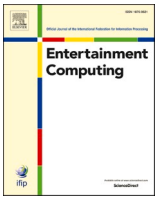
Entertainment Computing 48 (2024) 100603

Contents lists available at ScienceDirect

Entertainment Computing

journal homepage: www.elsevier.com/locate/entcom



Using provenance data and imitation learning to train human-like bots

Lauro Víctor Ramos Cavadas *, Sidney Melo, Troy Costa Kohwalter, Esteban Clua

Instituto de Computação, Niterói, RJ, Brazil

ARTICLE INFO

Keywords:
Non-player character
Imitation learning
Provenance
Games

ABSTRACT

Nonplayer Characters are becoming more realistic in their actions and behav- iors because of the development of gaming technology and gamers' increased demand for enhancements. While this progress is an exciting development, it has also become a major concern for game developers over the years, since players demand that NPCs look alike to other human players. Our major objective in this work is to make an NPC that satisfactorily mimics a player. This work proposes a method for training an NPC using imitation learning with the Generative Adversarial Imitation Learning framework to become similar to a human player. To simulate player behavior, our proposal trains agents using provenance data sets, cause-and-effect data mining, and the GAIL framework. The proposed model was developed to be universal and adaptable to different games. We validate our model using the DodgeBall game environment inside the Unity ML-Agents Toolkit for Unity Engine. Some players competed against our agent and found that our NPC was credible by observing his actions and behaviors. In this work, we present a new way of giving rewards compared to the model presented in the previous work. The tests and results found were also expanded, improving the validation of our model.

1. Introduction

Artificial Intelligence (AI) occupies an important place in games of any genre and is used by developers to create a world with a high degree of immersiveness in their virtual world. AI allows the developer to discover peculiar properties of a game world, such as the player's actions. The AI also makes it easier to improve the behavior of non-player characters (NPCs), allowing the developers to model the desired behavior more easily. The NPCs also may have more complex rules of behavior applied to them. The game industry is increasingly looking for non-player characters to have greater credibility in their games. This increases the immersion of the player and directly influences their enjoyment. NPCs are autonomous agents communicating with the player and the game world. As a result, they are crucial to the gameplay and the player's immersion. It takes a lot of domain expertise, knowledge engineering, scripting, intuition, and testing to create them using the conventional hand-crafting approaches.

In the meantime, the scale and depth of the criteria for the NPCs are growing. The scale requires various character types to be featured in the game to provide the impression of a well-populated virtual environment, and the depth is about making these NPCs more engaging. The existing approach to hand-crafting behaviors of NPCs is hard to scale in both dimensions, calling for alternative approaches [1]. Furthermore, empirical evidence indicates that players prefer to play with or against "human" NPCs [2].

In recent years, Imitation Learning (IL) has been investigated as a way to efficiently and intuitively program autonomous behavior [3,4,5,6,7,8]. IL is a technique for quickly picking up desired behavior by imitating it from an expert. IL is not only applicable to physical systems but is more commonly employed in robotics and can be useful for any system that needs autonomous behavior that resembles human experts. Inside IL, we decide to use the framework Generative Adversarial Imitation Learning (GAIL) [9] to train our NPC.

There have been many efforts to create believable NPCs in different game genres [10,11,12]. These works are classified into direct and indirect behavior imitation [11]. The direct imitation approach uses supervised learning algorithms that take human play as input traces. In contrast, the indirect imitation approach tackles this problem by maximizing a fitness function that evaluates the human likeness of an NPC's behavior.

Building NPCs that act like humans requires solutions to many challenging problems, including observing player activities, discovering player strategy models, developing practical AI technology, testing their believability, etc. In addition, it is difficult to discover which player's behaviors should be reproduced, and it is also very laborious and difficult to manually program all the actions that the NPC can perform,

* Corresponding author.
E-mail addresses: laurovrc@id.uff.br (L.V.R. Cavadas), sidneymelo@id.uff.br (S. Melo), troy@ic.uff.br (T.C. Kohwalter), esteban@ic.uff.br (E. Clua).

https://doi.org/10.1016/j.entcom.2023.100603
Received 10 April 2023; Received in revised form 2 August 2023; Accepted 14 August 2023
Available online 16 August 2023

especially when unpredictable situations may arise during the gameplay. Our proposal was built to help to facilitate the creation of believable NPCs. The game industry is also increasingly looking for NPCs to have greater believability. This increases the player's immersion and directly influences the provided fun for the players. Having a good reception from the players makes it easier to sell more game copies.

The main goal of this paper is to create a believable human-like NPC using real players' data. To reach this goal, we propose a model described by a sequence of steps for training a believable NPC using IL that does not require an expert to teach the NPC. Instead of the expert usage, we will use players' data from previous game sessions gathered by provenance. This way, the NPC will not behave with perfect mastery, which increases the believability to be a real player with their own correct and incorrect actions. As far as we know, provenance strategies for training IL was not used yet. Compared to a regular log, provenance may bring important features to this field due to these cause-and-effect relationships, which can bring more info about how the players play the game and build NPCs based on this. This can facilitate the visualization of the cause and effect of game elements' actions and help developers identify important gameplay elements. We also expect the presented model to be generic enough to be applied to any games with NPCs that can do the same actions as the player, like racing games, fighting games, puzzle games, shooter games, etc.

The rewards policy used in our model also differs from the rewards policy traditionally used to train the NPC. Our training is not based on giving rewards when the agent acts correctly or giving penalties when the NPC acts incorrectly. Instead, we reward the NPC when it reproduces the actions provided by the provenance data from players' game sessions. This model did not work very well as there was still a need to give rewards to the NPC according to the results of his actions because without them, he would not know the objective of the game. We name this reward "objective's rewards" and it is small if compared to the reward when performing a read action from the provenance file.

We choose the game engine Unity3D[1] to train and validate an NPC, using the DodgeBall[2] game environment present in open source Unity ML-Agents Toolkit[3]. Fig. 1 shows a picture of the DodgeBall game from the player's vision, showing the characters and the elements of the environment. The scenery elements are the walls, the bushes, and the balls distributed randomly at the beginning of the game. In the DodgeBall game, the character can move to the left, right, forwards, or backward. He can also perform a dash with cooldown time and throw a ball.

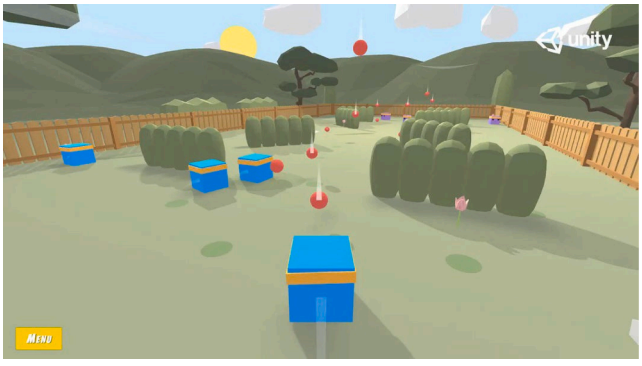


Fig. 1. Elements of DodgeBall game environment.

