Developing Learnable AI From Human Player's Behavior in Universal Fighting Engine

Ryoya Ito s1260233 Supervised by Prof. Maxim Mozgovoy

Abstract

Nowadays, video games have become a popular form of entertainment, and all kinds of games leverage artificial intelligence (AI) technology. Although the accuracy of game AI has been improving year by year, fighting game AI still often moves unnaturally. Such non-human-like behavior can spoil the enjoyment of the gaming experience. This study aims to solve this problem by designing an AI that can learn behavior from human players and act like a human. We also evaluate how the constructed AI and the human player are similar using a measure called cosine similarity.

1 Introduction

Video games are a popular form of entertainment enjoyed by many people. Among them, fighting games are one of the most popular game genres that have been gaining popularity over the years. Fighting games are one-on-one games in which players attack each other using a variety of actions such as punches, kicks, and special moves, and the player who loses the opponent's health or reduces it more within a time limit wins.

Recently, artificial intelligence (AI) technology has been introduced into various genres of games, and fighting games are no exception. The term "fighting game AI" mostly indicates the AI that controls the NPCs (Non-Player Characters) participating in a match. The performance of game AI has been improving every year. However, fighting game AI still often behaves in an unnatural and non-human-like manner. Such non-human-like behavior can cause the player to feel uncomfortable and thus undermine the quality of the gaming experience. In order to make games enjoyable, it is important for AI to have human-like characteristics [1].

In this study, we investigate whether it is possible to construct a fighting game AI that can behave like a human by learning behavior from an actual human player. For this purpose, we use an approach called "behavior capture." In this approach, a human player first plays a fighting game to collect match data. Next, the AI observes/learns from those match data and forms knowledge. Finally, the AI acts based on the acquired knowledge. In our research, we use the Universal Fighting Engine (UFE) as an environment. We also evaluate how the constructed AI is similar to the training source human player using a measure called cosine similarity.

2 Learning AI

2.1 Universal Fighting Engine

In our research, we design an AI system that runs on the Universal Fighting Engine (UFE) [7]. UFE is an open-source platform for developing one-on-one fighting games in Unity [8]. UFE is highly customizable; users can add, change, or delete options at will. UFE also supports a variety of actions such as attacks and special moves and has several pre-implemented character models with different types of actions that we can use. The players play the game by controlling these characters using the four directional keys and six attack buttons.

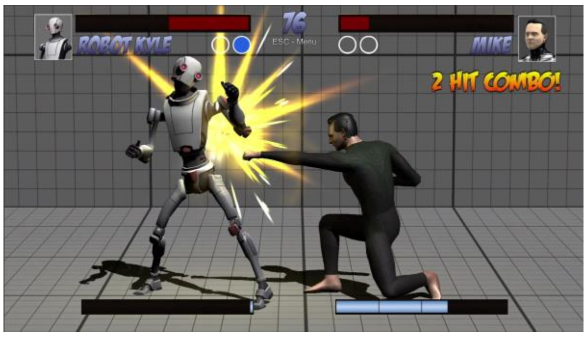


Figure 1: Universal Fighting Engine

2.2 Artificial Contender

The core algorithms and data structures of our AI system are implemented using TruSoft's "Artificial Contender" middleware [6]. We call the AI constructed with this tool Artificial Contender AI (ACAI). The architecture of our ACAI system is shown in Figure 2.

2.3 Recording Human Playing

In our behavior capture approach, we need training data from the human player for learning the AI. For this purpose, our system is equipped with a mechanism to record the situation of the match in the game as a log file by playing UFE. The log file consists of basic and important information in the fighting games. For example, the character's coordinates, health points of the character, and status of the character are included in the log file. Table 1 shows the details of the log file components.

