ParliRobo: Participant Lightweight AI Robots for Massively Multiplayer Online Games (MMOGs)

Jianwei Zheng
Tsinghua University

Changnan Xiao
ByteDance Inc.

Mingliang Li
Tsinghua University

Zhenhua Li*
Tsinghua University

Feng Qian
University of Southern California

Wei Liu
Tsinghua University

Xudong Wu
Tsinghua University

ABSTRACT

Recent years have witnessed the profound influence of AI technologies on computer gaming. While grandmaster-level AI robots have largely come true for complex games based on heavy back-end support, in practice many game developers crave for participant AI robots (PARs) that behave like average-level humans with inexpensive infrastructures. Unfortunately, to date there has not been a satisfactory solution that registers large-scale use. In this work, we attempt to develop practical PARs (dubbed ParliRobo) showing acceptably humanoid behaviors with well affordable infrastructures under a challenging scenario—a 3D-FPS (first-person shooter) mobile MMOG with real-time interaction requirements. Based on comprehensive real-world explorations, we eventually enable our attempt through a novel "transform and polish" methodology. It achieves ultralight implementations of the core system components by non-intuitive yet principled approaches, and meanwhile carefully fixes the probable side effect incurred on user perceptions. Evaluation results from large-scale deployment indicate the close resemblance (96% on average) in biofidelity metrics between ParliRobo and human players; moreover, in 73% mini Turing tests ParliRobo cannot be distinguished from human players.

CCS CONCEPTS

• **Information systems** → **Multimedia information systems**; **Massively multiplayer online games**.

KEYWORDS

Participant AI robots; deep reinforcement learning; (approximate) vision reconstruction; model pruning

ACM Reference Format:
Jianwei Zheng, Changnan Xiao, Mingliang Li, Zhenhua Li, Feng Qian, Wei Liu, and Xudong Wu. 2023. ParliRobo: Participant Lightweight AI Robots for Massively Multiplayer Online Games (MMOGs). In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23), October 29–November 3, 2023, Ottawa, ON, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3581783.3613764

*Corresponding author.

MM '23, October 29–November 3, 2023, Ottawa, ON, Canada.

ACM ISBN 979-8-4007-0108-5/23/10…$15.00
https://doi.org/10.1145/3581783.3613764

1 INTRODUCTION

AI technologies are increasingly used in computer gaming, mostly for two goals: winning or participation. The former has recently been achieved by applying modern AI techniques, *esp.,* deep reinforcement learning (DRL), to various simple and complex games, typically based on heavyweight back-end support. The resulting AI robots, *e.g.,* AlphaGo [12], AlphaStar [56], and OpenAI Five [2], manifest AI's power of defeating its human rivals in chess, real-time strategy (RTS), first-person shooter (FPS), and other game genres.

For the latter, there exist satisfactory participant AI robots (abbreviated as PARs) in simple games like Gomoku [63] and Super Mario [52]. However, we have not seen such robots in complex games (*e.g.,* Legends [40] and PUBG [26]) that involve various kinds of delicate interactions. In fact, in complex games existing PARs are generally considered "stiff, stupid, and ridiculous" by human players [17], even when built on expensive infrastructures. This could severely harm the user experience, and sometimes even discourage the participation of human players [36].

According to our interviews with a number of developers for complex games, the vast majority of them crave for PARs that behave like average-level humans rather than grandmaster-level AI robots, since the former can better act as non-player characters (NPCs) for promoting human players' immersive engagement and attracting new players. Meanwhile, the developers wish to limit the infrastructure cost to an affordable level for supporting the large-scale use, *i.e.,* they also want the PARs to be lightweight.

To understand real-world challenges, we attempt to develop lightweight PARs with acceptably humanoid behaviors for a popular 3D-FPS mobile MMOG (dubbed 8-Game) with 230K users as of 03/2022 and real-time interaction requirements. We choose this highly challenging scenario so that our solution (if feasible) can be reused in many other demanding games. In general, we leverage mainstream DRL techniques to enable this attempt, since the core problem of developing PARs for complex games is how to let them properly react (*i.e.,* take actions like human players) based on high-dimensional inputs under complex environments, which conforms to the basic philosophy of DRL.

As illustrated in Figure 1, the desired system is implemented in three major steps. First, we take sampled images of the environments and players visible to PARs as the state input (5 frames per second). Then, we employ pre-trained deep neural networks

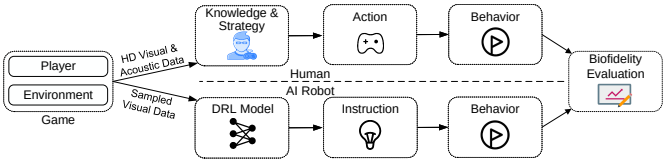

Figure 1: Basic workflow for the development and evaluation of participant DRL robots.

(DNNs) to extract the features of images, and utilize mainstream DRL models [39, 46] (which are trained with real human players' gaming traces) to generate action instructions for PARs. Finally, we compare the exhibited behaviors of PARs and closed-beta-test (CBT) human players to evaluate the biofidelity. During the process, however, we encounter three-fold significant difficulties.

- *Server Bottleneck.* Since the PARs are supposed to run smoothly on numerous heterogeneous mobile devices without interfering the already very "heavy" app of 8-Game, we strive to minimize the client-side overhead. This requires the server to play a centralized, "almighty" role—it collects image data from each robot, infers the next action for each robot, and then sends the action instruction(s) to each robot. Processing the visual features of image data at scale will incur considerable overheads.
- *Hearing Loss.* Modern 3D engines require audio production to be accompanied by visual rendering [14]. If the server wants to generate and leverage auditory information, it has to accomplish visual rendering in advance with tremendous overhead. In consequence, state-of-the-art solutions (*e.g.,* AlphaStar) choose to avoid both acoustic and visual rendering at the server. This is reasonable for most games (*e.g.,* RTS games) but unsuitable for 8-Game, as players need hearing to perceive the enemies behind.
- *Late Instructions.* The DRL inference delay for an action averages at 104 ms, which can cause an action instruction to reach a client behind time (and thus the PAR can be slow in response). In wired networks where the average delay is only 20 ms, the total interaction delay stays around 124 ms so very few instructions come late. But in mobile networks where the delay is considerably longer (avg. 70 ms) [31], up to 14.3% instructions arrive late.

In essence, we note that these difficulties are all attributed to intuitive (and thus heavyweight) implementations of the core system components. First, to emulate human vision, we take images as the state input, incurring tremendous overhead of visual data processing on the server. Second, the tight coupling of audio and visual data rendering in modern 3D engines hinders our straightforward integration of PARs' hearing abilities. Third, to generate human-like actions, we employ complex DRL models with many layers and parameters, substantially increasing the inference delay.

Addressing these issues is not easy, as conventional optimizations with common wisdom prove to be ineffective. To alleviate server bottlenecks, a natural method is to downscale input image resolutions, but this hinders PARs' ability to identify game elements due to the loss of critical visual features. Moreover, directly extracting audio features from 3D engines involves heavy convolution operations, and brings noises like background sounds. We also try to reduce inference delay via model quantization [64] (which downcasts model parameters from floats to integers), but this impairs PARs' decision-making abilities and makes them look "stiff".

Given that the difficulties cannot be resolved by intuitive implementations or optimizations, we decide to explore non-intuitive yet more principled approaches. Eventually, we successfully build the desirable system, ParliRobo, through a novel "transform and polish" methodology, which substantially lightens our original implementations in an indirect, nearly equivalent manner and meanwhile carefully fixes the probable side effect incurred on user perceptions.

Specifically, from the perspective of *visual feature representation*, instead of directly using images, we utilize ultralight structured information that only contains the properties (*e.g.,* locations and quantities) of crucial game elements (*e.g.,* players and props) to eliminate server bottlenecks. Inevitably, this causes certain visual detail losses, so we conduct fast approximate vision reconstruction by conformal mapping [43] and bilinear extrapolations/interpolations [57].

From the perspective of *auditory perception*, instead of accurately rendering or roughly extracting the audio data (both of which are computation-intensive), we synthesize directional vision to compensate a PAR's hearing upon specific game events like footsteps and gunshots. However, the artificial (vision-based) hearing cannot rationally reflect the sound attenuation; hence, we make it realistic through linear attenuation-based acoustic energy transformation.