To collect a ball, the player has to pass over a ball on the ground.

After the players faced the trained NPC using the described model, they rated our NPC's credibility and, compared to the trained NPC's believability using the traditional rewards model, our NPC was considered more "human". This result was important for us to verify that we were on the right path and to continue deepening the experiments using our model.

This paper extends our previous contribution [13], where we proposed a first set of policies. In the present paper, we improve the rewards policy to make the NPC even more believable. We also validated it with a larger group of people and brought new promising results with a more detailed analysis. The remainder of the paper is organized as follows: The following section presents the theoretical foundation, the third section presents the related work, and the fourth section presents our proposed model. The fifth section brings the results, and the last section concludes this work, pointing out conclusions about the proposal and future works.

2. Theoretical foundation

In this section, we will present the knowledge that formed the basis for our work. Provenance is well understood in the context of art or digital libraries. It refers to the documented history of an art object or the documentation of processes in a digital object's life cycle [14]. In 2006, at the *International Provenance and Annotation Workshop*, the participants were interested in the issues of data provenance, documentation, derivation, and annotation. As a result, the Open Provenance Model (OPM) [14] was created from the Provenance Challenge that was held in that workshop [15].

The OPM is a proposed model of provenance that was designed to meet the following requirements [14]:

- Allow provenance information to be exchanged between systems;
- Allow developers to build and share tools to operate on such provenance model;
- Define provenance in a precise, technology-agnostic manner;
- Support digital representation of provenance;
- Allow multiple levels of description to coexist;
- Define a core set of rules that identify the valid inferences that can be made on provenance representation.

The causality graph comprises nodes representing Artifacts, Processes, and Agents. *Artifacts* are immutable pieces of state that can represent a physical object or a digital representation in a computer system. *Processes* are actions or a sequence of actions performed or caused by artifacts, resulting in new artifacts. *Agents* are contextual entities acting as a catalyst of a process that can enable, facilitate, control or affect its execution. The graph's edges represent a causal dependency between its source, denoting the effect, and its destination that denotes the cause [15].

Belhajjame et al. [16] created a conceptual data model that forms a ba- sis for the W3C provenance (PROV) family of specifications, formalizing and replacing what was presented in [14]. PROV-DM distinguishes core structures, forming the essence of provenance information, from extended structures catering to more specific uses of provenance.

For capturing and storing provenance data from a game session based on the Provenance in Games conceptual framework. Kohwalter et al. [15] proposed a novel approach named PinGU[4]. The wealth of provenance data collected during a game session is fundamental for understanding the mistakes made and reproducing the same results later. Causal relationships between game elements are mapped as edges connecting their respective nodes, resulting in a game provenance

[1] https://unity.com. Last accessed: 10 Feb 2023.

[2] https://blog.unity.com/technology/ml-agents-plays-dodgeball. Last accessed: 10 Feb 2023.

[3] https://github.com/Unity-Technologies/ml-agents. Last accessed: 20 Jan 2023.

[4] https://github.com/gems-uff/ping.

graph. Causality indicates a relationship between two events, where the former event affects the latter. The provenance approach captures causal relationships explicitly defined by the game developer. Each edge captured through the provenance approach represents a type of relationship (that can also be causal) between game objects' actions and/or states. The most important advantages of provenance graphs are the modeling of causal relationships, which structures the provenance elements into a graph, and its richness of detail.

Machine Learning (ML) techniques can be used to automate the process of learning how to play a video game either progressively using players' game traces as input, through direct imitation approaches or using some form of optimization technique such as Evolutionary Computation or Reinforcement Learning to develop a fitness function that, for instance, "measures" the human likeness of an agent's playing style [11].

The principle behind Imitation Learning is to allow an agent to act and exhibit human behavior by implicitly giving the learner information about the world. In IL tasks, the agent seeks the best way to use a training set (pair of inputs and outputs) demonstrated by an expert to learn a policy and achieve an action as similar as possible to the expert's one [17].

Imitation is often needed to automate actions when the agent is human, and it is too expensive to run its actions in real time. Apprenticeship learning [18], on the contrary, executes pure greedy/exploitative policies and uses all (state/action) trajectories to learn a near-optimal policy using Reinforcement Learning (RL) approaches. It requires difficult maneuvers and is nearly impossible to recover from unobserved states. IL can often deal with those unexplored states, so it offers a more reliable framework for many tasks, such as self-driving cars [17].

Training NPCs using AI techniques can make them very efficient. When using RL to train NPCs in situations where they are the player's opponents in a game, the NPC can become unbeatable, making the NPC look like a professional player. When using IL, the NPC can behave like a real player, but they usually behave similarly to their trainer and don't have many variations of actions. Fig. 2 shows the classification of IL.

Ho and Ermon [9] were interested in a specific setting of IL: the problem of learning to perform a task from expert demonstrations, in which the learner is given only samples of trajectories from the expert, is not allowed to query the expert for more data. At the same time, training is not provided reinforcement signal of any kind [9].

At that moment, there were two main approaches suitable for this setting: Behavioral Cloning (BC) [20], which learns a policy as a supervised learning problem over state-action pairs from expert trajectories; and Inverse Reinforcement Learning (IRL) [21,22], which finds a cost function under which the expert is uniquely optimal.

While appealingly simple, BC only tends to succeed with large amounts of data due to compounding error caused by covariate shift [23,24]. IRL, on the other hand, learns a cost function that prioritizes entire trajectories over others. Hence, compounding error, a problem for methods that fit single- timestep decisions, is not an issue. Accordingly, IRL has succeeded in a wide range of problems, from predicting behaviors of taxi drivers [25] to planning footsteps for quadruped robots [26]. Unfortunately, many IRL algorithms are extremely expensive, requiring RL in an inner loop [9].

Based on this, Ho and Ermon described a new framework called Generative Adversarial Imitation Learning (GAIL). They show that a certain instantiation of the GAIL framework draws an analogy between IL and generative adversarial networks, which derives a model-free IL algorithm that obtains significant performance gains over existing model-free methods in imitating complex behaviors in large, high-dimensional environments. GAIL explores randomly to determine which actions bring a policy's occupancy measure closer to the expert's, rather than methods that interact with the trainer [9].