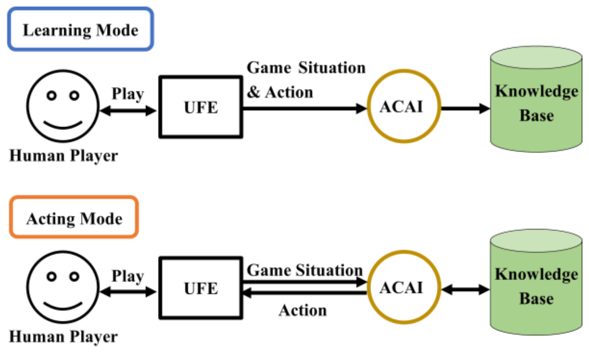


Figure 2: Architecture of the ACAI system

Component	**Meaning**
currentState	Character's state (e.g., Stand, Jump, Down)
currentSubState	Additional character's state other than currentState (e.g., Resting, Blocking)
x	Character's x coordinate
y	Character's y coordinate
z	Character's z coordinate
currentBasicMove	Character's basic movement state (e.g., Idle, MoveForward)
currentMoveName	Character's attack action (e.g., Light Punch, Jumping Heavy Kick)
IsBlocking	Whether or not the character is blocking (True or False)
characterHealthSelf	Character's health points
characterDistanceSelf	Distance between the characters

Table 1: The log file components

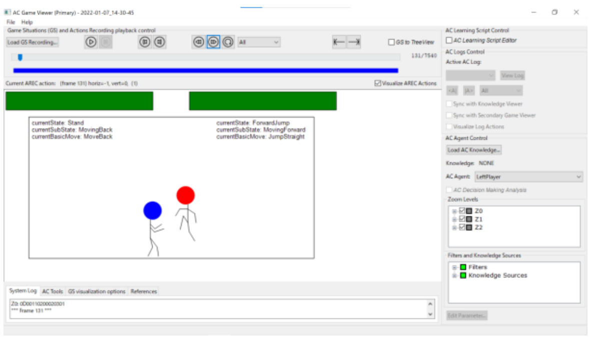
2.4 Acting Graph

Based on the given training data, the ACAI constructs the knowledge base according to a data structure called acting graph [4, 5]. The details of the construction process of the knowledge base are as follows. Every time the training source player performs an action, the ACAI stores the pairs of the performed action (we call it Action) and the state of the game world at that time (we call it GameSituation) in the knowledge base. Note that Action includes the case of "do nothing." GameSituation consists of the attributes of each character participating in the match, e.g., character coordinates and status. Action and GameSituation are recorded in the aforementioned log file, which is the training data.

Since the ACAI also stores the consecutiveness between GameSituations as a link in the knowledge base, the whole knowledge base can be considered as a directed graph that stores action chains. Such a graph is the acting graph. Each node in the acting graph corresponds to GameSituation, and each edge corresponds to Action that causes changes in the game situations. For example, the "crouch" action connects two game situations that are different in whether the character is crouching or not.

In addition, the Action object has a counter that counts the number of times the same (Action, GameSituation) pair is recorded. If the same (Action, GameSituation) pair is recorded multiple times, it means that the Action is preferred and used many times in that GameSituation. By weighting the actions with counters, the ACAI can prioritize the more frequent actions under such a certain game situation.

We use TruSoft's ACGameViewer tool to construct the knowledge base. This tool reads the log files obtained by playing UFE and constructs a knowledge base for learning the ACAI. Figure 3 shows a screenshot of ACGameViewer.



Current AREC action: (frame 131) horiz=-1, vert=0, (1)

currentState: Stand
currentSubState: MovingBack
currentBasicMove: MoveBack

currentState: ForwardJump
currentSubState: MovingForward
currentBasicMove: JumpStraight

Knowledge: NONE
AC Agent: LeftPlayer

```
Z0: 0D00110200020301
*** Frame 131 ***
```

Figure 3: ACGameViewer

3 Acting AI

3.1 Action Searching

The learned ACAI selects the most appropriate action for the current game situation using the constructed knowledge base. For this purpose, it is necessary to identify the node in the acting graph (i.e., GameSituation) that matches the current game situation received from the UFE and then select the outgoing edge (i.e., Action) from that node. The ideal scenario is to identify GameSituation in the knowledge base that perfectly matches the current game situation. However, a perfect matching of GameSituation is rarely achieved, and it is not realistic. Therefore, in our system, the ACAI identifies the closest GameSituation to the current game situation approximately, and it selects the Action corresponding to the found GameSituation.