From the perspective of *online inference*, we cut excessive "trivial" neurons off the DRL model using the lottery ticket hypothesis [10] to reduce inference delay. Unfortunately, the pruned model becomes weak in generalizability, *i.e.,* hard to accommodate new game functions; thus, we devise flexible model migration to retrieve neurons crucial to PARs' decision making with regard to new functions.

According to our experiment results with large-scale deployment and small-scale mini Turing tests, ParliRobo shows acceptably humanoid behaviors in 8-Game. With a well-established IRB and informed user consent, we record real-time gaming behavioral data (*i.e.,* shoots, hits, jumps, crouches, moving distances, and props gathered) of 96,812 opt-in users for two months (Oct.–Nov. 2021). In terms of most biofidelity metrics, ParliRobo exhibits a close resemblance (96% on average) to human players. Meanwhile, the average DRL inference delay for an action is reduced from 104 ms to 30.4 ms, so the portion of late instructions decreases from 14.3% to trivial (0.78%). Additionally, 34 users participate in the mini Turing tests, in the majority (73%) of which ParliRobo cannot be distinguished from human players by these users. In all experiments, the maximum number of concurrent PARs is 13,200, which can be well supported by four commodity servers; each server is equipped with an Intel Xeon 64-core CPU@2.40GHz and 128-GB DDR memory.

Finally, the code and data involved in this study have been released in part at https://ParliRobo.github.io/.

2 APPLYING MAINSTREAM DRL TECHNIQUES TO 8-GAME

2.1 Brief Introduction to 8-Game

In today's online gaming landscape, MMOGs have formed a $54 billion market in 2022 [51]. To provide a generic framework for realizing PARs in MMOGs, we target the most challenging MMOGs genre that features action-packed real-time 3D interactions, by

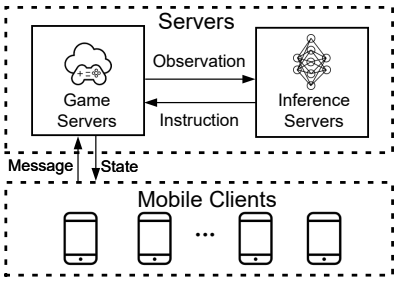

Figure 2: Architecture of our intuitive implementation.

collaborating with 8-Game, a popular 3D-FPS mobile MMOG with ~230,000 users as of 03/2022. Each round of 8-Game is a 30-player "death match" lasting 15 minutes, where players strive to gain as many scores as possible by eliminating others or picking up money. More details of 8-Game are described in Appendix A.1

2.2 Intuitive Implementation

Deep reinforcement learning (DRL) has overwhelmed other AI techniques in both simple and complex games based on heavy back-end support [2, 12, 38, 56]. Mainstream DRL techniques used in AI robots can be basically classified into two categories: value-based [39] and policy-based [46], according to their different status update mechanisms. The value-based DRL defines the reward function of actions as returns, and tends to choose actions that can maximize the reward sum. In contrast, the policy-based DRL defines the update policy and assigns a probability distribution to all possible actions.

To understand real-world challenges of developing PARs for MMOGs, we make a preliminary attempt by applying these two DRL techniques to 8-Game (Figure 2). Game servers manage general game events, such as maintaining player states and syncing them with clients. We sample rendered images from a PAR's view every 0.2 seconds, aligning with the typical human visual reaction time [24] and the common practice for AI robots in MMOGs [2]. Then, the sampled images are sent to inference servers as state inputs, which then generate action instructions for PARs.

We adopt VGG16 [49] to extract visual features from sampled images. VGG16 uses small convolution kernels to replace large ones, and thus incurs less overhead than alternatives like AlexNet [27]. We further employ both Deep Q-learning Network (DQN) [39] and Proximal Policy Optimization (PPO) [46], the representative value-based and policy-based DRL models, to infer action instructions for PARs, respectively. They are trained with human players' gaming traces in eight Nvidia V100 GPUs, and deployed on inference servers. The two solutions are denoted as PAR-DQN and PAR-PPO.

However, our evaluation of PAR-DQN and PAR-PPO during the close beta test (CBT) reveals their three shortcomings. The first stems from the server-side bottleneck, which results in unsatisfactory *service throughput*. As shown in Figure 3, a typical commodity server (with an Intel Xeon 64-core CPU@2.40GHz and 128-GB DDR memory) can only support no more than 1,000 concurrent PARs, with high CPU utilization being the primary cause.

Second, PARs lack hearing ability, preventing them from reacting to in-game sounds. To measure players' responsive ability to sound, we adopt the *action saliency distance* [28], defined as the cosine distance (ranging from [0, 2]) of two consecutive action vectors that represent player actions. A higher value of the distance indicates a more significant action change. We observe that when human players hear the sound events (*i.e.,* gunshots and explosions), the corresponding action saliency distance increases significantly. In contrast, PARs cannot perceive surrounding sound events with less significant action changes. This is due to the tight coupling of audio and visual rendering in modern 3D engines, causing most MMOGs to avoid using audio to prevent heavyweight visual rendering [14].

The third is the slow reaction of PARs, which makes them look inactive and stiff. To demystify this, we meticulously analyze the *interaction delay* between the server and client (*i.e.,* from the start of model inference to the time when instructions reach user devices). Figure 2 indicates that interaction delay includes inference delay and network delay. As shown in Figure 4, the network delay ranges from 49 ms to 100 ms, while the inference delay is longer (avg. 103 ms). Thus, the interaction delay often exceeds the 200-ms instruction cycle, making PARs perform actions less than 5 times/s.

2.3 Conventional Optimization

These shortcomings motivate us to optimize intuitive implementations from three perspectives, *i.e.,* input scale, hearing compensation, and model complexity. We denote PAR-PPO and PAR-DQN with all subsequent optimizations as PAR-PPO+ and PAR-DQN+.

Image Downscaling. We reduce the resolution of high-dimensional input images to alleviate the server-side overhead. Figure 5 shows that after downscaling the resolution from 1024×512 to 64×64, we achieve a significant improvement in service throughput; the concurrency of PAR-DQN and PAR-PPO with image downscaling (denoted as PAR-DQN$_1$ and PAR-PPO$_1$) improves from 831 (PARs per server) to 1,573, and 872 to 1,681, respectively. However, they experience a substantial biofidelity degradation, particularly in the ability to identify game elements. Figure 6 shows the distribution of players' shootings when enemies are visible (<50 m). PAR-DQN and PAR-PPO with image downscaling perform nearly half shoot actions (avg. 9.8 and 12.6) than PAR-DQN, PAR-PPO, and human players (avg. 16.9, 21.2, and 59.9), indicating that image downscaling causes visual feature losses.

Audio Rendering. To utilize audio information without incurring additional visual rendering overhead, we decouple audio rendering from visual rendering in 3D engines. We then extract the mel-frequency spectra [41] of audio, and feed them into a 1D convolutional neural network to derive features. The extracted audio and visual features are concatenated as input for DRL models (see Figure 8). Unfortunately, the additional convolution operations involved in audio feature extraction incur extra 22%–37% inference delay, resulting in 26% more late instructions.

Model Quantization. To reduce inference delay, we employ model quantization [64], a widely-used technique to accelerate neural network inference by downcasting model parameters from floats to integers. As shown in Figure 9, we add two functional layers to original models — a quantizer before it and a de-quantizer after it. The former converts inputs and weights of original models from 32-bit floats to 8-bit integers. After the DRL processing, the de-quantizer converts the integer outputs back to floats. The PARs with model quantization are denoted as PAR-DQN$_3$ and PAR-PPO$_3$. In this way,

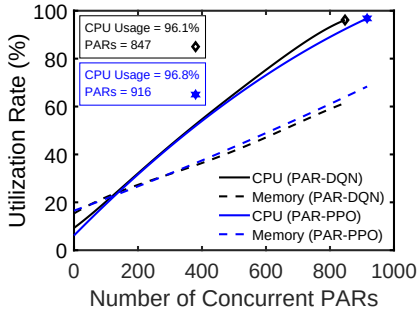

Figure 3: CPU and memory utilization of PAR-DQN and PAR-PPO.