Inside IL, our work uses the GAIL framework to train our models, directly extracting a policy from data. GAIL is interested in a specific set of imitation learning: the problem of learning to perform a task from expert demonstrations in which the learner is given only demonstrations from the expert, so he is not allowed to query the expert for more data. At the same time, training is not provided with a reinforcement signal. We chose to use GAIL instead of BC because the space of states we can have in an environment could be very large, and if the agent trained with BC didn't "see" a state before, the agent could not act as desired. GAIL focuses on learning a policy observed from the expert that is more approximate to a previous state and acts based on this without problems. It is important to remember that the "expert" that will train our NPC is not a skilled professional, but we will use casual players' data instead.

3. Related work

Previous works addressed the problem of imitating human players using different supervised learning approaches. [27] was used in Ms. Pac-Man and other game domains [12], and more recently, case-based reasoning [28] has been proposed due to its capacity for imitating spatially-aware autonomous agents in a real-time setting [29]. None of these works are based on real players' gameplay logs.

To build a system for learning and predicting individual player strategies by mining a series of actions from replays, Hsieh and Sun [30] adopted Aha et al.'s [31] Case-Based Reasoning (CBR) approach. Instead of constructing a new rule-based system, the main idea of CBR is to solve new problems based on the decision or solution of similar past experiences or problems. Although this work shows that the proposed system

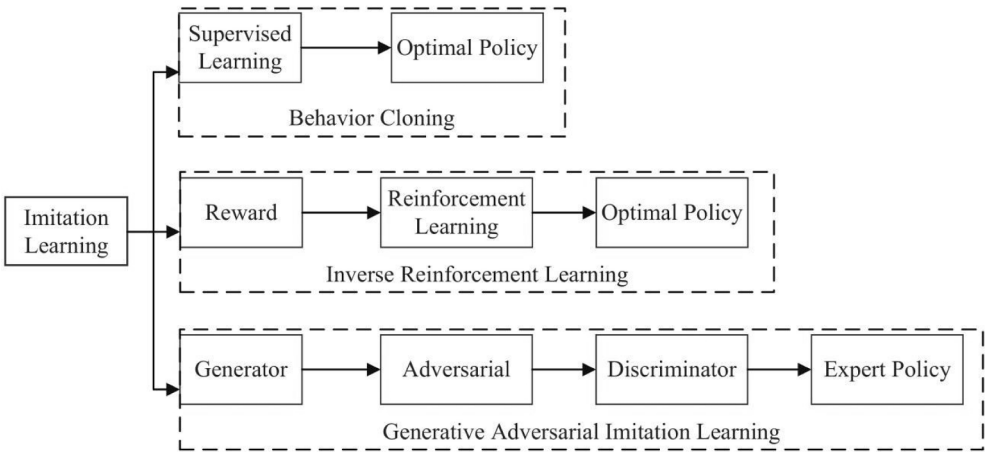


Fig. 2. Classification of Imitation Learning.
Source: [19]

can learn individual player strategies, it managed to learn the strategies of a player based on numerous replays of this human user. Our work differs from theirs since we propose using several gameplays from different players, and we are looking for action in the players' provenance file. When there is no similar action, we will take the closest possible actions to the existing ones.

Inspired by the idea of directly imitating human behavior, Karpov et al. [32] presented a component of the UT2 bot. The controller draws upon a previously collected database of recorded human games, which is indexed and stored for efficient retrieval. The controller works by quickly retrieving relevant traces of human behavior, translating them into the action space of the bot, and executing the resulting actions. The main difference between their work and ours is that the controller searches for a previously recorded situation. The agent will be stuck in the same position when this is not found. Our model uses the GAIL framework to prevent this situation.

Two designs for imitation-learning bots were discussed on Pelling and Gardner [33]. Both were based on support vector techniques and included a novel probabilistic model for in-combat jumping. One bot also includes an application of probability estimation trees to combat movement. Both designs appeared viable when tested under laboratory conditions and in the 2009 2K BotPrize competition format.

A model that enables the development of intricate behavior rules for artificially intelligently controlled video game characters was presented by Simonov et al. [34]. These rules allow the creation of characters with believable behavior that satisfies contemporary standards for artificial intelligence in video games. The model combines the Behavior Roles and Behavior Characteristics components to achieve believable situations. They enable characters to make more unique decisions and modify their behavior in response to the environment of a game. Unique sets of Behavior Characteristics are generated to create many agents with diverse and realistic behavior to in- habit vast game worlds. The difference between their work and ours is that our NPC knows how to act even if it does not know the current state of the environment. In their work, the NPC can get stuck if it is in an unknown state.

Finally, we discovered the most relevant work related to our research, conducted by Cruz and Uresti [35]. In their study, the authors addressed two primary challenges in their approach. The first challenge involved exploring domains with high-dimensional state-action spaces while adhering to constraints dictated by characteristics associated with human-like behavior. The proposed framework learns the model from a game by observing how humans play it to approach this. This procedure aimed to induce human-like behaviors in the bot that uses the learned model. Additionally, they proposed an exploration process—based on safe RL methods [36] indenting to refine the game model while maintaining induced human-like strategies. The second challenge was generating varied behaviors adapted to the opponent's playing style. They approached this problem by including a reward shaping mechanism [37,38]. The main difference compared to our model is the learning method. While their framework used RL in real-time to obtain the player's actions, we recreated games from provenance files and trained the agent with IL in our approach.

4. Framework for training NPCs using IL and provenance data

In this paper, we propose an Imitation Learning approach, based on provenance data, for training NPCs with believable behaviors instead of manually programming them. We differ from existing approaches since our model uses provenance data instead of a regular log. Besides this, while in most approaches, the training process is mostly based on rewards/penalties when the NPC acts correctly/incorrectly, we give rewards when it makes actions provided by the provenance dataset available from players' game sessions. We also give little rewards when it completes the correct tasks needed to turn an NPC competitive. Our reward system will be explained in detail later in this chapter.

In this section, we describe our proposed model, designed to be applied to various game types where the player and the NPC intended to be modeled can do the same actions, such as in fighting or racing games. We validate our proposal through the DodgeBall game, but the model can be applied to various games. DodgeBall game, a team-based game, has already been used for training NPCs through MultiAgent POsthumous Credit Assignment (MA- POCA) [39]. DodgeBall game allows the player to adopt different strategies, such as being more aggressive or carefully shooting balls from a long distance. In our model, we decide to train an NPC using GAIL. GAIL's goal consists of randomly exploring the action possibilities to determine which actions bring a policy's occupancy measure closer to the trainer's rather than strategies based on interacting with him. In our approach, we also will use provenance data gathered by the PinGU framework through different game-plays due to the richness and detailed data and feed it to the GAIL to train believable NPCs.