Our ACAI system defines two sets of attributes with different numbers of contained attributes and uses them for GameSituation matching. Table 2 shows the detailed contents of the attribute sets. The attribute set of abstraction level 0, which has higher precision, consists of eight attributes. On the other hand, the attribute set of abstraction level 1, which is less precise than level 0, contains only four attributes. GameSituation matching is done by collating the attributes contained in the target attribute set between the current game situation and the GameSituation in the knowledge base. First, we use the attribute set at level 0 to find a GameSituation that matches the current game situation. If the matching GameSituation is not found, we reduce the number of attributes used for matching (i.e., use the attribute set at level 1) and do the matching again with lower precision.

Abstraction level	**Attributes**
Level 0	Player’s x coordinate, Player’s y coordinate, Opponent’s x coordinate, Opponent’s y coordinate, Player’s State, Player’s SubState, Opponent’s State, Opponent’s SubState
Level 1	Player’s x coordinate, Player’s y coordinate, Opponent’s x coordinate, Opponent’s y coordinate

Table 2: The details of the attribute sets

In addition to GameSituation, action chains are also taken into account for action selection. The ACAI should not act according to the current game situation only, ignoring the past behavior of the training source player; the ACAI should also have the capability to reproduce a series of actions of the player (action chains) such as combo attacks. Since the knowledge base has the structure of the acting graph, if we want to reproduce an action chain, we can apply the (GameSituation, Action) pair adjacent to the previously applied pair in the graph. Whether the action chain is reproduced or not is determined by the link flag. If link flag is true, the action chain is reproduced.

Hence, we use a total of four retrieval queries with different searching levels, which are the combination of two attribute sets (Level 0 and Level 1) and the link flag (True or False). Table 3 shows the details of the retrieval queries. A lower searching level means a more strict and accurate search. The ACAI searches for appropriate actions, starting with the most accurate query (search level 1) and relaxing the searching conditions sequentially. If multiple applicable actions are found, the action is selected by random selection weighted by the action counter. If no applicable action is found, the ACAI does not take any action.

Searching level	**Attribute set level**	**Link flag**
Level 1	Level 0	True
Level 2	Level 0	False
Level 3	Level 1	True
Level 4	Level 1	False

Table 3: The details of the retrieval queries

3.2 Action Filter

Following the approach described above, technically, the ACAI can decide the action to perform based on the acquired knowledge. However, the ACAI often selects weak and inefficient actions. For example, the ACAI can select to "stand on the spot." Once standing on a stick due to this decision, the ACAI learns that "it is OK to stand" and continues to stand on a stick forever. In other words, the "do nothing" action causes a loop in the acting graph. Such game situations are undesirable for players and should be avoided.

To solve this problem and further improve the decision-making performance of ACAI, we introduce an action filter into the system. The action filter analyzes the actions extracted by the ACAI and accepts only the actions that satisfy certain criteria as feasible. In contrast, actions that do not satisfy certain criteria are rejected as unacceptable by the action filter. The action filter system allows us to exclude weak and inefficient actions selected by the ACAI.

We implement the "LongNoActionFilter" as an action filter. This filter scans the stored list of the most recently executed actions. If all the actions in the list are "do nothing", the filter rejects the "do nothing" action from being selected again. In other words, the "LongNoActionFilter" prevents the ACAI from selecting "do nothing" if it has been selecting "do nothing" for a while. We can deal with the problem that the character keeps standing in place by using this filter.

3.3 AI Controller

By simply selecting the appropriate action to perform, the ACAI-controlled character cannot perform that action in the UFE. This is because the UFE does not recognize the selected action; for the ACAI to act the UFE, we need a mechanism that could be called an AI controller, which reads the input sequence assigned to the selected action and passes it to the UFE. Therefore, we have created an AI controller script. When the desired action is selected by the decision-making system, this script reads the input button sequence assigned to that action. The input button sequence is read in three separate categories: horizontal direction keys, vertical direction keys, and attack keys (see source code 1). The input sequence is passed to the UFE, and actions are performed based on it.

```
// a fragment of AI Controller
...

if(inputReference != null)
{
  if(inputReference.inputType ==
    InputType.HorizontalAxis)
    return new
    InputEvents(currentAction.horizontalAxis);

  if(inputReference.inputType ==
    InputType.VerticalAxis)
    return new
    InputEvents(currentAction.verticalAxis);

  if(inputReference.inputType ==
    InputType.Button &&
    currentAction.buttonPressed(
    inputReference.engineRelatedButton))
    return new InputEvents(true);
}

return InputEvents.Default;

...
```

