain the significance of physics-based gameplay data more concretely. As we note in Sec. 6, incorporating additional physical features such as energy expenditure and fatigue into the FM formulation may be beneficial to this end.

6. Conclusion

This work introduces a generative framework for producing a VR player model that exhibits diverse, skilled gameplay behavior. By calibrating to individual skill levels and movement patterns, Robo-Saber demonstrates that *a player's gameplay can be simulated on entirely new game content* using a few example gameplay segments. While the precise concept of difficulty is elusive, estimating how well a particular player would perform on a wide range of maps paves the way for practical applications. Map designers could leverage Robo-Saber to simulate diverse user gameplay and assess their designs automatically. Players could discover new content tailored to their skill levels and personal play styles, enhancing their overall experience. These capabilities position the incorporation of style-aware player models in user modeling as a fertile research di-

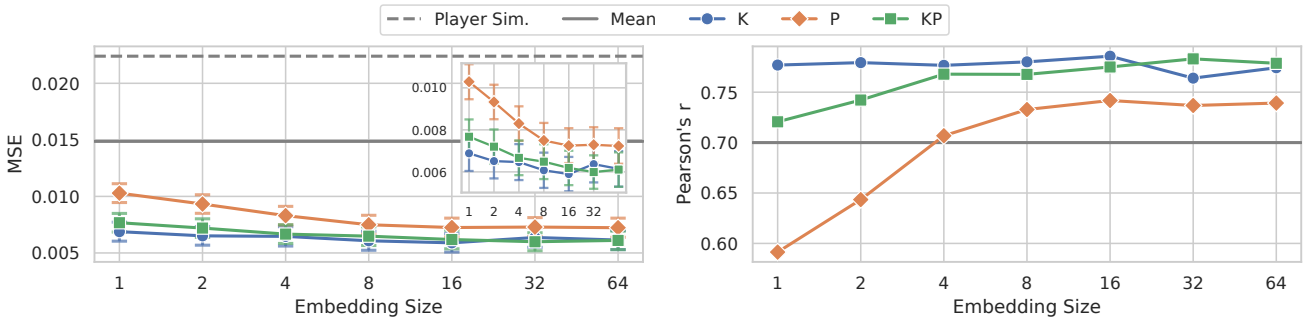

Figure 9: *Examining the effect of including physics-based tracking results in factorization machine (FM) training. Each point corresponds to a converged FM model's report of the validation metric in question. K=kinematic, P=physics-based, KP=both kinematic and physics-based, annotating the synthetic dataset included in FM training. "Mean" is the baseline of using the mean score in* **R** *as a constant predictor. Standard error of the mean is visualized with bars at each point.* ***Left:*** *As the FM embedding size grows, MSE on* **N** *for KP converges to lower values than those of others, while possibly underfitting at smaller embedding sizes. P nearly competes with K, suggesting the presence of some meaningful signal for PSP. The inset figure zooms into the curves at the bottom.* ***Right:*** *Pearson's r for* **N** *converges to higher values for KP at larger embedding sizes but shows diminishing returns.*

rection for enhancing VR application design and user experience, with our work contributing both a benchmark and baseline solution.

Applicability for other VR scenarios. We envision VR platforms' inherent telemetry and data-collection capabilities to enable developers of both large and small games to collect the motion data needed to train with our method. In our work, Robo-Saber's ability to extract style signals relies on the behavioral diversity accumulated over 70k players. How well this performs at smaller scales (*e.g.*, dozens/hundreds of players) is an important direction for future work and may require improved feature engineering and model architectures. Developing and evaluating a player model requires either programmatically accessing official gameplay or building a sufficiently faithful proxy for it (such as TorchSaber in this work). Given such an environment and suitable $3p$ example data, our framework should generalize to VR scenarios beyond *Beat Saber*. Behavioral variation across players is ubiquitous in VR games due to morphological and habitual factors, and our work validates contextual exemplar embeddings as a general approach to capture this diversity. As *tabula-rasa* DRL remains infeasible for modeling diverse player behaviors, learning from data-mined play sequences will likely remain a promising direction. Currently, *Beat Saber* is the only VR game with substantial open-source gameplay data. We hope to spur further data collection and follow-up work on generative player models for other VR games.

Utility of automated playtesting. We emphasize that our goal is to augment, not replace, costly human playtesting. Presently, only human playtesters can provide reliable information about subjective experience like fun. Robo-Saber's primary utility is to test the feasibility of a new map and predict its difficulty relative to other maps, helping craft a smooth difficulty curve. Evaluating the general utility of automated playtesting for design and personalization would be an orthogonal yet exciting direction for future work.

Performance of physics-based tracking. Further work is needed before a physics-based player model can fully realize its benefits. A primary limitation of our system is the degraded gameplay performance due to physics-based tracking. As some generated $3p$ goals are too challenging or dynamically infeasible for the tracker, a tighter communication between the kinematic generator and the tracking controller could alleviate the issue. Additionally, as seen in the video, the produced whole-body motions remain far from human-like or realistic. We highlight bridging these gaps as a compelling direction for future work. To maximize physics-based gameplay performance, one might integrate the simulation of not only the game mechanics but also physics and tracking performance into candidate trajectory selection, or fine-tune the $3p$ generation and/or physics-based tracking end-to-end using DRL.

In potential follow-up work, we also propose utilizing quantities from physics simulations to enhance the modeling of player diversity and to estimate variables beyond game score. For instance, the cumulative fatigue modeling presented in CFLN*20,CXK*23 may be useful. 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.