For NPC agents to mimic a human-like behavior, we must train an AI, producing a "brain", a trained AI inserted in an agent. To achieve this, we build the model's workflow that is illustrated in Fig. 3 that has four phases:

1. choice of game features needed to recreate player actions;
2. collect provenance data from gameplay sessions;
3. recreate players' game sessions based on collected provenance data;
4. use the recreated game session to train the NPC with the GAIL framework.

Before explaining the first phase of the model's workflow, we have to configure the game environment. When preparing the game, we preferably change the number of players so that it becomes one versus-one dispute, and, if necessary, we also change the victory condition.

After the game's environment configuration is important to note that when the player produces input data, the agent's control behavior checks which input was triggered and, according to it, fills the variable created to represent each action. At the end of each frame, the values contained in these variables are passed to a function, which checks the values of the variables and performs the correct actions. As an illustration, the corresponding variable is assigned a positive value when the player presses the key responsible for character movement in a forward direction. At the frame's conclusion, the variable is examined, and since it holds a positive value, the character moves forward accordingly.

4.1. Feature selection

The first stage of the method consists of verifying and choosing which gameplay's information and which inputs are required to be saved on the provenance file. It is necessary to save all inputs referring to the actions that the NPC will be able to do.

Saving the corresponding inputs that trigger the selected actions is crucial, as these actions are mapped to the repertoire of actions the NPC can perform. For example, one of the inputs to an action will be saved. The player must, however, be capable of doing this action. Provenance will also compile any data that is associated with an action. To collect provenance information, we use PinGU, which steps are described in more detail in the next sections.

4.2. Gathering player session data via provenance

In this stage, we include all the gathered provenance data at the playeŕs action, represented in a graph containing vertices representing player actions during the game session. The edges are the relationships between actions, agents, and events, and the game state information during each moment an action was recorded. Using PinGU, we can obtain data after four phases.

To initiate the process, a game object is introduced in the scene, functioning as a centralized server for managing provenance data. Two classes, namely ProvenanceController and InfluenceController, are then

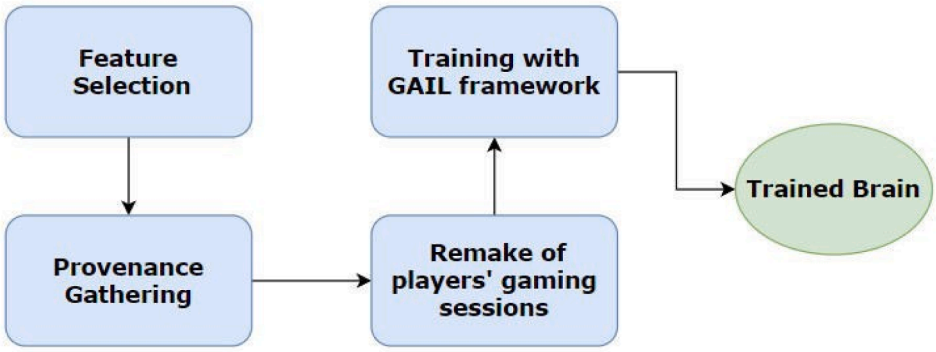


Fig. 3. Workflow composed of the phases of our proposed model.

linked to this game object. These classes work collaboratively to handle all aspects of provenance data management and construct the necessary graph for the game.

In the second phase, the ExtractProvenance class needs to be associated with the object created in the previous phase. Subsequently, it will be attached to the player's agent in the game. This class generates all provenance nodes related to the game entity. Once the nodes are created, the ProvenanceController class takes over and inserts these generated nodes into the graph.

In the third phase, we identify the various activities within the game and establish their relationships with other actions that require mapping. For instance, in our implementation, we differentiate between walking, rotating, throwing the ball, and being hit. However, it's important to note that provenance is not limited to actions alone; it also encompasses all other game-related activities, such as automatically picking up a ball from the ground (without requiring input from the player) or determining whether the player wins or loses.

During the fourth phase, we focus on creating domain-specific provenance tracking components and integrating them with the player's agent. We implement a corresponding provenance function within the object for each relevant action that requires tracking. This ensures that the provenance of every significant action the player takes is properly recorded and monitored throughout the game. As an example, when the player receives damage in the game, we create a new entry in the provenance file to document crucial details like the exact time of the event, the player's position, the number of remaining lives, the current health status, and the specific event labeled as"Being Hit". Fig. 4 shows the function called when an agent takes damage. This function generates a new vertex in the provenance graph, capturing all the relevant information selected in subsection 4.1.

For every mapped action executed, we record the corresponding input. For example, if the player moves along the x-axis during the game, this information is logged in the provenance file. This can be observed in Fig. 5, which presents a snippet of the provenance file containing a description of the player's action (Walking) and the associated saved input values.

Finally, it is necessary to add a provenance data export function to an event to save the current provenance graph to an external XML file. Fig. 6 shows a small snippet of the provenance file containing some

```
public void Prov_TakeDamage(string infID)
{
    Prov_GetPlayerAttributes();
    prov.NewActivityVertex("Being Hit", this.gameObject);
    // Check Influence
    prov.HasInfluence_ID(infID);
}
```

Fig. 4. Snippet of code to create a new vertex "Being Hit" on the provenance graph.

```
- <vertex>
    <ID>vertex_7</ID>
    <type>Activity</type>
    <label>Walking</label>
    <date>0,09331001</date>
  - <attributes>
    - <attribute>
        <name>XInput</name>
        <value>-1</value>
      </attribute>
    - <attribute>
        <name>ZInput</name>
        <value>0</value>
      </attribute>
    - <attribute>
        <name>RotateInput</name>
        <value>0</value>
      </attribute>
    - <attribute>
        <name>ShootInput</name>
        <value>0</value>
      </attribute>
    - <attribute>
        <name>DashInput</name>
        <value>0</value>
      </attribute>
    - <attribute>
        <name>HealthBlue</name>
```

Fig. 5. Snippet of provenance file with mapped inputs made in the player's session.

activities collected during the game.

For all the identified events that are important for the NPC training, we need to create a code that will create a new node on the provenance graph at the moment that the event occurs. For example, when a player loses the match, we call a function on the code that creates a new vertex on the provenance file. As the graph is very important to the developer because the information on it shows cause-effect relationships, the effort pays off.