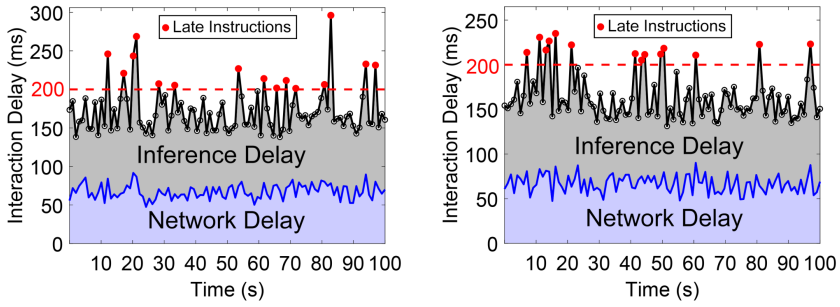

Figure 4: Interaction delay of PAR-DQN and PAR-PPO. Red ones correspond to late instructions whose delay exceeds the 200-ms instruction cycle.

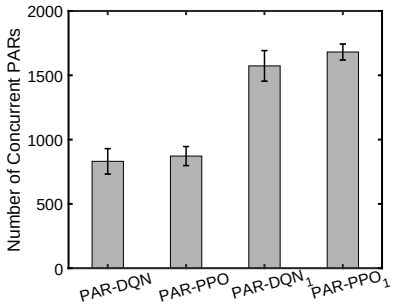

Figure 5: Number of concurrent PARs supported by a single server.

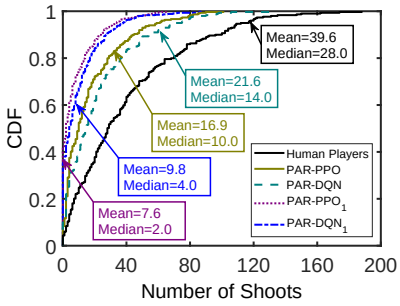

Figure 6: Distribution of shoot numbers for PARs and humans.

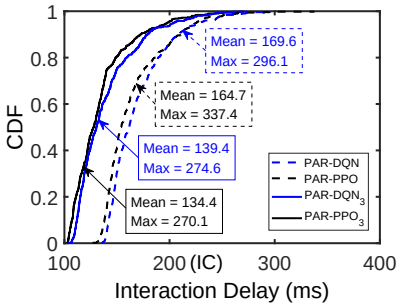

Figure 7: Distribution of interaction delay for PARs.

the average inference delays of PAR-DQN$_3$ and PAR-PPO$_3$ are reduced by ~20% compared with the original model (see Figure 7). However, their reactions are still inaccurate from a common user's perspective (*e.g.,* inaccurate shooting). Delving deeper, we find that such quantization causes a loss of precision in parameters [42], incurring significant drops in models' reward sum (~32 %).

3 DESIGN OF PARLIROBO

In this section, we present ParliRobo, a practical PAR solution consisting of three key designs (see Figure 10). Since PPO outperform DQN in §2.2 and §2.3, we adopt PPO as the final inference model.

3.1 Crucial Visual Representation (CVR)

In §2.3, while image downscaling significantly improves service throughput, it negatively impacts the biofidelity of PARs due to the loss of crucial visual features. Thus, CVR directly capture the meta-information of essential game elements from the 3D engine.

Specifically, we extract and maintain important properties (*e.g.,* coordinates and quantities) of all crucial game elements (*e.g.,* buildings, props, and players). We then serialize them into lightweight and structured messages (*i.e.,* type/value pairs) using Google Protocol Buffers (GPB) [13]. To ensure these messages act as the visual state inputs that can be fed into the DRL model for training or inference, we further adopt the below conformal mapping method [43].

Visual-Aware Feature Extraction with Conformal Mapping. When playing 3D-FPS MMOGs, human players can only see objects within their field of vision. As shown in Figure 12, this field of vision can be considered a 2D circular sector area from the game map's top view. Thus, we only need to extract the visual features of crucial game elements locating within the PARs' vision field.

To provide PARs with human-like visual features in MMOGs, we consider these key aspects of human vision simultaneously: distance, angles, and orientation. To this end, we leverage conformal mapping [43] which performs coordinate system transformation to ensure that distant elements appear sparser than closer ones while preserving angles and orientations among them. As illustrated in Figure 12, suppose a player is located at (x_0, y_0) on the global map with a constant positive viewing angle $\bar{\theta}$, heading towards the direction θ_0, and $E_1(r, -\theta)$ is an element in the circular sector area, where $\theta > 0$. We us the following 2D complex function [37] to complete the conformal mapping: $M(z) = -\frac{\ln(z)}{i \cdot \bar{\theta}}$, where $z = r \cdot e^{i \cdot (-\theta)}$ is the point $(r, -\theta)$ under the complex coordinate system. Then we have: $M(z) = u + v \cdot i = \frac{\theta}{\bar{\theta}} + \frac{\ln(r)}{\bar{\theta}} \cdot i$.

Thus, we have the mapped point $(u, v) = \left(\frac{\theta}{\bar{\theta}}, \frac{\ln(r)}{\bar{\theta}}\right)$ in the projected vision field for any $(r, -\theta)$ in the original circular sector area. Conversely, for any (u, v) in the projected vision field, it corresponds to the $(r, -\theta) = \left(e^{v \cdot \bar{\theta}}, u \cdot \bar{\theta}\right)$ in the original circular sector area. Since $e^{v \cdot \bar{\theta}}$ is strictly convex, as long as we equally divide the projected vision field into a 64×64 (an empirical size based on our long-term practice) feature matrix, we can map the crucial game elements near the human player to the matrix points intensively, and map those far from the human player sparsely. Also, the angles

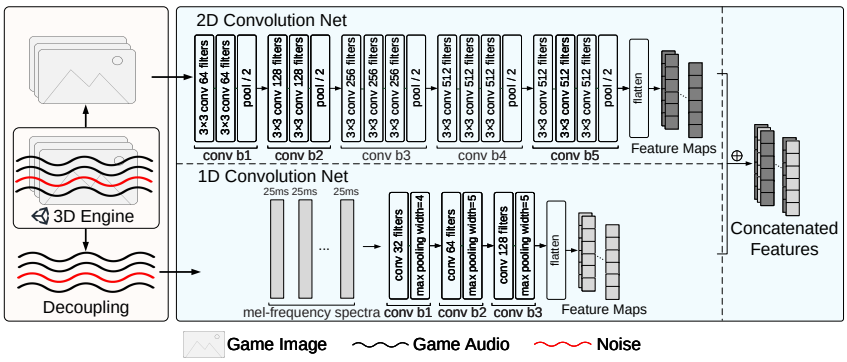

Figure 8: The workflow and detailed neural network structure where visual and audio data are decoupled in 3D engines, and their features are extracted by CNN.

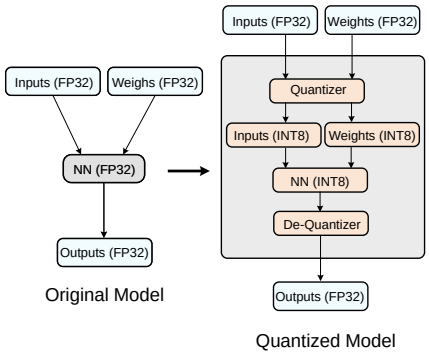

Figure 9: Compressing DRL models through model quantization.

and orientations among game elements after the matrix projection are naturally preserved by the conformal mapping. Therefore, we finally reconstruct the human vision and preserve its both key properties in terms of the distance and angles/orientations.

Visual Compensation with Bilinear Extra-/Interpolation. Although the conformal mapping can reconstruct human vision, it may cause feature losses. Since the projected feature matrix is a 64×64 grid where the coordinates of each two neighboring grid points are not continuous, not all elements in the vision field can be precisely mapped to the points in the feature matrix, which may cause PARs to perform actions (*e.g.,* shooting) inaccurately.