Acknowledgements

We would like to thank Adas Slezas and Markus Laattala for their generous help with custom motion capture collection, Vivek Nair and Džiugas Ramonas (aka CyberRamen) for their guidance during the early phases of this project, and Pauli Kemppinen, Heikki Timonen, Erik Härkkönen, and Michiel van de Panne for the insightful discussion. Our game visualizations are built on AllPoland's wonderful ArcViewer project. Finally, we thank *Beat Saber*'s custom mapping community and the original authors of the maps and music featured in our project. We acknowledge the following funding sources: NSERC Postgraduate Scholarship–Doctorate (PGS-D, 567794-2022) and FCAI Doctoral Open Application Grant.

References

[ANBH23] ALEXANDERSON S., NAGY R., BESKOW J., HENTER G. E.: Listen, denoise, action! audio-driven motion synthesis with diffusion models. *ACM Transactions on Graphics (TOG) 42*, 4 (2023), 1–20. 3

[AWL*20] ABERMAN K., WENG Y., LISCHINSKI D., COHEN-OR D., CHEN B.: Unpaired motion style transfer from video to animation. *ACM Transactions On Graphics (TOG) 39*, 4 (2020), 64–1. 3

[AZL23] AO T., ZHANG Z., LIU L.: Gesturediffuclip: Gesture diffusion model with clip latents. *ACM Transactions on Graphics (TOG) 42*, 4 (2023), 1–18. 3

[BBM*25] BARQUERO G., BERTSCH N., MARRAMREDDY M., CHACÓN C., ARCADU F., RIGUAL F., HE N. S., PALMERO C., ESCALERA S., YE Y., ET AL.: From sparse signal to smooth motion: Real-time motion generation with rolling prediction models. *arXiv preprint arXiv:2504.05265* (2025). 2, 17

[Bea19] BEAT SABER MODDING GROUP: Beat saber modding group wiki. Web page, November 2019. URL: https://bsmg.wiki/. 5

[CFLN*20] CHEEMA N., FREY-LAW L. A., NADERI K., LEHTINEN J., SLUSALLEK P., HÄMÄLÄINEN P.: Predicting mid-air interaction movements and fatigue using deep reinforcement learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (2020), pp. 1–13. 3, 9, 10

[CXK*23] CHEEMA N., XU R., KIM N. H., HÄMÄLÄINEN P., GOLYANIK V., HABERMANN M., THEOBALT C., SLUSALLEK P.: Discovering fatigued movements for virtual character animation. In *SIGGRAPH Asia 2023 Conference Papers* (2023), pp. 1–12. 10

[CZS*] CHEN L., ZHAO Y., SCHNEIDER J., GAO Q., KANNALA J., SCHÖLKOPF B., PAJARINEN J., BÜCHLER D.: From midi to motion: Learning to play the piano at scale with bi-manual dexterous robot hands. In *ICRA 2025 Workshop"Handy Moves: Dexterity in Multi-Fingered Hands"Paper Submission*. 3

[DBK*20] DOSOVITSKIY A., BEYER L., KOLESNIKOV A., WEISSENBORN D., ZHAI X., UNTERTHINER T., DEHGHANI M., MINDERER M., HEIGOLD G., GELLY S., ET AL.: An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020). 4

[DCLT19] DEVLIN J., CHANG M.-W., LEE K., TOUTANOVA K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)* (2019), pp. 4171–4186. 4

[DHS*19] DU H., HERRMANN E., SPRENGER J., FISCHER K., SLUSALLEK P.: Stylistic locomotion modeling and synthesis using variational generative models. In *Proceedings of the 12th ACM SIGGRAPH Conference on Motion, Interaction and Games* (2019), pp. 1–10. 3

[DKP*23] DU Y., KIPS R., PUMAROLA A., STARKE S., THABET A., SANAKOYEU A.: 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* (2023), pp. 481–490. 2

[dWLC22] DE WOILLEMONT P. L. P., LABORY R., CORRUBLE V.: Automated play-testing through rl based human-like play-styles generation. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (2022), vol. 18, pp. 146–154. 3

[FIK*24] FISCHER F., IKKALA A., KLAR M., FLEIG A., BACHINSKI M., MURRAY-SMITH R., HÄMÄLÄINEN P., OULASVIRTA A., MÜLLER J.: Sim2vr: Towards automated biomechanical testing in vr. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (2024), pp. 1–15. 2, 3, 9, 17

[GEP*18] GUDMUNDSSON S. F., EISEN P., POROMAA E., NODET A., PURMONEN S., KOZAKOWSKI B., MEURLING R., CAO L.: Human-like playtesting with deep learning. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)* (2018), IEEE, pp. 1–8. 3

[GMMEW23] GROSPRÊTRE S., MARCEL-MILLET P., EON P., WOLLESEN B.: How exergaming with virtual reality enhances specific cognitive and visuo-motor abilities: An explorative study. e13278. URL: https://doi.org/10.1111/cogs.13278, doi:10.1111/cogs.13278. 17

[GMZ*24] GUO Y., MU Y., ZUO Y., DAI J., YAN X., LU Y., CHENG X.: Generative human motion stylization in latent space. In *Proceedings of the International Conference on Learning Representations (ICLR)* (2024). URL: https://arxiv.org/abs/2401.13505. 3