Once this configuration is complete, it's time to collect the players' data on the game. We set up an agent to gather provenance data associated with it for validation reasons. The player will use this agent to face an enemy trained with the existing standard AI. The human player will play some matches, and the data generated by each match will be saved in the provenance file. Ultimately, we will have several game sessions with the player(s). In the provenance, it is possible to collect and merge data from a large set of players in different matches and save it in the same provenance file. Using every match as a training tool is

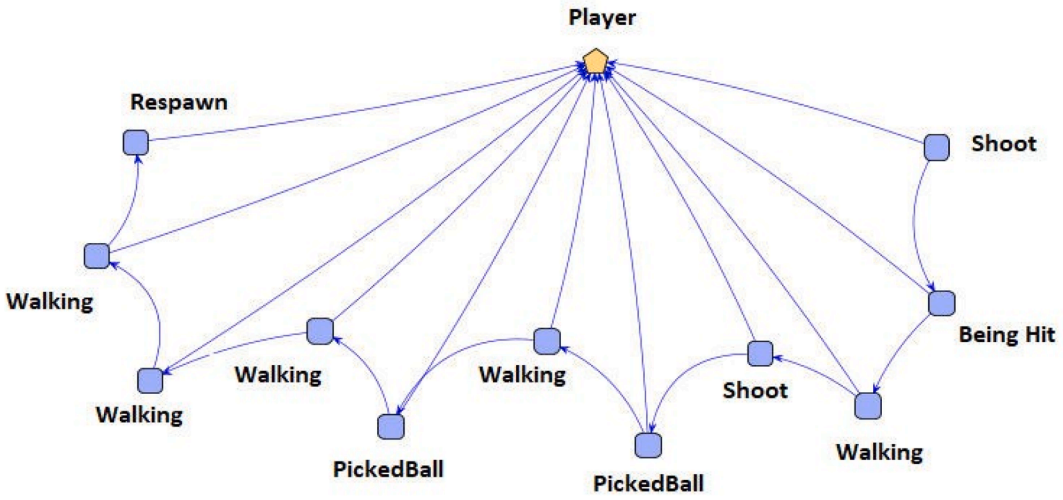


Fig. 6. Snippet of provenance graph with some activities saved in the provenance file.

also feasible by saving it in a separate file. This data will be examined in the next steps.

4.3. Recreating players' sessions based on provenance data for training

Every event or input is saved via provenance resulting in the creation of a node that is recorded on the provenance file. We can save one or more nodes on the provenance file at each frame.

We begin by reading the provenance file to reproduce the player's activities during the game. We must read and understand each node from the provenance file in XML format, one at a time. At this moment, public properties and fields of an object are converted to a serial format for storage or data transmission. Based on the results, a deserialization stage recreates the object in its initial state. It is feasible to think about serialization as storing an object's state in a stream or buffer.

Through a script interpretation stage, we get the values for each input at each node of the source file. We must transform the data from its string format, which is in XML format, to a numerical representation of the variables that control the character's actions. As a result, we pass each value to the variables in Unity Engine that represent the actions from the corresponding entries in the provenance file. During each frame, the function responsible for reading the character's actions updates the values of these variables based on the information retrieved from the provenance file's corresponding node. All the process steps to recreate the player's actions from provenance data are represented in Fig. 7. The process will be executed during the agent's training, described in the next stage.

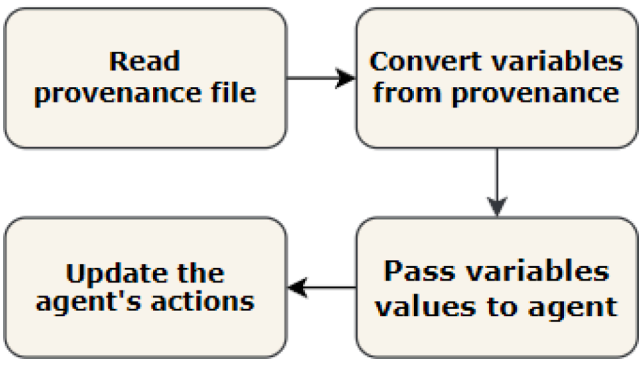


Fig. 7. Required steps to recreate actions saved on provenance file for training the agent.

4.4. Training with the GAIL framework

At this stage, the game is ready to recreate the contents of a provenance file already gathered from previous gameplay sessions. Before starting the game and recreating the actions represented by the provenance, it is necessary to configure how the agent will be trained. Since GAIL does not interact with the expert during training, the NPC won't learn from an expert and will not behave like this expert. As stated before, we chose the framework GAIL to train our NPC because the policy learning from GAIL can help the NPC avoid unseen situations in games that can cause undesired behaviors. The created NPC will act according to the learned policy and behavior from the nodes saved on the provenance file.

We first created a neural network configuration file for training an NPC using GAIL with data from many players stored as a provenance file, including the GAIL framework configuration. Unity ML-Agents Toolkit already has GAIL implemented, so we need to configure the network parameters[5], such as strength and learning rate (used to update the discriminator). Fig. 8 shows a snapshot from the training configuration file with the GAIL configuration used to implement the training on the DodgeBall game.

Numerous game sessions from the same player or other users can be used during this phase. If more than one user is being acquired, the procedure begins with the data from the first player, and the training is increased with the data from all the other players.

After loading the training data from the provenance file and finishing the GAIL configuration, it is possible to start training with our model. When starting the game, the NPC starts to reproduce the actions described in each node from the provenance file, as explained in the previous section. Usually, we reward good deeds like defeating the enemy and penalize bad actions like crashing into a wall so that it can know if the action had generated a good or a bad result. However, when choosing this traditional approach, the bot will be trained to be extremely efficient, always taking the best actions and becoming a formidable enemy. This work proposes the creation of an NPC that behaves like a human, and to reach that goal, we will reward the bot when it takes the actions described on the provenance file, whether these actions are good or bad, so it will learn to behave exactly as the human player.

Training the NPC through several steps does not guarantee that the

[5] https://github.com/Unity-Technologies/ml-agents/blob/main/docs/Training-Configuration-File.md. Last accessed: 30 Jul 2022.

```
reward_signals:
      gail:
        gamma: 0.90
        strength: 0.6
        network_settings:
          normalize: false
          hidden_units: 128
          num_layers: 2
          vis_encode_type: simple
          memory: null
          goal_conditioning_type: hyper
          deterministic: false
        learning_rate: 0.0005
        encoding_size: null
        use_actions: false
        use_vail: false
        demo_path: Demos/DodgeBallDemo_0.demo
init_path: null
keep_checkpoints: 50
checkpoint_interval: 3000000
max_steps: 1000000000
```

Fig. 8. Screenshot taken from the neural network configuration file.

NPC learns the objectives of the game. For example, when we collect the provenance data of a match, let's assume the NPC is at coordinate *x1—z1* and the enemy is at coordinate *x2—z2*. The player then realizes an action considered correct for winning and receives a positive reward, like throwing a ball and hitting the opponent. At this moment, the coordinates and the information about the result of the action are saved in a node in the provenance file. In the training step, when we reproduce this action using the information from this node, the player will do the same, but there is no guarantee that the opponent in the current game is not at a coordinate other than *x2—z2*.