We thus use the bilinear extrapolation/interpolation [57] to compensate for the lost visual features caused by the discrete sampling. Compared to other alternatives (*e.g.,* linear-based [4] and bicubic-based [6]), the bilinear-based can better balance the complexity and effectiveness [15]. As Figure 13 shows, assuming that $\tilde{F}(x, y)$ is the original visual feature at $(x, y) \in \mathbb{R}^2$ of the global map, we divide the original global map into a 2D mesh grid, and extrapolate $\tilde{F}$ to its four neighboring integral grid points, which is formalized as:

$$\begin{cases} F(i, j) = (i + 1 - x) \cdot (j + 1 - y) \cdot \tilde{F}(x, y), \\ F(i + 1, j) = (x - i) \cdot (j + 1 - y) \cdot \tilde{F}(x, y), \\ F(i, j + 1) = (i + 1 - x) \cdot (y - i) \cdot \tilde{F}(x, y), \\ F(i + 1, j + 1) = (x - i) \cdot (y - i) \cdot \tilde{F}(x, y), \end{cases} \tag{1}$$

where $i = \lfloor x \rfloor$, $j = \lfloor y \rfloor$, and $F(i, j)$ is the extrapolated visual features at the integral point (i, j).

Then, for (u, v) in the projected matrix, we identify its original point $(r, -\theta)$ in the vision field based on $M(z) = -\frac{\ln(z)}{i \cdot \theta}$ where $z = r \cdot e^{i \cdot (-\theta)}$, and its corresponding point (x_{uv}, y_{uv}) in the global map is: $(x_{uv}, y_{uv}) = (r \cos(\theta_0 - \theta) + x_0, r \sin(\theta_0 - \theta) + y_0)$. We perform bilinear interpolation to obtain the compensated feature F_c at (x_{uv}, y_{uv}), which is also the feature at (u, v) in the projected feature matrix:

$$\begin{aligned} F_c(x_{uv}, y_{uv}) = {} & (i + 1 - x_{uv}) \cdot (j + 1 - y_{uv}) \cdot F(i, j) \\ & + (x_{uv} - i) \cdot (j + 1 - y_{uv}) \cdot F(i + 1, j) \\ & + (i + 1 - x_{uv}) \cdot (y_{uv} - j) \cdot F(i, j + 1) \\ & + (x_{uv} - i) \cdot (y_{uv} - i) \cdot F(i + 1, j + 1), \end{aligned} \tag{2}$$

where $i = \lfloor x_{uv} \rfloor$, $j = \lfloor y_{uv} \rfloor$. In this way, we fill the feature points of projected matrix with interpolated visual features.

3.2 Realistic Hearing Compensation (RHC)

RHC is designed to address the dilemma: hearing is crucial to PARs in 3D-FPS MMOGs, but accurately rendering and roughly extraction of audio are computation-intensive and incur heavy overhead. To overcome this, we leverage synthesized directional vision to compensate for PARs' hearing. We first number all types of sound events (*e.g.,* gunshots and explosions), and then mark the region (*i.e.,* the corresponding 2D integer mesh grid in §3.1) where they can propagate. However, this does not reflect the attenuation characteristics of sounds. Consequently, PARs can perceive sound events but struggle to locate them, leading them to search aimlessly instead of purposefully turning towards the sound source like human players.

To address this, we further propose a linear attenuation-based acoustic energy transformation scheme that enables PARs to locate the sound source. This is based on the facts that humans use the sound intensity difference to perceive the sound orientation and the sound travels with linear attenuation in 8-Game (*cf.* §2.1). Let A denote the attenuation function of the sound, and $A(r, t)$ represent the sound intensity. $A(r, t)$ is related to the sound type t and inversely proportional to the distance r. That is: $A(r, t) = \begin{cases} -k_t \cdot r + b_t & \text{if } r \le R \\ 0 & \text{if } r > R \end{cases}$, where k_t denotes the sound attenuation coefficient of sound type t, and b_t denotes the original sound intensity, and R is the maximum sound propagation range. Then we integrate A to calculate the cumulative sound intensity S of the sound from (x_t, y_t) on each grid cell D that PARs can perceive: $S = \iint_D A\left(\sqrt{(x - x_t)^2 + (y - y_t)^2}, t\right) dxdy, \ (x, y) \in D.$

An area with a higher accumulated sound intensity value means that it is closer to sound sources, and vice versa. Finally, we append S to the corresponding feature point of the projected matrix.

3.3 Flexible Model Pruning (FMP)

To avoid the impairment in PARs' decision-making abilities caused by model quantization (*cf.* §2.3), we resort to an advanced model compression technique, the lottery ticket hypothesis (LTH) [10], which trims excessive "trivial" neurons from the DRL model to reduce inference delay. LTH reveals that dense neural networks contain less complex subnetworks (the so-called "winning tickets") that can achieve comparable performance to the original network.

As shown in Figure 11, we first randomly initialize the DRL model's parameters and run a training session to update the weights

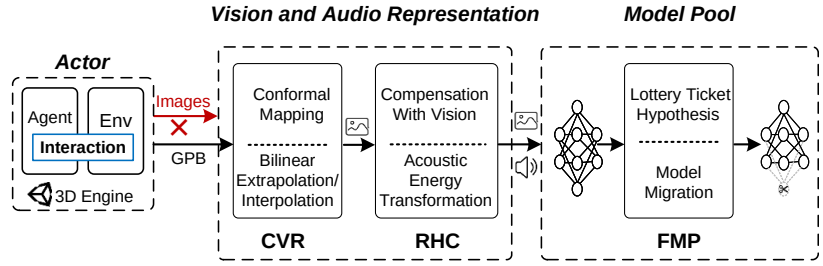

Figure 10: Architectural overview of ParliRobo.

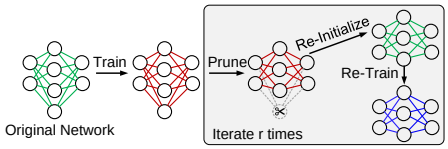

Figure 11: Leveraging lottery ticket hypothesis to prune DRL models. Lines with the same color indicate that the same parameters are used in neural networks.

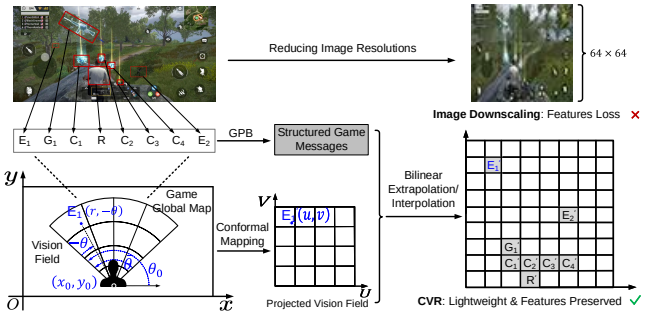

Figure 12: Comparison of image downscaling and crucial visual representation (CVR).

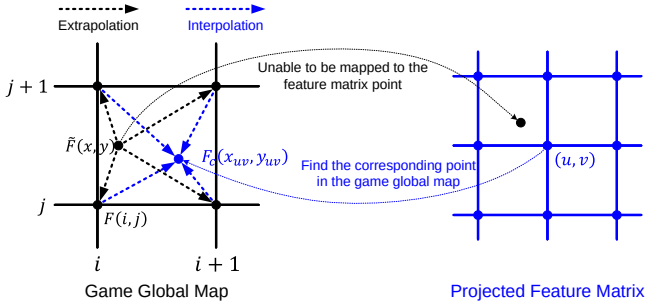

Figure 13: Compensating for the visual features using bilinear extrapolation/interpolation.

of these parameters. Afterwards, we cut off the last $\sqrt[r]{p}\%$ of the parameters (*i.e.,* setting values to zero) with the lowest-magnitude weights, and re-initialize the rest parameters with the original weights of the unpruned model. Further, we re-train the pruned model. After r iterations, we cut off $p\%$ parameters of the original model, and obtain the final "winning ticket".

However, when 8-Game updates, the DRL model needs to be re-pruned and re-trained. Unfortunately, the pruning process is computationally expensive and has to be run thoroughly for different models, since LTH is iterative and magnitude-based, resulting in weak generalizability for the pruned model. To address this, instead of redesigning and re-pruning DRL models, FMP strategically replicates (or drops) and reorders atomic units of the original network, whose corresponding source winning ticket can be stretched (or squeezed) into a target winning ticket for a new model [8]. This scheme is derived from our long-term practical experience: since these updates are usually minor and the input/output dimensions do not change, original and new models share similar architectures.