[HHKK17] HOLDEN D., HABIBIE I., KUSAJIMA I., KOMURA T.: Fast neural style transfer for motion data. *IEEE computer graphics and applications 37*, 4 (2017), 42–49. 3

[HKS17] HOLDEN D., KOMURA T., SAITO J.: Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 1–13. 2

[HSK16] HOLDEN D., SAITO J., KOMURA T.: A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (ToG) 35*, 4 (2016), 1–11. 3

[HTZ*25] HUANG X., TRUONG T., ZHANG Y., YU F., SLEIMAN J. P., HODGINS J., SREENATH K., FARSHIDIAN F.: Diffuse-cloc: Guided diffusion for physics-based character look-ahead control. *ACM Transactions on Graphics (TOG) 44*, 4 (2025), 1–12. 2, 3

[IFK*22] IKKALA A., FISCHER F., KLAR M., BACHINSKI M., FLEIG A., HOWES A., HÄMÄLÄINEN P., MÜLLER J., MURRAY-SMITH R., OULASVIRTA A.: Breathing life into biomechanical user models. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology* (2022), pp. 1–14. 2, 3, 9

[IRCvdP25] IOANNIDIS N., REDA D., COHAN S., VAN DE PANNE M.: Diffusion-based planning with learned viability filters. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 8*, 4 (2025), 1–23. 5

[JAU*21] JOKINEN J., ACHARYA A., UZAIR M., JIANG X., OULASVIRTA A.: Touchscreen typing as optimal supervisory control. In *Proceedings of the 2021 CHI conference on human factors in computing systems* (2021), pp. 1–14. 3

[JGP17] JANG E., GU S., POOLE B.: Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations* (2017). URL: https://openreview.net/forum?id=rkE3y85ee. 4

[JPL*25] JI B., PAN Y., LIU Z., TAN S., YANG X.: Sport: From zero-shot prompts to real-time motion generation. *IEEE Transactions on Visualization and Computer Graphics* (2025). 3

[JZDP25] JIAO J., ZENG R., DAI J., PAN J.: Bach: Bi-stage data-driven piano performance animation for controllable hand motion. *Computer Animation and Virtual Worlds 36*, 3 (2025), e70044. 3

[KGBH22] KRISTENSEN J. T., GUCKELSBERGER C., BURELLI P., HÄMÄLÄINEN P.: Personalized game difficulty prediction using factorization machines. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology* (2022), pp. 1–13. 8, 18

[KJS*25] KIM B., JEONG H. I., SUNG J., CHENG Y., LEE J., CHANG J. Y., CHOI S.-I., CHOI Y., SHIN S., KIM J., ET AL.: Personabooth: Personalized text-to-motion generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference* (2025), pp. 22756–22765. 3

[KVB20] KRISTENSEN J. T., VALDIVIA A., BURELLI P.: Estimating player completion rate in mobile puzzle games using reinforcement learning. In *2020 IEEE Conference on Games (CoG)* (2020), IEEE, pp. 636–639. 8

[LCK*23] LUO Z., CAO J., KITANI K., XU W., ET AL.: Perpetual humanoid control for real-time simulated avatars. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), pp. 10895–10904. 9, 18

[LZCVDP20] LING H. Y., ZINNO F., CHENG G., VAN DE PANNE M.: Character controllers using motion vaes. *ACM Transactions on Graphics (TOG) 39*, 4 (2020), 40–1. 2

[MGT*19] MAHMOOD N., GHORBANI N., TROJE N. F., PONS-MOLL G., BLACK M. J.: Amass: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF international conference on computer vision* (2019), pp. 5442–5451. 18

[MWG*21] MAKOVIYCHUK V., WAWRZYNIAK L., GUO Y., LU M., STOREY K., MACKLIN M., HOELLER D., RUDIN N., ALLSHIRE A., HANDA A., ET AL.: Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470* (2021). 18

[NGM*23] NAIR V., GUO W., MATTERN J., WANG R., O'BRIEN J. F., ROSENBERG L., SONG D.: Unique identification of 50,000+ virtual reality users from head & hand motion data. In *32nd USENIX Security Symposium (USENIX Security 23)* (2023), pp. 895–910. 7

[NGO*24] NAIR V., GUO W., O'BRIEN J. F., ROSENBERG L., SONG D.: Deep motion masking for secure, usable, and scalable real-time anonymization of ecological virtual reality motion data. In *2024 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)* (2024), IEEE, pp. 493–500. 7, 17

[NGW*23] NAIR V., GUO W., WANG R., O'BRIEN J. F., ROSENBERG L., SONG D.: 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). 2, 5

[NRO23] NAIR V., RADULOV V., O'BRIEN J. F.: Results of the 2023 census of beat saber users: Virtual reality gaming population insights and factors affecting virtual reality e-sports performance. 1–19. URL: http://graphics.berkeley.edu/papers/Nair-ROT-2023-05/, doi:10.48550/arXiv.2305.14320. 17

[OJH22] OULASVIRTA A., JOKINEN J. P., HOWES A.: Computational rationality as a theory of interaction. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (2022), pp. 1–14. 3

[Ren10] RENDLE S.: Factorization machines. In *2010 IEEE International conference on data mining* (2010), IEEE, pp. 995–1000. 8, 18

[RGR*21] ROOHI S., GUCKELSBERGER C., RELAS A., HEISKANEN H., TAKATALO J., HÄMÄLÄINEN P.: Predicting game difficulty and engagement using ai players. *Proceedings of the ACM on Human-Computer Interaction 5*, CHI PLAY (2021), 1–17. 8