Source code 1: A fragment of the AI Controller script

4 AI Evaluation

We need to evaluate the performance of the constructed ACAI. For the performance evaluation of fighting game AI, there are some methods such as checking cosine similarity or conducting a Turing test tuned for fighting games [2, 3]. In this study, we use the measure of cosine similarity to analyze and evaluate the performance of the ACAI from the aspect of similarity in behavior among players.

In the approach of checking cosine similarity, "behavior fingerprints," which are numerical vectors describing the player's behavior, are compared among players. We represent three consecutive actions of a player in the game by a tuple (A1, A2, A3). The behavior fingerprint is a list describing the occurrence probability for all possible combinations of tuples in the game, obtained by dividing the occurrence frequency of the tuple in the game by the total number of game frames. The player's action is defined by three elements: currentState, currentSubState, and currentBasicMove. Therefore, we can obtain the behavior fingerprint of a certain player by analyzing the aforementioned log file in which these elements are recorded. The cosine similarity representing the similarity between two players in the range of [0, 1] is calculated by the following formula:

$$\begin{aligned} simlarity(A,B) &= \cos(\boldsymbol{a},\boldsymbol{b}) \\ &= \frac{\boldsymbol{a}\cdot\boldsymbol{b}}{\|\boldsymbol{a}\|\|\boldsymbol{b}\|} \\ &= \frac{\sum_{i=1}^{N} a_i b_i}{\sqrt{\sum_{i=1}^{N} a_i^2}\sqrt{\sum_{i=1}^{N} b_i^2}} \end{aligned}$$

where $\boldsymbol{a}$ is the fingerprint of player A, $\boldsymbol{b}$ is the fingerprint of player B, and N is the total number of tuples.

In order to test whether the following four hypotheses are valid, we conduct an experiment to compare the behaviors. Note that the experiment is based on the conclusion from our previous research that humans have different playstyles which can be distinguished in UFE [3].

Hypothesis 1: The similarity between the ACAI and its training source player is high.

Hypothesis 2: The similarity between the ACAI and players other than the training source player is not high.

Hypothesis 3: The similarity between different human players is the same extent as the similarity between ACAIs constructed from them.

Hypothesis 4: The similarity between the ACAI and a Fuzzy AI (this AI is described below) is low.

We conduct the experiment as follows. Note that in all cases, the opponent is Fuzzy AI with the difficulty set to Normal in order to make the match conditions the same. Fuzzy AI is an AI based on fuzzy logic, which is standard in UFE. Also, note that a match is over when either player wins two rounds and that each round is a maximum of 100 seconds.

1. Three human players A, B, and C play 20 games of UFE and then create 20 log files recording their behavior and game situations for each player.
2. We construct ACAIa, ACAIb, and ACAIc learned from players A, B, and C, respectively by using these log files.
3. We make both the constructed ACAI players and the Normal Fuzzy AI play 20 games of UFE and then create 20 log files for each of them.
4. The steps up to this point create 20 log files recording the behavior of each of the seven players (A, B, C, ACAIa, ACAIb, ACAIc, and Normal Fuzzy AI) for a total of 140 log files. We divide the 20 log files of each player into two datasets of 10 files each for a total of 14 datasets.
5. We calculate behavior fingerprints and cosine similarity for all possible combinations of datasets, including those of the same player, and record the results.

5 Results and Discussion

The experiment of the similarity check yielded the results shown in Tables 4-6.

There are four ways to measure the similarity between different players. For example, we can calculate the similarity between player A and ACAIb using the following four combinations of datasets.