Meanwhile, the convolutional layers and the associated pooling layer in Figure 8 are the natural choice for atomic units. Since the output is connected to each neural layer directly or indirectly, and the relationships between them are determined [50, 62], we conduct a correlation analysis between atomic units and corresponding game function outputs to decide which atomic units need to be replicated (or dropped) when adding (or removing) game functions.

4 EVALUATION

We deploy the trained DRL models on four inference servers to generate actions for PARs, and employ eight game servers to conduct general game events (*e.g.,* maintaining player states and synchronizing them with clients); each is equipped with an Intel Xeon 64-core CPU @2.40GHz, and 128-GB DDR4 memory (see Figure 2).

4.1 Evaluation on Components

This part evaluates ParliRobo in three aspects by incrementally enabling one component at a time: *service throughput*, *acoustic response rate*, and *interaction delay*. More details of these evaluation metrics are described in Appendix A.2

Crucial Visual Representation (CVR). When CVR is enabled, as Figure 14a shows, the service throughput of PAR-PPO with CVR (⑤, 2,918) is ~3.4 times of that of PAR-PPO (①, 872), and is 73.6% more than that of PAR-PPO with image downscaling (②, 1,681). This indicates that CVR significantly reduces the overhead on the server by transforming input states from heavyweight images (adopted by ① and ②) to ultralight structured representation.

In addition, by comparing ①, ②, and ⑤ in Figure 14b, we notice that CVR and image downscaling have little effect on PARs' acoustic response rates, since neither of them involves any auditory information. Meanwhile, the results of ①, ②, and ⑤ (164.7 ms *vs.* 145.5 ms *vs.* 123.9 ms) in Figure 14c indicate that image downscaling and CVR also effectively decrease the average interaction delay of PAR-PPO, and thus reduce late instructions (interaction delay>200 ms) from 14.3% to 6.4% and 3.7%, respectively.

Realistic Hearing Compensation (RHC). While keeping CVR active, we incrementally enable RHC. As depicted in Figure 14b, by strategically compensating PARs' auditory perception through synthesized directional vision and realistic acoustic energy transformation, PAR-PPO with CVR and RHC (⑥) achieves a great increase in the acoustic response rate from 15.7% to 82.1% compared to PAR-PPO with CVR (⑤). Further, PARs boosted by RHC enjoy a higher acoustic response rate (⑥, 82.1%) than those with audio rendering (③, 74%), and is closest to that of human players (⑧, 92%), which indicates that RHC is more effective than audio rendering for compensating PARs' hearing since it introduces no noise.

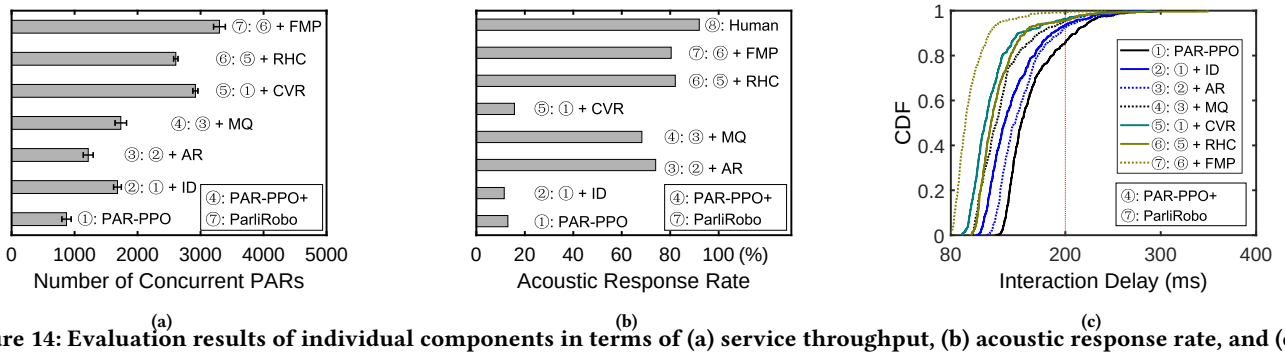

Figure 14: Evaluation results of individual components in terms of (a) service throughput, (b) acoustic response rate, and (c) interaction delay. ID, AR, and MQ denote the image downscaling, audio rendering, and model quantization of PAR-PPO+. CVR, RHC, and FMP represent individual components of ParliRobo. Interaction delay ≥ 200 ms leads to late instructions.

Meanwhile, as plotted in Figure 14a, the service throughput of ⑥ slightly decreases from 2,918 to 2,607 compared to that of ⑤. Even so, ⑥ still enjoys a higher service throughput (2,607) than that of ① (872) and ③ (1,217), since RHC is more lightweight than audio rendering. Moreover, Figure 14c shows that the interaction delay of ⑥ (avg. 130.5 ms) is higher than that of ⑤ (avg. 123.9 ms) with an increase in late instructions from 3.7% to 4.5%. Such performance degradations is due to the extra computation overhead introduced by RHC for compensating PARs' auditory perception.

Flexible Model Pruning (FMP). Figure 14c demonstrates that when FMP is further enabled, ParliRobo (⑦) owns the lowest interaction delay (80.3 ms on average), and thus greatly reduces the proportion of high interaction delays (> 200 ms), resulting in a reduction of late instructions from 14.3% (①, PAR-PPO) to 0.78%. Compared with PAR-PPO with CVR and RHC (⑥), ⑦ significantly reduce the inference delay by cutting excessive "trivial" neurons off the complex DRL model. Moreover, as illustrated in Figure 14a, the service throughput of ⑦ greatly increases from 2,607 to 3,307, compared with that of ⑥. This improvement indicates that FMP can also alleviate the computational overhead on the server.

Nevertheless, Figure 14b shows that FMP incurs a performance degradation on acoustic response rate by comparing ⑥ and ⑦ (from 82.2% to 80.4%), and a similar trend also exists in model quantization by comparing ③ and ④ (from 74% to 68.3%). This is because FMP and model quantization either cut "trivial" neurons off the complex DRL model or downcast their precision, which weakens the inference ability of DRL models. Fortunately, such degradation (1.8%) caused by FMP is moderate and acceptable given its benefits in service throughputs and interaction delays.

4.2 Individual Biofidelity

Individual biofidelity metrics (IBM) measure the resemblance between PARs and human players in these key micro-level behaviors: shooting (IBM_s), hitting (IBM_h), jumping (IBM_j), crouching (IBM_c), moving distances (IBM_m), and props gathered (IBM_p), covering most gaming behaviors in 3D-FPS MMOGs. More details of individual biofidelity metrics are depicted in Appendix A.2.

Figure 15 shows that ParliRobo exhibits more resemblances with human players compared to PAR-PPO and PAR-PPO+ across all micro-level behaviors. The distribution of ParliRobo's each behavior is closest to that of human players compared to PAR-PPO and PAR-PPO+, with the lowest value of the corresponding *IBM*. For instance, Figure 15b shows that the distribution of ParliRobo's hitting frequency basically fits that of human players (IBM_h=0.023), which reveals that ParliRobo owns the ability (comparable to that of human players) to accurately locate and hit enemies.

Table 1: Mini Turing test results of human players and PAR systems. The percentages in bold indicate the TPRs.

Solution	Labeled as Humans	Labeled as Robots
Human	2958 (87%)	442 (13%)
PAR-PPO	1326 **(39%)**	2074 (61%)
PAR-PPO+	1904 **(56%)**	1496 (44%)
ParliRobo	2482 **(73%)**	918 (27%)

4.3 Mini Turing Test

However, the micro-level metrics are unable to well reflect the overall biofidelity of PARs [19]. Thus, we additionally employ the mini Turing test [18, 53] to evaluate the macro biofidelity of PARs perceived by real users and define the *mini Turing test pass rate* (*TPR*) as $\frac{\#\ of\ the\ tests\ where\ PARs\ are\ labeled\ as\ humans}{\#\ of\ the\ total\ tests}$. To obtain *TPR*, we conduct another IRB-approved user trial by employing 34 users under their consent. More details of the crowdsourcing study are stated in Appendix A.2. Table 1 reveals that the *TPR* of ParliRobo (73%) is comparable to that of human players (87%), and significantly higher than that of PAR-PPO (39%) and PAR-PPO+ (56%). Note that 13% of human players are labeled as PARs, which is probably because they are unskilled and thus look like a robot.