[RGS*24] RAAB S., GAT I., SALA N., TEVET G., SHALEV-ARKUSHIN R., FRIED O., BERMANO A. H., COHEN-OR D.: Monkey see, monkey do: Harnessing self-attention in motion diffusion for zero-shot motion transfer. In *SIGGRAPH Asia 2024 Conference Papers* (2024), pp. 1–13. 3

[RRT*20] ROOHI S., RELAS A., TAKATALO J., HEISKANEN H., HÄMÄLÄINEN P.: Predicting game difficulty and churn without players. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play* (2020), pp. 585–593. 3, 8

[Ruh20] RUHF K. D.: *Physically Active Virtual Reality and Parkinson's Disease: A Pilot Study*. PhD thesis, Wake Forest University, 2020. 17

[SCNW19] SMITH H. J., CAO C., NEFF M., WANG Y.: Efficient neural networks for real-time motion style transfer. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 2*, 2 (2019), 1–17. 3

[SGT*25] SAWDAYEE H., GUO C., TEVET G., ZHOU B., WANG J., BERMANO A. H.: Dance like a chicken: Low-rank stylization for human motion diffusion. *arXiv preprint arXiv:2503.19557* (2025). 3

[SSH*24] STARKE S., STARKE P., HE N., KOMURA T., YE Y.: Categorical codebook matching for embodied character controllers. *ACM Transactions on Graphics (TOG) 43*, 4 (2024), 1–14. 2, 3, 4, 5, 7, 17

[SWJ*24] SHI Y., WANG J., JIANG X., LIN B., DAI B., PENG X. B.: Interactive character control with auto-regressive motion diffusion models. *ACM Transactions on Graphics (TOG) 43*, 4 (2024), 1–14. 2

[SZFJ*25] SHI D., ZHU Y., FERNANDES JUNIOR F. E., ZHAI S., OULASVIRTA A.: Simulating errors in touchscreen typing. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (2025), pp. 1–13. 8

[Tea21] TEAM B.: Beatsaver. Web page, April 2021. URL: https://beatsaver.com/. 3, 5

[Tha21] THAI T.: *The Influence Of Exergaming On Heart Rate, Perceived Exertion, Motivation To Exercise, And Time Spent Exercising*. PhD thesis, Salem University, 2021. 17

[TLH*25] TRUONG T. E., LIAO Q., HUANG X., TEVET G., LIU C. K., SREENATH K.: Beyondmimic: From motion tracking to versatile humanoid control via guided diffusion. *arXiv preprint arXiv:2508.08241* (2025). 3

[TRC*24] TEVET G., RAAB S., COHAN S., REDA D., LUO Z., PENG X. B., BERMANO A. H., VAN DE PANNE M.: Closd: Closing the loop between simulation and diffusion for multi-task character control. *arXiv preprint arXiv:2410.03441* (2024). 2

[TRG*23] TEVET G., RAAB S., GORDON B., SHAFIR Y., COHEN-OR D., BERMANO A. H.: Human motion diffusion model. In *The Eleventh International Conference on Learning Representations* (2023). URL: https://openreview.net/forum?id=SJ1kSyO2jwu. 2

[TZCvdP22] TAO T., ZHAN X., CHEN Z., VAN DE PANNE M.: Style-erd: Responsive and coherent online motion style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 6593–6603. 3

[VSP*17] VASWANI A., SHAZEER N., PARMAR N., USZKOREIT J., JONES L., GOMEZ A. N., KAISER Ł., POLOSUKHIN I.: Attention is all you need. *Advances in neural information processing systems 30* (2017). 2, 4

[WCHW21] WEN Y.-H., CHEN C.-C., HSU K.-H., WANG J.-C.: Autoregressive stylized motion synthesis with generative flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 14504–14513. URL: https://arxiv.org/abs/2105.01963. 3

[Wil92] WILCOXON F.: Individual comparisons by ranking methods. In *Breakthroughs in statistics: Methodology and distribution*. Springer, 1992, pp. 196–202. 6

[WXS*24] WANG R., XU P., SHI H., SCHUMANN E., LIU C. K.: Fürelise: Capturing and physically synthesizing hand motion of piano performance. In *SIGGRAPH Asia 2024 Conference Papers* (2024), pp. 1–11. 3

[XLP*25] XIAO L., LU S., PI H., FAN K., PAN L., ZHOU Y., FENG Z., ZHOU X., PENG S., WANG J.: Motionstreamer: Streaming motion generation via diffusion-based autoregressive model in causal latent space. *arXiv preprint arXiv:2503.15451* (2025). 3

[XSYP25] XU M., SHI Y., YIN K., PENG X. B.: Parc: Physics-based augmentation with reinforcement learning for character controllers. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers* (2025), pp. 1–11. 2

[XTS*23] XIE Z., TSENG J., STARKE S., VAN DE PANNE M., LIU C. K.: Hierarchical planning and control for box loco-manipulation. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 6*, 3 (2023), 1–18. 2

[XWCH15] XIA S., WANG C., CHAI J., HODGINS J.: Realtime style transfer for unlabeled heterogeneous human motion. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 1–10. 3

[YLHX22] YE Y., LIU L., HU L., XIA S.: Neural3points: Learning to generate physically realistic full-body motion for virtual reality users. In *Computer Graphics Forum* (2022), vol. 41, Wiley Online Library, pp. 183–194. 2