For efficiently teaching the game's objectives to the NPC is necessary to change part of the rewards policy, reducing the rewards when it completes the correct tasks required to turn an NPC competitive. Using the correct proportion of rewards is necessary to avoid the NPC becoming invincible. We made several tests with different rewards, and our NPC became invincible sometimes or didn't know what was the game's objective at other times due to the change in the objective's rewards. These tests were made via observation and feedback from users that were not the users that tested our final model. After these tests, we found a proportion that can make our NPC acts like a real player.

When the training begins, and the NPC starts to imitate the player's actions according to the provenance file, we reward the NPC with a weight of *1* for every action. When the NPC completes a task needed to understand the game's objective, we reward it with a weight of *0.1*. This weight difference makes the NPC focus much more on acting as the player while still trying to win the game. The weights of the rewards also work in theory in other games since the important thing is the proportion of rewards during training.

5. Results

For validating and testing our proposal, we use a game called DodgeBall. The scenario where the game occurs has walls and obstacles scattered throughout the environment. Thrown balls bounce off obstacles or walls. There are two types of games: capture the flag and elimination. In both modes, the match has eight characters, 4 per side. We validate using the elimination mode where the character's life is 2. The characters must collect the balls on the ground and throw them toward the enemy. When the ball hits a character, it loses 1 of life. The match ends when all the characters from the opposite side are defeated.

Considering our test scenario, four players are on each team, as shown in Fig. 1. We reduced the number of players from the original game, so each team has just one character left. We also have to change the win conditions to allow victory when defeating one enemy since there is only one enemy in the proposed game.

The existing agent in the DodgeBall game was trained using a traditional reward system, which constantly improves the agent's capabilities by rewarding good actions and penalizing bad actions. As such, the agent can be better over time than the human expert that trained it and is almost unbeatable. Another important point in our model is to reward the agent when it takes actions performed by a player and recorded in the provenance, regardless of whether the actions had good or bad results. This will make the agent not perfect and make the same mistakes as a real player.

We invited twelve players, ranging in age from 17 to 32, to play the DodgeBall game in six different scenarios as part of the validation of our model. Traditional A, B, and C scenarios have agents with brain training through the traditional reward system. During training, we give positive rewards for successful tasks and penalties for unsuccessful tasks. In scenarios named Model A, Model B, and Model C, the agent has a brain trained with our training methodology, rewarding all the actions read from the provenance file. The difference between scenarios A, B, and C in both training methods is the number of steps for training. Table 1 shows the information about each scenario configuration.

After the training process ends, we have the already trained NPC with the traditional reward system and a trained agent trained with our model. Using our model, if the NPC has encountered the game's current state before, the agent will replicate the same action as previously observed. However, if the current state is new to the NPC, it will search for an action that closely resembles the present situation and act accordingly. We ask the players to play the game facing the agent using the traditional reward system and the agent trained using our model. We ask them to rank the NPC based on how similar they believe the NPC's moves and act, according to a human player, through a Likert scale ranging from 1 to 10, with 1 denoting the complete opposite of a player behavior and 10 designating the close to or perfect as a player behavior. The following subsections describe the NPC behavior and player's feedback in each scenario.

5.1. Traditional A

In this scenario, the NPC's movements were very simple and carried out without much logic. The logic of collecting balls from the ground was far from the ideal. The movement's initial outcome was far from ideal and needed more training time. The NPC crashed into walls or obstacles. The NPC did not execute the dash move.

Concerning the shots, the NPC occasionally took balls to make the throw, but as soon the NPC collected a ball, it threw it in the direction it was facing without aiming the player.

The NPC's moves were simple, so players easily beat this NPC. The ball throws were badly executed. Only four of the twelve human players thought they were dealing with a real player. When they saw the NPC throw the ball in the wrong direction, all were certain it was not a real player.

Table 1
Scenarios.

Scenario name	Trained traditionally	Trained using our model	Number of steps
Traditional A	Yes	No	3,000,000
Traditional B	Yes	No	4,500,000
Traditional C	Yes	No	6,000,000
Model A	No	Yes	3,000,000
Model B	No	Yes	4,500,000
Model C	No	Yes	6,000,000

5.2. Traditional B

In this scenario, some improvements were observed in the NPCs' movements, with the character moving between obstacles and searching for balls to collect and throw. Compared to scenario A, the outcome was improved, particularly regarding the movements, even while using the dash movement. When the NPC picked up a ball from the ground, it could stay with it longer before throwing. The aim was not precise, even with improvements in throwing the ball.

The moves made by the NPC, such as being shrewd when seeking for balls and holding onto the balls before throwing them, were perceived by all the players as being more realistic. Human players noticed certain ball throws made by the NPC as being random. Yet, compared to the prior situation, about 35% of ball shots succeeded in hitting the player.

5.3. Traditional C

The NPC's mobility is significantly more effective in this configuration, gathering balls swiftly and avoiding collisions with obstructions. With a hit rate close to 55%, ball throwing has greatly improved. The NPC was viewed as aggressive and attempting to end the match quickly.

In this scenario, the NPC was considered challenging to beat and even won some games. Players generally thought they were playing against a skilled opponent because of the NPC's movement and effective throws. Despite the advancements, some players complained that the NPC was extremely challenging and, because of their efficiency, the NPC believability was perceived as a real player with some great skills or a machine.

5.4. Model A

The NPC tried to collect the balls while moving without using the dash move and colliding with walls and barriers sporadically. Compared to scenario Traditional A, the NPC holds the balls longer. However, the throw lacked accuracy. Although the NPC presented little hardship to the players, they nevertheless thought it was an improvement over Traditional A.

5.5. Model B

In this case, the NPC gathered balls efficiently, hit walls and barriers less often, and improved accuracy when throwing balls. However, the shots were still made with regular errors. The NPC often throws the ball after picking it up from the ground, as some players use this tactic, but the NPC does it at the wrong time. Players generally said the NPC was more believable and appeared to be a player for most of the game. Comparing this scenario with Traditional B, the participants felt an increase in the credibility of the NPC.

5.6. Model C

The NPC navigated the last scenario fairly successfully, collecting the balls with a dash and only colliding with obstacles after the dash. Only one type of throwing error remains: when an NPC "saw" the player, but there were balls on the ground before them or the edge of a barrier to stop the ball. The majority of shooting errors are ones that genuine players frequently commit. The players during tests were perceived as making more errors when throwing rather than moving.

Since this situation closely resembled a real player, human players believed this version was the most entertaining despite a few small flaws. The movement was appreciated since it moved well but made minor shooting and movement mistakes. The faults made by the NPC were seen by the players similarly to those expected from a human player.