5 RELATED WORK

AI Techniques for Game Bots. In the past decade, the research community has conducted a plethora of studies on AI robots, mainly for three typical genres of games (*i.e.,* board, card, and MMOGs), through DRL [21, 33, 47] and/or searching tree-based methods [9, 22, 23], while the majority of them aim to develop winning AI robots and provide small-scale services [61]. In board games, for example, DeepMind leverages Monte Carlo Tree Search [7] together with DRL [1] and self-play [16] to develop the AlphaGo series [12, 20, 48] that defeat professional players in Go. In card games, related studies [5, 65] introduce Counterfactual Regret Minimization (CFR) and leverage DNN to develop AI robots in Poker games.

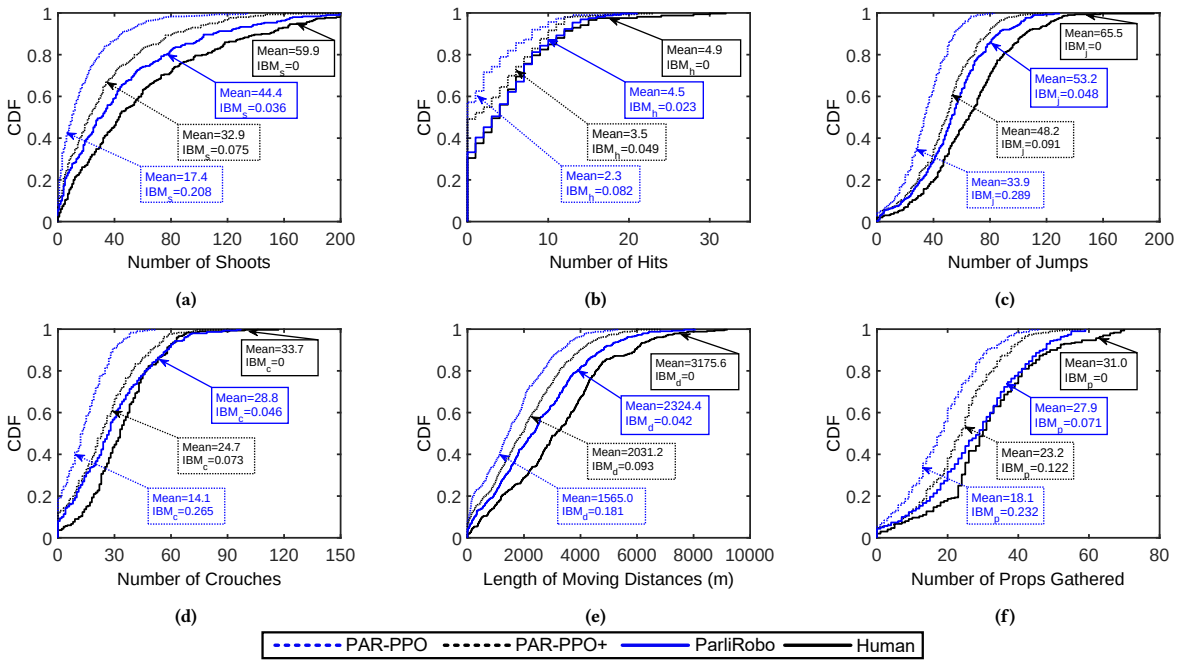

Figure 15: Performance of PAR-PPO, PAR-PPO+, ParliRobo, and human players on micro-level behaviors. *IBM* **(ranging from 0 to 1) denotes the corresponding divergence of each behavior; a smaller value indicates a better human-robot resemblance.**

Other studies [3, 26, 54] focus on developing AI robots for MMOGs which involves more complex environments and real-time interactions than board and card games. AlphaStar [56] adopts supervised learning to initialize robot parameters and multi-agent RL to improve performance, and outperforms 99.8% human players in StarCraft [3]. Open AI Five [2], the AI robot of Dota2 [54], leverages distributed self-play DRL to boost the training process, making it the first AI system that defeat the world champions of the game.

Our study distinguishes itself from the above studies in three aspects. First, instead of focusing on defeating human players, we make efforts to provide generic framework to realize participant AI robots for MMOGs. Second, ParliRobo is lightweight yet effective owing to our novel "transform and polish" methodology that substantially lighten the key components of the PAR system. Last, we demonstrate the practicality of our AI robots by real-world deployment at scale, which serves a great number of human players.

Evaluation Metrics. Players' ever-growing expectations of gaming experience, together with the increasing complexity of games, have been driving both academia and industry to define suitable metrics that better evaluate AI robot performance as well as player experience [45, 55]. For winning AI robots [2, 12, 30, 48, 56, 59, 60], the winning rate is an adequate metric to evaluate their performance. Nevertheless, it is not suitable for evaluating PARs whose purpose is to provide average-level participation service for human players at scale, rather than defeat human players [25, 35].

In the literature, there is a limited number of studies on PARs' evaluation metrics, yet all of them perform evaluations at small scale and/or concerning specific aspects. Lample et al. [29] leverage the behavioral data (*i.e.,* the number of objects, kills, deaths, and suicides) as metrics to evaluate PARs' resemblances with humans. Lin et al. [32] propose the playstyle metric from observations and actions, aiming to measure how close two players' behaviors are. Osborn et al. [44] define the Gamalyzer metric based on refinements to edit distance [34] of play traces in a game-independent way. In comparison, we not only evaluate the biofidelity of PARs from both micro and macro perspective, but also consider the infrastructure costs under large-scale deployment. We hope that these metrics can benefit future researches on PARs.

6 CONCLUSION

This paper presents the design, implementation, real-world deployment, and comprehensive evaluation of ParliRobo, a lightweight yet effective participant AI robot (PAR) system for MMOGs. We reveal the obstacles of developing satisfactory PARs at scale in the context of a popular 3D-FPS mobile MMOG, and demonstrate that they cannot be well addressed through intuitive implementations and optimizations. Instead, ParliRobo overcomes them via a non-conventional "transform and polish" methodology that substantially lightens the key components of a PAR system by approximate transformation, and meanwhile carefully repairs the incurred side effects on user perceptions. ParliRobo is now being used to serve millions of human players with acceptable biofidelity and back-end cost. We believe that many lessons and experiences we have learned from building ParliRobo could also benefit the research on PARs for other MMOGs as well as related AI robot applications.

ACKNOWLEDGMENTS

This work is supported in part by National Key Research and Development Program of China under grant 2022YFB4500703, National Natural Science Foundation of China under grants 61902211 and 62202266, China Postdoctoral Science Foundation under grant 2022M721831, and Microsoft Research Asia under grant 100336949.

REFERENCES

[1] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. Deep Reinforcement Learning: A Brief Survey. *IEEE Signal Processing Magazine* 34, 6 (2017), 26–38.

[2] Christopher Berner, Greg Brockman, Brooke Chan, et al. 2019. Dota 2 with Large Scale Deep Reinforcement Learning. arXiv:1912.06680

[3] Blizzard Entertainment, Inc. 2022. StarCraft II Official Game Site. https://starcraft2.com/.

[4] Thierry Blu, Philippe Thévenaz, and Michael Unser. 2004. Linear Interpolation Revitalized. *IEEE Transactions on Image Processing* 13, 5 (2004), 710–719.

[5] Noam Brown, Adam Lerer, Sam Gross, and Tuomas Sandholm. 2019. Deep Counterfactual Regret Minimization. In *Proc. of PMLR ICML*. 793–802.

[6] Ralph E Carlson and Frederick N Fritsch. 1985. Monotone Piecewise Bicubic Interpolation. *SIAM J. Numer. Anal.* 22, 2 (1985), 386–400.

[7] Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. 2008. Monte-Carlo Tree Search: A New Framework for Game AI. In *Proc. of AAAI*, Vol. 4. 216–217.