[YM16] YUMER M. E., MITRA N. J.: Spectral style transfer for human motion between independent actions. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 1–8. 3

[ZBL*19] ZHOU Y., BARNES C., LU J., YANG J., LI H.: On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 5745–5753. 5

[ZSKS18] ZHANG H., STARKE S., KOMURA T., SAITO J.: Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 1–11. 2

[ZXJ*24] ZHONG L., XIE Y., JAMPANI V., SUN D., JIANG H.: Smoodi: Stylized motion diffusion model. In *European Conference on Computer Vision* (2024), Springer, pp. 405–421. 3

Figure 10: *The physics-based version of Robo-Saber plays colored notes. The red and blue notes are correctly cut in sequence, with matching saber colors and directions. The dotted notes can be hit from any direction.*

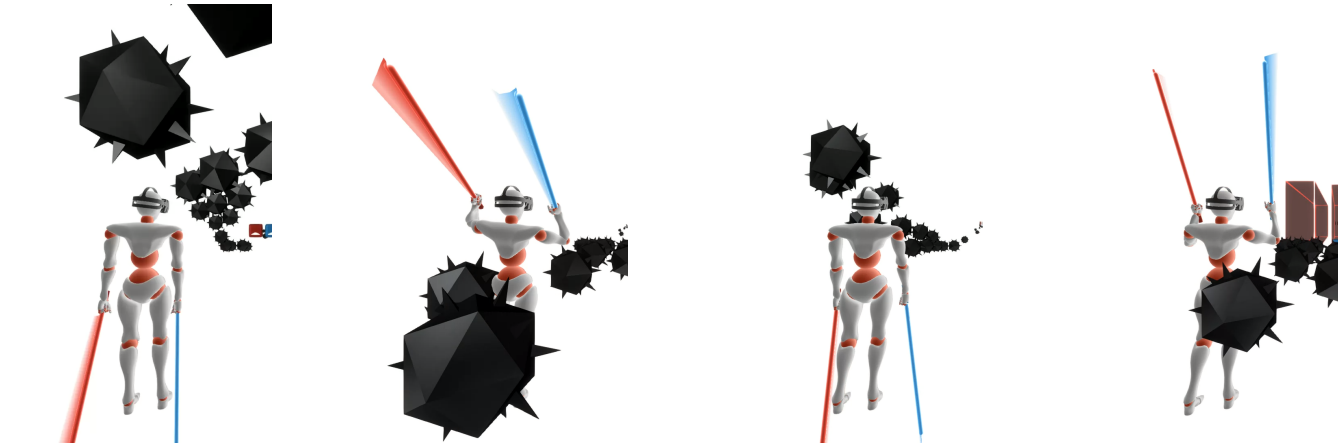

Figure 11: *Robo-Saber avoids a long sequence of bomb notes by moving its hands up and down and orienting the sabers away from them.*

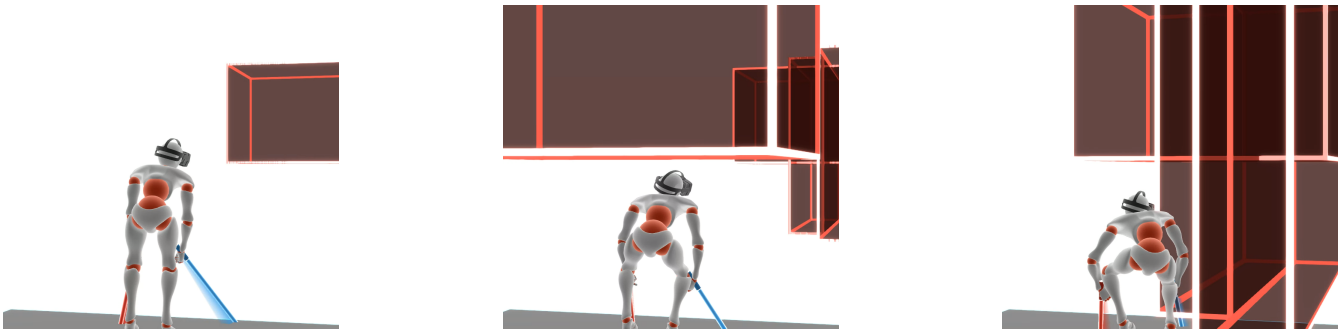

Figure 12: *Robo-Saber avoids obstacles (red boxes) by ducking to lower its head and swaying away from them. Robo-Saber is first in performing articulated whole-body gameplay movements for VR.*