- Player A's dataset 1 and ACAIb's dataset 1
- Player A's dataset 1 and ACAIb's dataset 2
- Player A's dataset 2 and ACAIb's dataset 1
- Player A's dataset 2 and ACAIb's dataset 2

Table 4 shows the average values of the similarity obtained in the four measurements for each pair of players. Table 5 and Table 6 show the maximum and minimum values of the similarity obtained from the four measurements, respectively. However, note that the similarity between the same players is recorded with the same value in all three tables since the similarity check between the same players can be performed in only one way, using dataset 1 and dataset 2 of the target player. In these tables, the higher similarity is marked with a brighter green color.

Human A	0.91						
Human B	0.58	0.98					
Human C	0.62	0.73	0.92				
ACAIa	0.79	0.55	0.56	0.94			
ACAIb	0.65	0.88	0.72	0.64	0.98		
ACAIc	0.71	0.62	0.81	0.79	0.73	0.94	
Fuzzy AI	0.50	0.47	0.71	0.40	0.48	0.64	0.99
Average	**Human A**	**Human B**	**Human C**	**ACAIa**	**ACAIb**	**ACAIc**	**Fuzzy AI**

Table 4: The average similarity for each player pair

Human A	0.91						
Human B	0.62	0.98					
Human C	0.69	0.78	0.92				
ACAIa	0.83	0.56	0.61	0.94			
ACAIb	0.66	0.89	0.75	0.65	0.98		
ACAIc	0.73	0.63	0.83	0.80	0.74	0.94	
Fuzzy AI	0.53	0.50	0.72	0.44	0.50	0.65	0.99
Maximum	**Human A**	**Human B**	**Human C**	**ACAIa**	**ACAIb**	**ACAIc**	**Fuzzy AI**

Table 5: The maximum similarity for each player pair

Human A	0.91						
Human B	0.54	0.98					
Human C	0.53	0.68	0.92				
ACAIa	0.75	0.54	0.50	0.94			
ACAIb	0.65	0.87	0.68	0.62	0.98		
ACAIc	0.68	0.61	0.78	0.77	0.72	0.94	
Fuzzy AI	0.46	0.45	0.70	0.37	0.47	0.63	0.99
Minimum	**Human A**	**Human B**	**Human C**	**ACAIa**	**ACAIb**	**ACAIc**	**Fuzzy AI**

Table 6: The minimum similarity for each player pair

For all of the players in the experiment, the similarity between the same players was high, over 91%. This result means that all the players consistently fought with the same playstyles, even in different matches. Also, compared to the similarity between the same human players, the average similarity between different human players was only 58%-73%. This result confirms the conclusion of the study in [3] that human playstyles can be identified in UFE. However, since the average similarity between human B and C was 73%, and since the maximum was 78%, it should be noted that they have relatively similar playstyles on our measure.

Hypothesis 1 is correct because the average similarity between the ACAI and its teacher, the training source human player, was as high as 79%-88%. When comparing the ACAIs with the human players, all the ACAIs showed the highest similarity to the teacher player. Observations of matches in which the ACAI participated showed that it reproduced the attack actions and tactics (e.g., how to guard and keep distance) frequently used by its teacher player. We believe that the similarity was high because the ACAI has reproduced the unique human-like playstyle of its teacher players. However, this result does not support that the ACAI can perfectly reproduce the behavior of its teacher players. This is because the similarity between the ACAI and its teacher player was lower than that of the same players. In other words, the playstyles of the ACAI and the teacher player can still be distinguished.

Comparing the ACAIs with the non-teacher players, we can see that Hypothesis 2 is not necessarily true. Indeed, the similarity between the ACAIs and non-teacher players was lower than that between ACAI and teacher players. In particular, ACAIa showed a strong tendency for such similarity. However, the fact that the average similarity between ACAIb and human C and between ACAIc and human A was high (approximately 70%) contradicts Hypothesis 2. The high similarity between ACAIb and human C can be attributed to the similar playstyles of human B, the teacher of ACAIb, and human C.

Comparing the ACAIs constructed from different teacher players to each other, Hypothesis 3 is not true. If playstyles between different human players are identifiable, playstyles between different ACAIs constructed from them should be identifiable to the same extent. The average similarity between human A and human B was 58%, which was almost as low as that between ACAIa and ACAIb of 64%. Similarly, the average similarity between human B and human C was 73%, which was the same as that between ACAIb and ACAIc. However, even though the average similarity between human A and human C was 62%, it deviated from that between ACAIa and ACAIc, which was 79%, contradicting Hypothesis 3.