[8] Xiaohan Chen, Yu Cheng, Shuohang Wang, Zhe Gan, Jingjing Liu, and Zhangyang Wang. 2021. The Elastic Lottery Ticket Hypothesis. In *Proc. of NeurIPS*, Vol. 34. 26609–26621.

[9] Michele Colledanchise, Ramviyas Parasuraman, and Petter Ögren. 2018. Learning of Behavior Trees for Autonomous Agents. *IEEE Transactions on Games* 11, 2 (2018), 183–189.

[10] Jonathan Frankle and Michael Carbin. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *Proc. of ICLR*.

[11] Bent Fuglede and Flemming Topsoe. 2004. Jensen-Shannon Divergence and Hilbert Space Embedding. In *Proc. of IEEE ISIT*. 31.

[12] Gibney, Elizabeth. 2016. Google AI Algorithm Masters Ancient Game of Go. *Nature News* 529, 7587 (2016), 445.

[13] Google. 2021. Official Website of Google Protocol Buffers. https://developers.google.com/protocol-buffers.

[14] David Grelaud, Nicolas Bonneel, Michael Wimmer, et al. 2009. Efficient and Practical Audio-Visual Rendering for Games Using Crossmodal Perception. In *Proc. of ACM I3D*. 177–182.

[15] Dianyuan Han. 2013. Comparison of Commonly Used Image Interpolation Methods. In *Proc. of ICCSEE*. 1556–1559.

[16] Johannes Heinrich and David Silver. 2016. Deep Reinforcement Learning from Self-Play in Imperfect-Information Games. *arXiv preprint arXiv:1603.01121* (2016).

[17] Henry Ewins. 2020. Like Animals, Video Game AI Is Stupidly Intelligent. https://www.eurogamer.net/articles/2020-01-09-like-animals-video-game-ai-is-stupidly-intelligent.

[18] Philip Hingston. 2009. A Turing Test for Computer Game Bots. *IEEE Transactions on Computational Intelligence and AI in Games* 1, 3 (2009), 169–186.

[19] Philip Hingston. 2010. A New Design for a Turing Test for Bots. In *Proc. of IEEE CIG*. 345–350.

[20] Sean D Holcomb, William K Porter, Shaun V Ault, et al. 2018. Overview on Deepmind and Its Alphago Zero AI. In *Proc. of ACM ICBDE*. 67–71.

[21] Shiyu Huang, Hang Su, Jun Zhu, and Ting Chen. 2019. Combo-Action: Training Agent for FPS Game with Auxiliary Tasks. In *Proc. of AAAI*, Vol. 33. 954–961.

[22] Matteo Iovino, Edvards Scukins, Jonathan Styrud, Petter Ögren, and Christian Smith. 2020. A Survey of Behavior Trees in Robotics and AI. *arXiv preprint arXiv:2005.05842* (2020).

[23] Matteo Iovino, Jonathan Styrud, Pietro Falco, and Christian Smith. 2021. Learning Behavior Trees with Genetic Programming in Unpredictable Environments. In *Proc. of IEEE ICRA*. 4591–4597.

[24] Aditya Jain, Ramta Bansal, Avnish Kumar, and KD Singh. 2015. A Comparative Study of Visual and Auditory Reaction Times on the Basis of Gender and Physical Activity Levels of Medical First Year Students. *International Journal of Applied and Basic Medical Research* 5, 2 (2015), 124.

[25] Aaron Khoo and Robert Zubek. 2002. Applying Inexpensive AI Techniques to Computer Games. *IEEE Intelligent Systems* 17, 4 (2002), 48–53.

[26] KRAFTON, Inc. 2022. PUBG Mobile. https://www.pubgmobile.com/.

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet Classification with Deep Convolutional Neural Networks. In *Proc. of NeurIPS*, Vol. 25.

[28] Shaofan Lai, Wei-Shi Zheng, Jian-Fang Hu, and Jianguo Zhang. 2017. Global-Local Temporal Saliency Action Prediction. *IEEE Transactions on Image Processing* 27, 5 (2017), 2272–2285.

[29] Guillaume Lample and Devendra Singh Chaplot. 2017. Playing FPS Games with Deep Reinforcement Learning. In *Proc.of AAAI*.

[30] Junjie Li, Sotetsu Koyamada, Qiwei Ye, et al. 2020. Suphx: Mastering Mahjong with Deep Reinforcement Learning. *arXiv preprint arXiv:2003.13590* (2020).

[31] Zhenhua Li, Yafei Dai, Guihai Chen, and Yunhao Liu. 2023. *Content Distribution for Mobile Internet: A Cloud-Based Approach, Second Edition*. Springer Nature Press.

[32] Chiu-Chou Lin, Wei-Chen Chiu, and I-Chen Wu. 2021. An Unsupervised Video Game Playstyle Metric via State Discretization. In *Proc. of PMLR UAI*. 215–224.

[33] Tianyu Liu, Zijie Zheng, Hongchang Li, et al. 2019. Playing Card-Based RTS Games with Deep Reinforcement Learning. In *Proc. of IJCAI*. 4540–4546.

[34] Andres Marzal and Enrique Vidal. 1993. Computation of Normalized Edit Distance and Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15, 9 (1993), 926–932.

[35] Maja J Matarić. 2019. Human-Machine and Human-Robot Interaction for Long-Term User Engagement and Behavior Change. In *Proc. of ACM MobiCom*. 1–2.

[36] Michael Matuschek. 2022. Using Adaptive AI to Improve the Gaming Experience. https://www.mouser.com/blog/using-adaptive-ai-improve-gaming-experience.

[37] MIPAV. 2020. Transform: Conformal Mapping Algorithms. https://mipav.cit.nih.gov/pubwiki/index.php/Transform:_Conformal_Mapping_Algorithms.

[38] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. 2013. Playing Atari with Deep Reinforcement Learning. *arXiv preprint arXiv:1312.5602* (2013).

[39] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. 2015. Human-Level Control through Deep Reinforcement Learning. *Nature* 518, 7540 (2015), 529–533.

[40] Moonton. 2022. Mobile Legends: Bang Bang. https://m.mobilelegends.com/.

[41] Ghulam Muhammad, Yousef A Alotaibi, Mansour Alsulaiman, and Mohammad Nurul Huda. 2010. Environment Recognition Using Selected MPEG-7 Audio Features and Mel-Frequency Cepstral Coefficients. In *Proc. of IEEE ICDT*. 11–16.

[42] Yury Nahshan, Brian Chmiel, Chaim Baskin, et al. 2021. Loss Aware Post-training Quantization. *Machine Learning* 110, 11 (2021), 3245–3262.

[43] Zeev Nehari. 2012. *Conformal Mapping*. Courier Corporation.

[44] Joseph C Osborn and Michael Mateas. 2014. A Game-Independent Play Trace Dissimilarity Metric. In *Proc. of FDG*.

[45] Mark Owen Riedl and Alexander Zook. 2013. AI for Game Production. In *Proc. of IEEE CIG*. 1–8.

[46] John Schulman, Filip Wolski, Prafulla Dhariwal, et al. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[47] Kun Shao, Zhentao Tang, Yuanheng Zhu, et al. 2019. A Survey of Deep Reinforcement Learning in Video Games. *arXiv preprint arXiv:1912.10944* (2019).

[48] David Silver, Thomas Hubert, Julian Schrittwieser, et al. 2018. A General Reinforcement Learning Algorithm That Masters Chess, Shogi, and Go through Self-Play. *Science* 362, 6419 (2018), 1140–1144.

[49] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-scale Image Recognition. *arXiv preprint arXiv:1409.1556* (2014).

[50] Jost Tobias Springenberg, Alexey Dosovitskiy, et al. 2015. Striving for Simplicity: The All Convolutional Net. In *Proc. of ICLR*.

[51] TheExpressWire. 2023. 2023-2029 Massive Multiplayer Online (MMO) Games Market Size Detailed Report with Sales and Revenue Analysis | Research by Absolute Reports. https://www.digitaljournal.com/pr/news/2023-2029-massive-multiplayer-online-mmo-games-market-size-detailed-report-with-sales-and-revenue-analysis-research-by-absolute-reports.