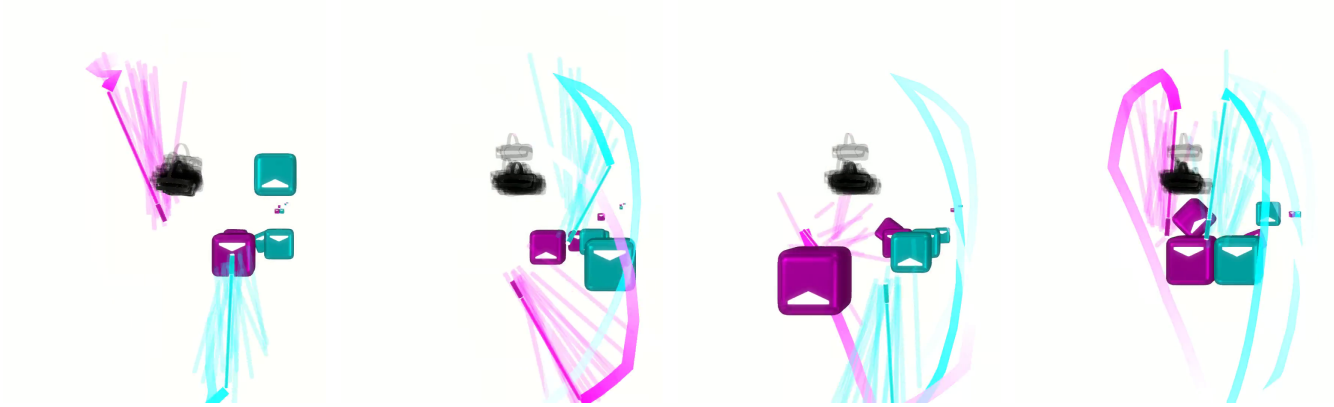

Figure 13: *Robo-Saber samples viable* 3*p trajectory samples (shown as semi-transparent headset and sabers) for the same game state input, from which the most optimal one is selected. Many possible trajectories for cutting the colored notes appear, and after evaluation, the one with the largest pre- and post-swing angle is selected.*

Figure 14: ***Top:*** *Conditioning Robo-Saber on different* $\mathbf{x}^{:N_{ref}}$ *induces variations in behavior. Some correct and incorrect saber movements emerge, reflecting skill level variations present in conditioning signals.* ***Bottom:*** *Varying the random seed instead while conditioned on the same reference, the movement variation is less noticeable.*

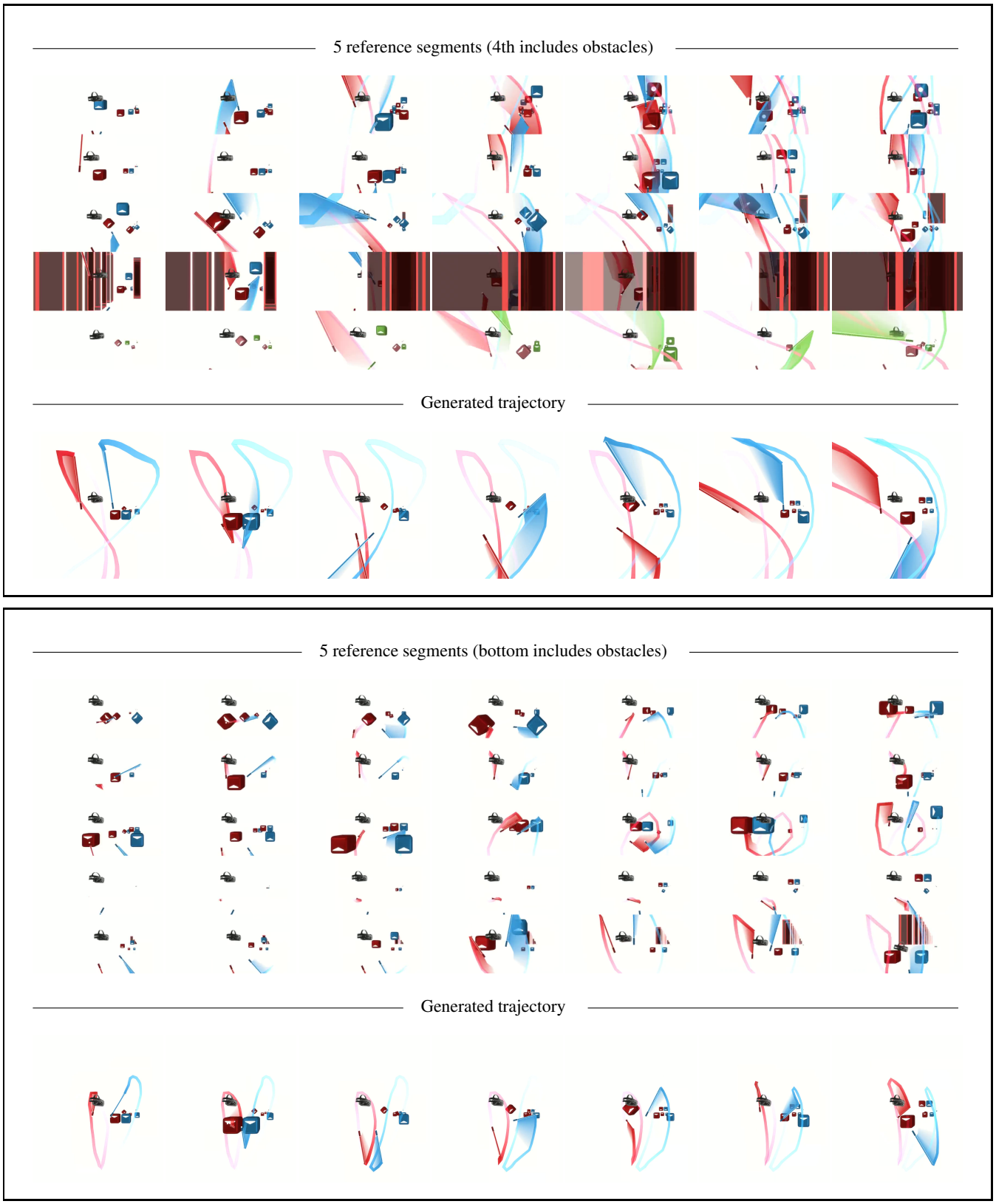

Figure 15: *Two image sequences, each comparing input reference segments (first 5 rows) and the resulting 3p generation (last row).* ***Top:*** *Conditioning on an expert's skilled gameplay results in finessed and confident movements, featuring anticipation and fast swing speed.* ***Bottom:*** *Conditioning on a novice's reference segments yields hesitant and cautious positioning. The two are evaluated on the same map.*