5.7. Analysis of results

Table 2 shows the ratings given by the players related to the NPC's movements for each of the specified scenarios. Table 3 shows the ratings given by each player in the specified scenarios about the NPC's ball throwing. According to Tables 2 and 3, there were some gains in both training models as the number of steps increased. The ball shots were executed with higher precision and exhibited less randomness, as the movement closely resembled the movement performed by a player. The NPC was getting more difficult to deal with as it improved at moving and hurling the ball. The NPC originally appeared to be less like a real player but gradually appeared to be more like one.

Despite its increased difficulty, the Traditional C NPC's remarkable shooting accuracy and fluid movement unmistakably conveyed the impression of facing a machine rather than a human opponent. When comparing the ratings of NPC's shots and movements of Traditional B against the ratings NPC of Traditional C, both trained in the traditional model of reward, it is possible to notice that some ratings got worse in the scenario Traditional C.

The rating given to the NPC of scenario Model C improved the perception of being a real player compared to the scenario results from Model B. It became real but continued to make some minor errors that typically human players would make. This was true when comparing the same number of steps in the training proposed in our model. Among all the situations, the NPC in scenario Model C generally succeeded in conveying the strongest sense of being a real player.

Because there were more steps taken during training in scenario Model C than in Model B and Model A, the NPC got more convincing results. According to the increase in training steps, the NPC in scenarios Traditional A, Traditional B, and Traditional C became extremely effective when rewarding proper behavior and punishing bad behavior. A machine or a very skilled player could only attain this efficiency, lowering the NPC's believability.

These results demonstrated that, despite some enhancements, our model worked properly and that the NPC perceived by players resembles a real player more closely than the NPC created using the conventional training models. The DodgeBall game proved satisfactory for validating our solution. It is not a highly specific game with complex actions but is similar to most games with simple elements, such as moving and shooting.

6. Conclusion and future work

In competitive multiplayer games, players seek experiences that offer greater immersion and believable behavior. One of the key goals of AI techniques is to develop agents that behave like humans. Our proposal offers a novel paradigm built on provenance approaches for producing NPCs that perform actions close to those performed by human players, enhancing the player's sense of immersion as they are observed.

Our approach requires gathering information from many players via provenance data and using these data to train NPCs with the GAIL framework. In the literature, these two ideas have not yet been applied together. As provenance also involves cause-and-effect links, it gives more specific knowledge about the events that occur while a player acts. The GAIL framework for imitation learning enables the NPC to learn a policy and create a roughly accurate action that a player would take.

It is crucial to train NPCs without the help of an expert so that their performance is more similar to real players rather than optimized. Also, the expert would need much gaming time to train the agent. Our method can combine the data from various participants into a single brain through provenance models. Although the model was created to be general and usable in various games, the environment used was the DodgeBall game from the Unity Engine ML-Agents package. We changed the game to a one-against-one format and used the elimination game mode for validation. Currently, we are simulating player behaviors based on their input. Future advancements will enable to handle more

Table 2
NPC's movement credibility ratings assigned by players.

	Players												Average Rating
	A	B	C	D	E	F	G	H	I	J	K	L	
Traditional Training													
Traditional A	5	3	4	4	4	5	4	5	4	5	3	4	4,16
Traditional B	8	6	9	7	7	7	6	7	8	8	7	9	7,41
Traditional C	7	7	8	7	6	6	7	7	6	7	6	9	6,91
Our Model Training													
Model A	6	4	6	5	5	6	4	5	5	6	5	7	5,33
Model B	8	7	9	7	8	8	7	7	8	9	7	10	7,91
Model C	9	8	10	7	8	9	8	9	8	9	8	10	8,58

Table 3
NPC's shots credibility ratings assigned by players.

	Players												Average Rating
	A	B	C	D	E	F	G	H	I	J	K	L	
Traditional Training													
Traditional A	3	2	3	3	2	4	3	3	4	2	2	3	2,83
Traditional B	6	7	6	7	6	6	7	7	6	5	6	7	6,5
Traditional C	6	8	6	6	7	7	6	7	7	5	6	7	6,5
Our Model Training													
Model A	3	3	4	3	2	4	4	5	5	4	5	5	3,91
Model B	5	6	7	5	5	7	6	8	6	6	7	8	6,33
Model C	6	7	8	7	5	9	8	8	8	6	8	9	7,41

complex behaviors by utilizing the influence already available at the captured provenance.

Twelve users were selected to participate in the game in six different scenarios. In scenarios Traditional A, Traditional B, and Traditional C, they were confronted with an NPC trained with the traditional reward model, which promotes good behavior and penalizes negative behavior. Following, they faced NPC in Model A, Model B, and Model C, trained with our model's reward system, and received rewards for all activities in the provenance file. In addition, we distribute a small number of rewards known as objective rewards to NPCs for the discovery of the game's goal. The players rated the NPC based on how it behaved in the game. When we compared the ratings the players gave to the different scenarios Traditional A, Traditional B, and Traditional C with the scenarios Model A, Model B, and Model C, we noticed a significant improvement in the ratings of our model, indicating that our approaches converge to credible behaviors. The NPC started to behave more like a genuine player, according to input from all players, which was reflected in the ratings. The scores for our NPC give us a positive feeling that while our model still requires work, it is headed in the right direction. The model can be scaled up to provide better results and an NPC that behaves almost like a real player.

We should improve the training procedure in the following stage of our work and observe how the agent acts while using more provenance data from more players. The impact of the number of graphs on NPC training should also be examined. Also, practicing for longer periods and with more circumstances to compare is crucial. With more formal believability testing methods found in the literature, we also want to examine the plausibility of an agent developed through the model described in this work.

Another future task is to group various users' profiles such that the recorded sessions adhere to some user profiles (kids, experienced players, casual gamers, etc.). Allowing players to select the type of NPC they would face would make the game less frustrating and more enjoyable.