Since the average similarity between the ACAIs and the Fuzzy AI was only 40%-64%, Hypothesis 4 is true. The ACAIs are more human-like than the Fuzzy AI in that they showed higher similarity to the teacher players than to the Fuzzy AI. Note that the similarity between human C and the Fuzzy AI was relatively high. This is because human C used combo attacks that were similar to those frequently used by the Fuzzy AI. Accordingly, the ACAIc constructed from human C also had higher similarity to the Fuzzy AI than the other ACAIs. We assume that this was because ACAIc was trying to reproduce the combo attack used by human C, the teacher player.

Based on the results of testing the above hypotheses, we can say that although the ACAI learned the behavior of the teacher player well, the accuracy of behavior capture is still not sufficient. As mentioned above, when we actually observed the ACAI in the matches, we found that the ACAI learned and reproduced the teacher player's behavior well, especially in terms of attacking actions and tactics. We believe that such behavior of the ACAI resulted in the high similarity with the teacher player and the human-like nature which surpassed that of the Fuzzy AI. On the other hand, observation of the matches revealed that the ACAI, unlike the teacher player, often repeated the same action more often than necessary and interrupted combo attacks midway. We think that this inaccuracy of the ACAI, which still cannot reproduce the behavior accurately, resulted in the contradiction of the similarity observed in the testing of hypotheses 2 and 3.

We assume that the low accuracy of the current ACAI is due to the fact that decision-making is not working fully. If the decision-making is done correctly, the ACAI should be able to perform combo attacks without interruption or take different options instead of repeating the same action. These problems in the performance of the decision-making can be solved and improved by tuning some parameters such as the components of GameSituation, attributes used for GameSituation matching, and by adding new action filters.

6 Conclusion

In this paper, we have investigated whether it is possible to construct a human-like fighting game AI using a behavior capture system for UFE. This system records the matches and learns the behavior of human players to construct a fighting game AI that reproduces their behavior. Evaluation by the cosine similarity has shown that the constructed ACAI has relatively high similarity to the training source human player.

However, the performance of the ACAI is still not sufficient due to the fact that the current system has some issues. In addition to the issues mentioned above, the current system also has the problem that in some cases, ACAI cannot perform special attacks (e.g., Fireball) that require a button sequence of three or more inputs to execute, as intended. Our future task is to improve the performance of the ACAI by solving these issues that our system faces.

We believe that human-like fighting game AI will make matches more exciting and fun. We hope that our behavior capture system will help people to gain a more meaningful fighting game experience.

References

[1] B. Soni, P. Hingston, "Bots trained to play like a human are more fun," 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), 2008, pp. 363-369.

[2] G. Mola Bogdan, M. Mozgovoy, T. Ito, T. Rikimaru, "Believability Assessment for Fighting

Game AI," Proceedings of Game-On'2017 Conference, Carlow, Ireland, 2017, pp. 87-89.

[3] K. Yuda, S. Kamei, R. Tanji, R. Ito, I. Wakana and M. Mozgovoy, "Identification of Play Styles in Universal Fighting Engine," Proceedings of Game-On'2020 Conference, Aveiro, Portugal, 2021, pp. 72-75.

[4] M. Mozgovoy, I. Umarov, "Behavior Capture with Acting Graph: a Knowledgebase for a Game AI System," Lecture Notes in Computer Science, 2011, vol. 7108, pp. 68-77.

[5] M. Mozgovoy, I. Umarov, "Building a Believable Agent for a 3D Boxing Simulation Game," Proceedings of the 2nd International Conference on Computer Research and Development, Kuala Lumpur, Malaysia, 2010, pp. 46-50.

[6] Artificial Contender [Internet resource],
URL: http://www.trusoft.com (Date: 1.18.2022)

[7] Universal Fighting Engine [Internet resource],
URL: http://www.ufe3d.com (Date: 1.18.2022)

[8] Unity [Internet resource],
URL: https://www.unity.com (Date: 1.18.2022)