[52] Julian Togelius, Sergey Karakovskiy, Jan Koutník, and Jurgen Schmidhuber. 2009. Super Mario Evolution. In *Proc. of IEEE CIG*. 156–161.

[53] Alan M Turing. 2012. Computing Machinery and Intelligence (1950). *The Essential Turing: the Ideas That Gave Birth to the Computer Age* (2012), 433–464.

[54] Valve Corporation. 2022. Dota2 Official Game Site. https://www.dota2.com/home.

[55] Michael Van Lent, John Laird, Josh Buckman, et al. 1999. Intelligent Agents in Computer Games. In *Proc. of AAAI*. 929–930.

[56] Vinyals, Oriol and Babuschkin, Igor and Chung, Junyoung and others. 2019. Alphastar: Mastering the Real-Time Strategy Game Starcraft II. *DeepMind Blog* (2019), 2.

[57] Muhammad Abdul Wahab. 2017. Interpolation and Extrapolation. In *Proc. Topics Syst. Eng. Winter Term*, Vol. 17. 1–6.

[58] Wikipedia. 2022. Kullback Leibler divergence. https://en.wikipedia.org/wiki/Kullback-Leibler_divergence.

[59] Deheng Ye, Guibin Chen, Wen Zhang, et al. 2020. Towards Playing Full MOBA Games with Deep Reinforcement Learning. In *Proc. of NeurIPS*, Vol. 33. 621–632.

[60] Deheng Ye, Zhao Liu, Mingfei Sun, et al. 2020. Mastering Complex Control in MOBA Games with Deep Reinforcement Learning. In *Proc. of AAAI*, Vol. 34. 6672–6679.

[61] Sule Yildirim and Sindre Berg Stene. 2010. *A Survey on the Need and Use of AI in Game Agents*. InTech.

[62] Matthew D Zeiler and Rob Fergus. 2014. Visualizing and Understanding Convolutional Networks. In *Proc. of Springer ECCV*. 818–833.

[63] Dongbin Zhao, Zhen Zhang, and Yujie Dai. 2012. Self-Teaching Adaptive Dynamic Programming for Gomoku. *Neurocomputing* 78, 1 (2012), 23–29.

[64] Yiren Zhou, Seyed-Mohsen Moosavi-Dezfooli, Ngai-Man Cheung, and Pascal Frossard. 2018. Adaptive Quantization for Deep Neural Network. In *Proc. of AAAI*.

[65] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. 2007. Regret Minimization in Games with Incomplete Information. In *Proc. of NeurIPS*, Vol. 20.

A APPENDIX

A.1 Introduction to 8-Game

In today's online gaming landscape, MMOGs have formed a $54 billion market in 2022 [51]. In this work, to provide a generic framework for realizing PARs in MMOGs, we target the most challenging MMOGs genre that involves action-packed real-time 3D interactions, by collaborating with 8-Game, a popular 3D-FPS mobile MMOG with ~230,000 users as of 03/2022. Each round of 8-Game is a 30-player "death match" that lasts for 15 minutes, where each player's target is to gain as many scores as possible by killing others or picking up money. When the time of a game round runs out, players are ranked in descending order by their scores, and the top three players win the game. Although gathering as many as 30 players per round makes 8-Game exciting, the operators find that during off-peak hours, the players often need to wait for a considerable time to start a round. This promotes our collaboration with 8-Game to develop PARs.

In detail, the battlefield of 8-Game is a 4 $km \times 4$ km square island where players can walk, jump, crawl, gather props (*e.g.,* money, weapons, and medicine bags), and fight with others. Players experience the game through the protagonist's eyes with a 110-degree wide view angle and a 250 m visual scope. Additionally, players can hear sound events (*e.g.,* footsteps and gunshots) within a range of 50 m. Note that in 8-Game, sound travels with linear attenuation (*cf.* §3.2) and is not affected by obstructions. During a round, players have three chances to revive; each time players are revived, they are initiated with the same hit point (HP), which will drop by a certain amount when players get attacked at a specific body part. Once a player "dies" (*i.e.,* the HP reaches zero), half of their money will be transferred to the one killing them.

A.2 Evaluation Metrics

We adopt the following metrics to comprehensively evaluate PARs in terms of their biofidelity and infrastructure costs.

Service Throughput. Considering the need for large-scale use, we leverage the service throughput to measure the scalability of each solution on a single commodity physical server (with an Intel Xeon 64-core CPU@2.40GHz and 128-GB DDR memory). In §2.2, we demonstrate that the main burden imposed on the server comes from high CPU utilization. Thus, we record the number of concurrent PARs when the CPU utilization of an inference server reaches 100%, and measure the service throughput by the average number of the concurrent PARs supported by the four servers.

Acoustic Response Rate. Our observations (see from §2.2) show that when human players hear sound events (*i.e.,* gunshots and explosions), they tend to exhibit a high action saliency which is defined as the cosine distance of two consecutive action vectors ranging from [0,2]. Thus, we devise the *acoustic response rate* to quantifies PARs' ability to perceive and respond to sound events. In detail, we manually create sound events (*e.g.,* shooting and/or moving) behind PARs within a radius of 50 meters every second in 200 seconds, and count the number of responses (*e.g.,* turning and/or shooting) when the corresponding action saliency distance exceeds an empirical threshold of 0.6. The ratio of the responses to the sound events is regarded as the acoustic response rate.

Interaction Delay. As stated in §2.2, the interaction delay shows how quick PARs react to the environment. According to the system architecture in Figure 2, we take interaction delay as the sum of the client-server network delay (t_1), inter-server network delay (t_2), and inference delay for an action (t_3). Specifically, we sample interactions every second over a 100-second period, and record the corresponding t_1, t_2, and t_3.

Individual Biofidelity metrics (*IBM*). The *individual biofidelity metrics (IBM)* measure the resemblance of PARs to human players in these key micro-level behaviors: shooting, hitting, jumping, crouching, moving distances, and props gathered, which cover most gaming behaviors in 3D-FPS MMOGs. To eliminate probable measurement bias caused by individual sampling, we collect the above behavioral data of 96,812 opt-in users and 19,362 already-deployed PARs for two months (Oct.–Nov. 2021[1]) with a well-established IRB and informed user consent.

Specifically, we denote the *IBM* for the above micro behaviors as IBM_s, IBM_h, IBM_j, IBM_c, IBM_m, and IBM_p. They are calculated from the Jensen-Shannon Divergence (JSD, one of the mainstream statistical methods for measuring the divergence of two distributions [11]) based on behavior frequency distributions of PARs (P) and human players (H). Formally, IBM_b on each micro behavior b is defined as:

$$IBM_b = JSD(P_b \| H_b) = \frac{1}{2} KL(P_b \| M_b) + \frac{1}{2} KL(H_b \| M_b), \quad (3)$$

where $M_b = \frac{P_b + H_b}{2}$ and KL divergence [58] is the relative entropy of two distributions. A lower *IBM* indicates a closer behavioral resemblance between PARs and human players.

Mini Turing Test. However, the micro-level metrics are unable to well reflect the overall biofidelity of PARs [19]. Thus, we additionally employ the mini Turing test [18, 53] to evaluate the macro biofidelity of PARs perceived by real users and define the *mini Turing test pass rate* (*TPR*):

$$TPR = \frac{\#\ of\ the\ tests\ where\ PARs\ are\ labeled\ as\ humans}{\#\ of\ the\ total\ tests} \quad (4)$$

To obtain the *TPR*, we conduct another IRB-approved user trial under their consent. We employ 34 test users who have no common interest with 8-Game from a large university. These participants are diverse in terms of their gender (17 females), age (from 20 to 49), and education levels (from freshman to Ph.D.). We record game video clips (~35 seconds on average) of 100 PARs for each system, and 100 human players whose privacy information is well protected. Then, these participants watch all video clips, and label the relevant players in videos as PARs or human players according to their perceptions.

[1] After Nov. 2021 (until Apr. 2023) we constantly update the DRL model with new data and game features (if any), as well as monitor the performance of ParliRobo with the help from opt-in users. The results indicate that based on monthly maintenance of the model, the performance of ParliRobo can remain pretty stable – the fluctuations lie between -0.9% and 2.1% as compared to the performance during Oct.–Nov. 2021.