We also intend to validate our method on other games. The model was intended to be general in theory; nevertheless, it does need training using the GAIL framework and preparation of the provenance gathering step, being this is the only limitation applicable in our work.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] I. Borovikov, J. Harder, M. Sadovsky, A. Beirami, Towards interactive training of non-player characters in video games, arXiv preprint arXiv:1906.00535 (2019).
[2] R. Arrabales, J. Muñoz, A. Ledezma, G. Gutierrez, A. Sanchis, A machine consciousness approach to the design of human-like bots, in: Believable Bots, Springer, 2013, pp. 171–191.
[3] P. De Haan, D. Jayaraman, S. Levine, Causal confusion in imitation learning, Adv. Neural Informa. Process. Syst. 32 (2019).
[4] S.K.S. Ghasemipour, R. Zemel, S. Gu, A divergence minimization perspective on imitation learning methods, in: Conference on Robot Learning, PMLR, 2020, pp. 1259–1277.
[5] A. Billard, D. Grollman, Robot learning by demonstration, Scholarpedia 8 (12) (2013) 3824.
[6] J.A. Bagnell, An Invitation to Imitation, Tech. rep., Carnegie-Mellon Univ Pittsburgh Pa Robotics Inst, 2015.
[7] T. Osa, J. Pajarinen, G. Neumann, J.A. Bagnell, P. Abbeel, J. Peters, et al., An algorithmic perspective on imitation learning, Found. Trends® Robotics 7 (1–2) (2018) 1–179.
[8] A.G. Billard, S. Calinon, R. Dillmann, Learning from Humans, Springer Handbook of Robotics, 2016, pp. 1995–2014.
[9] J. Ho, S. Ermon, Generative adversarial imitation learning, Adv. Neural Inform. Process. Syst. 29 (2016).
[10] A. Khalifa, A. Isaksen, J. Togelius, A. Nealen, Modifying mcts for human-like general video game playing, 2016.
[11] J. Togelius, R. De Nardi, S.M. Lucas, Towards automatic personalised content creation for racing games, in: 2007 IEEE Symposium on Computational Intelligence and Games, IEEE, 2007, pp. 252–259.
[12] J. Ortega, N. Shaker, J. Togelius, G.N. Yannakakis, Imitating human playing styles in super mario bros, Entertain. Comput. 4 (2) (2013) 93–104.

[13] L.V.R. Cavadas, E. Clua, T.C. Kohwalter, S.A. Melo. Traininghuman-like bots with imitation learning based on provenance data, in:2022, IEEE, 2022, pp. 1–6.
[14] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwas- nikowska, S. Miles, P. Missier, J. Myers et al., The open provenance model core specification (v1. 1), Future Gen. Comp. Syst. 27(6) (2011) 743–756.
[15] T. Kohwalter, E. Clua, L. Murta, Provenance in games, in: Braz. Symp. Games Digit. Entertain. SBGAMES, 2012, pp. 162–171.
[16] K. Belhajjame, R. B'Far, J. Cheney, S. Coppens, S. Cresswell, Y. Gil, P. Groth, G. Klyne, T. Lebo, J. McCusker, et al., Prov-dm: The prov Data Model, W3C Recommendation 14 (2013) 15–16.
[17] A. Attia, S. Dayan, Global overview of imitation learning, arXiv preprint arXiv: 1801.06503 (2018).
[18] P. Abbeel, A.Y. Ng, Apprenticeship learning via inverse reinforcement learning, in: Proceedings of the Twenty-first International Conference on Machine Learning, 2004, p. 1.
[19] J. Hua, L. Zeng, G. Li, Z. Ju, Learning for a robot: deep reinforcement learning, imitation learning, transfer learning, Sensors 21 (4) (2021) 1278.
[20] D. Pomerleau, An autonomous land vehicle in a neural network, Adv. Neural Inform. Process. Syst. 1 (1998).
[21] S. Russell, Learning agents for uncertain environments, in: Proceedings of the Eleventh Annual Conference on Computational Learning Theory, 1998, pp. 101–103.
[22] A.Y. Ng, S. Russell et al., Algorithms for inverse reinforcement learning, in: Icml, vol. 1, 2000, p. 2.
[23] S. Ross, D. Bagnell, Efficient reductions for imitation learning, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 661–668.
[24] S. Ross, G. Gordon, D. Bagnell, A reduction of imitation learning and structured prediction to no-regret online learning, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.
[25] B.D. Ziebart, A.L. Maas, J.A. Bagnell, A.K. Dey et al., Maximum entropy inverse reinforcement learning, in: Aaai, vol. 8, Chicago, IL, USA, 2008, pp. 1433–1438.
[26] N.D. Ratliff, D. Silver, J.A. Bagnell, Learning to search: functional gradient techniques for imitation learning, Auton. Robots 27 (1) (2009) 25–53.
[27] M. Miranda, A.A. Sanchez-Ruiz, F. Peinado, A neuroevolution approach to imitating human-like play in ms. pac-man video game, in: CoSECivi, 2016, pp. 113–124.
[28] M. Miranda, A A. Sánchez-Ruiz, F. Peinado, A cbr approach for imitat- ing human playing style in ms. pac-man video game, in: International Conference on Case-Based Reasoning, Springer, 2018, pp. 292–308.
[29] M.W. Floyd, A. Davoust, B. Esfandiari, Considerations for real-time spatially-aware case-based reasoning: a case study in robotic soccer imitation, in: European Conference on Case-Based Reasoning, Springer, 2008, pp. 195–209.
[30] J.-L. Hsieh, C.-T. Sun, Building a player strategy model by analyzing replays of real-time strategy games, in: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), IEEE, 2008, pp. 3106–3111.
[31] D. W. Aha, M. Molineaux, M. Ponsen, Learning to win: Case-based plan selection in a real-time strategy game, in: International Conference on Case-based Reasoning, Springer, 2005, pp. 5–20.
[32] I.V. Karpov, J. Schrum, R. Miikkulainen, Believable bot navigation via playback of human traces, in: Believable Bots, Springer, 2013, pp. 151–170.
[33] C. Pelling, H. Gardner, Two human-like imitation-learning bots with probabilistic behaviors, in: 2019 IEEE Conference on Games (CoG), 2019, pp. 1–7.
[34] A. Simonov, A. Zagarskikh, V. Fedorov, Applying behavior character- istics to decision-making process to create believable game ai, Procedia Comput. Sci. 156 (2019) 404–413.
[35] C. Arzate Cruz, J.A. Ramirez Uresti, Hrlb 2: a reinforcement learning based framework for believable bots, Appl. Sci. 8 (12) (2018) 2453.
[36] J. Garcıa, F. Ferńandez,, A comprehensive survey on safe reinforcement learning, J. Mach. Learn. Res. 16 (1) (2015) 1437–1480.
[37] A. Y. Ng, D. Harada, S. Russell, Policy invariance under reward trans- formations: Theory and application to reward shaping, in: Icml, vol. 99, 1999, pp. 278–287.
[38] E. Wiewiora, G.W. Cottrell, C. Elkan, Principled methods for advising reinforcement learning agents, in: Proceedings of the 20th International Conference on Machine Learning (ICML-03), 2003, pp. 792–799.
[39] A. Cohen, E. Teng, V.-P. Berges, R.-P. Dong, H. Henry, M. Mattar, A. Zook, S. Ganguly, On the use and misuse of absorbing states in multi-agent reinforcement learning, arXiv preprint arXiv:2111.05992 (2021).