Appendix A: Beat Saber: A Benchmark VR Application

Beat Saber is a skill-intensive VR rhythm game in which players swing virtual sabers to cut colored notes with the correct hand and direction, while avoiding bombs with the sabers and obstacles with the head. The game has been established as a prominent benchmark VR application in various computational domains, including data anonymization (*e.g.*, NGO*24), motion generation (*e.g.*, SSH*24, BBM*25), and *in silico* user modeling (*e.g.*, FIK*24), as well as in non-computational fields such as health research [Tha21, GMMEW23] and neurological therapy [Ruh20].

Computational challenges for Beat Saber. Computationally, *Beat Saber* challenges agents to map complex spatio-temporal input to body movements: players must precisely cut many dozens of notes, each varying in color, position, and timing, in rapid succession. While the objective is simple, the game does not explicitly instruct on the whole-body coordination required, giving rise to highly diverse movement styles across the player population. Moreover, *Beat Saber*'s space of object configuration is huge: with 2 note colors, 9 note directions, and 12 grid positions, a sequence of 10 colored notes would already yield 1e23 possible combinations.

Beat Saber is difficult to curate. Beat Saber's 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* [NRO23]. 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 design 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 the same map, based on the players' 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 player model in conjunction with collaborative filtering to perform personalized score prediction, showing promise in this direction.

Appendix B: Data Quality Control and Preprocessing

We apply the following quality control (QC) measures to BOXRR-23, which consists of $3p$ movement sequences, each corresponding to a player's attempt on a map. We first remove every replay with missing map information or score record. As *Beat Saber* features many *mods* and *game modes* that significantly alter the game's behavior, we simplify our task by removing records that include mods or non-standard game modes. Additionally, records with excessive jitter or no movement are excluded, as well as records with incorrect left-handed indication (*i.e.*, the record metadata shows left-handed but the gameplay is actually right-handed). As the BeatSaver database is concurrently receiving updates without versioning, we simply remove maps that have received updates since the original $3p$ data were collected.

For the pose data QC, we standardize all players' height to 1.5044m at rest, matching that of the standard humanoid character for the physics-based tracking experiments in Sec. 5. As *Beat Saber*'s game grid's height is adjusted based on the player's height, we apply a height offset to the headset and handhelds' coordinates. As issues in hardware calibration can cause an incorrect global offset in the recorded $3p$ data, we center the *xyz* coordinates by subtracting the median value from each dimension. Finally, we correct for any coordinate-shifting anomalies during gameplay by detecting clusters of headset positions using kernel density estimation and removing any resulting modes.

Appendix C: Network Architectures for Miscelleneous Models

Here, we outline the architectures used for the Gumbel-Softmax VAE, oracle classifier, and the Transformer baseline for PSP. All of the models share the common trait of being sequence-based models, and for the interest of consistency, we elect to use Transformer as the backbone for all of the models. The Transformer hyperparameters, such as number of layers and attention heads, remain the same across all models. For the most thorough details on our implementation, we refer the reader to the official codebase linked to the project webpage: `https://robo-saber.github.io/`.

Gumbel-Softmax VAE. The length-T pose sequence $\mathbf{p}_{t:t+T}$ is further processed into a length-$3T$ sequence of 9-dimensional vectors, as to model the dense relationship between body parts across time. We use multi-headed self-attention with GELU activation, and then aggregate the final layer's latent sequence with an MLP head, flattening the whole sequence into a vector and finally encoding it into $\mathbf{z}_t^{3\mathrm{p}}$.

Oracle player classifier. The oracle receives 6 gameplay segments and identifies which held-out player produced them. The oracle player classifier is very similar to the style encoder $\mathcal{E}^{\mathrm{style}}$, which takes $\mathbf{x}^{:N_{\mathrm{ref}}}$ and $\mathbf{p}_{t_i^*:t_i^*+T}$ to produce $\mathbf{z}^{:N_{\mathrm{ref}}}$. Instead of an embedding, the classifier uses an MLP head to produce player classification logits. We stay consistent with the use of all-zero sentinel token for predicting these logits. To compute the logits for a generated trajectory, we take the mean across 6 predicted logits, produced from feeding randomly-chosen gameplay segments from the generated trajectory to the oracle.

Appendix D: TorchSaber: Implementation Details

In Sec. 3.3, we introduced TorchSaber (TS), the GPU-accelerated and simplified *Beat Saber* simulator, which is incorporated into the sampling process and evaluation of our method. TS is based on a vectorized implementation of the slab method for collision detection, utilizing PyTorch for GPU-accelerated tensor operations. Given two consecutive frames, we produce 1.2-meter-long line segments for the left and right sabers. We place 5 keypoints along each line segment at regular intervals. Representing the displacement between frames allows us to compute 5 rays, which describe the position changes of these keypoints.

We compute collision masks between displacement rays and note boxes using the slab method. A collision is registered if any displacement ray penetrates a box boundary. The head-to-obstacle

collisions are computed simply as a point-box collision, using the normal vector of the obstacle box's faces and the head's displacement with respect to the obstacle box's vertices.

Geometries. For computing collisions, TS utilizes the following geometries, based on available *Beat Saber* documentation. First, the headset is modeled as a point in 3D. The sabers are modeled as line segments, each 1.2 meters long. Colored and bomb notes are modeled as cubes sized $(0.5, 0.5, 0.5)$. The boxes for the obstacles are scaled according to their durations, width, and height specified in the map data. Each object is instantiated at the point determined by its appearance in the map data. For each object, the x-position (forward) is calculated from its relative timestamp, multiplied by the note jump distance specified by the map. 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 H. We find $1.05 - \frac{H}{2}$ to be an appropriate offset. Emulating the original game's behavior, we set the objects' x-offset so they arrive just in front of the player at the targeted beat. We allow objects to persist in view up to 0.1 seconds after the beat, so that they exit when they are behind the player. These parameters are optimized in the direction that maximizes the match between the computed TS and the recorded replay scores of real players.

Appendix E: Reward Function Details

Here, we define the reward terms that appear in Sec. 3.3. The TorchSaber score r_{TS} is as described in Sec. 3, 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 as 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 "omnidirectional" notes, any cut is declared a good cut. Simplifying *Beat Saber*'s intricate scoring criteria, we 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 (pre-swing) and after (post-swing) the good cut are at least -100 and 60 degrees, respectively.

The total TS score for a map is computed by taking the average score across colored notes appearing. For the sake of simplicity and ease of implementation, we excluded combos and cut angles from consideration, which still retained a strong positive correlation between TS scores and real *Beat Saber* scores from BOXRR-23.

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 bomb-saber collision penalty r_{Bomb} is similarly calculated. At each length-T segment, given n_{Bombs} that appear, we compute $n_{\text{BombCollision}}$, the total number of appeared bombs that have collided with any of the sabers. Then r_{Bombs} is simply calculated as $\frac{n_{\text{BombCollision}}}{n_{\text{Bombs}}}$.

Appendix F: Factorization Machines

We follow KGBH22 and employ factorization machines (FMs, Ren10) for CF. FMs learn embeddings for each player and map, along with a simple linear model, such that the score for the player-map pair is predicted as:

$$\hat{y}^{\text{player} \cdot \text{map}} = \left(\mathbf{z}^{\text{player}} \cdot \mathbf{z}^{\text{map}}\right) + w_0 + w^{\text{player}} + w^{\text{map}} \tag{14}$$

where $y^{\text{player} \cdot \text{map}} \in \mathbb{R}$ is the predicted score, $\mathbf{z}^{\text{player}}$, $\mathbf{z}^{\text{map}}$ are the learned embeddings for the player and the map, and w_0, w^{player}, and w^{map} are the global bias and player/map-specific weights of the linear model, respectively.

Simply, the embeddings and the linear model are optimized by a gradient-based optimizer on the squared error of the score:

$$\mathcal{L}_{\text{FM}} = \left(y^{\text{player} \cdot \text{map}} - \hat{y}^{\text{player} \cdot \text{map}}\right)^2 \tag{15}$$

For more details of FMs, we refer the reader to Ren10.

Appendix G: Physics-basd Tracking Implementation

Built within Isaac Gym [MWG*21], PHC [LCK*23] produces full-body joint actuations that align the robot's head and hands in both position and rotation with the input $3p$ pose at each timestep. Using the usual DRL notation, the tracking controller π outputs the actions $\mathbf{a}_t$ at each timestep t as:

$$\mathbf{a}_t = \pi(\cdot \mid \mathbf{s}_t, \mathbf{p}_t) \tag{16}$$

where $\mathbf{s}_t$ is the state of the simulated character and $\mathbf{p}_t$ is the target $3p$ pose. The entire $3p$ trajectory may be produced in advance and subsequently provided to the tracking controller as a target sequence, thereby fully decoupling kinematic planning from physics-based control. As depicted in Fig. 1, the system's overall behavior of synthesizing movements based on game state observations is unchanged. Accordingly, Robo-Saber operates as an open-loop, whole-body robotic agent for VR gameplay.

We first tested the pre-trained $3p$ tracking variant of the PHC controller without modification. We found that it could not maintain the character's balance with *Beat Saber*-specific movements; therefore, we fine-tuned the tracking policy using 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 the original AMASS [MGT*19] training data and new training data in order to prevent the pretrained controller from forgetting its tracking abilities. The sabers are attached to the PHC character's hands, aligned with the direction of the fingers.

Appendix H: Table of Hyperparameters

Table 1 discloses every hyperparameter involved in producing this work.

Symbol	Description	Value
T	Number of frames per $3p$ motion chunk	16
-	Interval for interpolating the T frames	4
n	Length of latent sequence for each domain (colored notes, bomb notes, obstacles)	20
s	The default lookahead, in seconds, used in training	2.0
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	10
$\lambda_{\text{Obstacle}}$	Reward weight for head-obstacle collision penalty	10
-	Batch size for kinematic $3p$ generator training	128
-	$3p$ generator model AdamW learning rate	5e-5
-	Number of Transformer encoder layers	4
-	Number of Transformer attention heads	4
$\lvert\mathbf{z}^{\text{player}}\rvert, \lvert\mathbf{z}^{\text{map}}\rvert$	Embedding size for factorization machine (FM)	16
-	Batch size for FM training	512
-	Embedding & linear weight dropout rate for FM training	0.5
-	FM optimizer AdamW weight decay rate	1e-1
-	FM optimizer AdamW learning rate	3e-4

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