Fighting-Game Gameplay Generation Using Highlight Cues

Ryota Ishii, Keita Fujimaki, Ruck Thawonmas, *Senior Member*

***Abstract*—In this paper, we propose a fighting game AI that selects its actions from the perspective of highlight generation using Monte-Carlo tree search with three highlight cues in the evaluation function. The intended use of this proposed AI is to generate gameplay in live streaming platforms such as Twitch and YouTube where a large number of spectators watch gameplay to entertain themselves. A gameplay analysis and user study are conducted using FightingICE, a fighting game platform in an international fighting game AI competition. Results show that gameplay generated by two AI players of the proposed method has more promising characteristics than not only gameplay generated according to an existing method but also gameplay by the top two AI players in the 2019 competition and that it is more entertaining than the latter gameplay.**

***Index Terms*—Monte-Carlo tree search, live streaming, highlight generation, fighting game AI, FightingICE**

I. Introduction

In recent years, live streaming platforms such as Twitch and YouTube have been increasingly popular. A large group of spectators belong to “Let’s Play” communities [1] who watch gameplay videos to entertain themselves. As a result, this type of spectators has gained a lot of interest by researchers in various areas.

Recently, Thawonmas and Harada proposed a concept called procedural play generation (PPG) [2]. Their goal is to automatically generate gameplay according to spectators’ preferences. PPG requires a system that analyzes and recommends gameplay and a mechanism or AI that generates various kinds of gameplay which entertain different types of spectators. In our previous work on PPG, a method was proposed using Monte-Carlo tree search (MCTS) [3], [4] to generate playing styles [5] in a fighting game. We also focus on the AI part for the fighting game genre in this work.

In this paper, inspired by existing highlight generation methods that select exciting scenes for sports spectators, we propose a fighting game AI for generating entertaining gameplay where a combination of highlight indicators or cues is used in the evaluation function of MCTS. In particular, to increase the entertainment of resulting gameplay, we aim at solving an issue residing in our previous AI [6] that it was strongly biased in its action selection.

This work was supported in part by KAKENHI (19K12291). R. Ishii is now with Hitachi Channel Solution, Corp. He was with the Graduate School of Information Science and Engineering, Ritsumeikan University, Shiga, Japan, where K. Fujimaki is currently with. R. Thawonmas is with the College of Information Science and Engineering of the same university. (e-mail: ruck@is.ritsumei.ac.jp)

Manuscript received mmmm dd, yyyy; revised mmmm dd, yyyy.

The main contribution of this paper is that the aforementioned issue is solved. From experimental results, gameplay generated by the proposed AI has more variation in action execution than gameplay by our previous AI. It is also more entertaining than gameplay by top-performance AIs from the 2019 Fighting Game AI Competition (FTGAIC)[1], which are used as a benchmark.

II. Related Work

A. Monte-Carlo Tree Search in Fighting Games

Although AIs using deep learning techniques [7-9] have been recently published, MCTS, combining a Monte-Carlo method and game tree search using a given forward model, is a popular technique to implement a fighting game AI. Recent high-performance entries in FTGAIC were based on a sample AI [10] using the open loop approach [11] of MCTS. In the sample AI, to cope with the real-time property of the game, only actions of the AI player of interest are considered in the tree while actions of the opponent are randomly generated. In this work, we follow this recipe and use a separate clone of the proposed MCTS to control each AI player.

In the open-loop approach, a node stores the statistics, used in node selection described below, of a series of actions or a path from the root node to that node. As with other game tree search approaches, however, the root node represents the current game state defined by information such as the Hit-Point (HP), energy, coordinates, and action of each character and the game remaining time. An edge represents the ongoing execution of an action of interest. Four steps exist: selection, expansion, simulation, and backpropagation. They are described in the following, respectively.

1) Selection: Nodes are selected from the root node until a leaf node is reached according to a selection criterion in use. We use Upper Confidence Bounds ($UCB1$) [12], which is widely used for this task, defined by the following equation:

$$UCB1_i = \overline{X}_i + C\sqrt{\frac{2\ln N}{N_i}} \tag{1}$$

where N_i is the number of times node (action) i has been visited, N is the number of visits to its parent node, and C is a constant. In addition, $\overline{X}_i$ is the average evaluation value of the path from the root node to node i:

$$\overline{X}_i = \frac{1}{N_i}\sum_{j=1}^{N_i} Eval_j \tag{2}$$

[1]http://www.ice.ci.ritsumei.ac.jp/%7eftgaic/

where $Eval_j$ is an evaluation function returning a reward value gained in the jth simulation from the perspective of the AI player. Note that every node of the tree contains the $UCB1$ value and a counter counting how many times it has been visited. In this work, the selected path is the one that contains the nodes with the highest $UCB1$ value, from the root node until a leaf node.

2) Expansion: After a leaf node has been reached in the $Selection$ step, if the number of times it has been explored exceeds a threshold N_{max} and the depth of the tree is lower than a threshold D_{max}, all of its child nodes are created at once from it. Note that the initial tree consists of the root node and all of its direct child nodes.

3) Simulation: A simulation of the length of L_{sim} is performed by sequentially executing all the actions in the selected path while the opponent's actions are chosen for execution at random. This is done to prevent both AI players from behaving similarly since they separately use the same MCTS mechanism and at the same time to make the branching factor more manageable. If L_{sim} has not passed yet after those actions of the AI player have been executed, a rollout will be carried out until L_{sim} runs out using randomly selected actions. At the end of each simulation, the character's $Eval_j$ is calculated. Note that an opponent model can be employed to select the opponent's actions when the AI player fights against another AI player controlled by a different mechanism.

4) Backpropagation: The value of $Eval_j$ obtained in the $Simulation$ step is back-propagated from the leaf node to the root node during which the $UCB1$ value at each node along the path is also updated accordingly.

MCTS repeats the above four steps until a given time limit, T_{max}, is reached. An action is then chosen among all of the direct child nodes of the root node as the next action according to a given recommendation policy. In this work, it is action i^* that has the highest $\overline{X}_i$. As done in the open-loop approach, the chosen child node will be used as the next root node, now representing the state after the action is executed, and its sibling nodes will be pruned.

B. Highlight Generation

With a rapid increase in sports broadcasting, it is necessary to generate a highlight that allows audiences to see exciting scenes at their convenient timing. However, manual generation of highlights is time consuming, so a number of automatic methods for generating highlights have been proposed. Typically, scenes are evaluated based on several cues, and scenes with high evaluation values are selected for a highlight. For example, for boxing, a highlight can be generated based on the camera-flash timing and the distance between both players [13], i.e., a set of scenes with a close distance during flash light becomes a highlight. Because gameplay in our work is not generated while spectators are watching the game together with some also taking photos on the same location, like a boxing stadium, the camera-flash timing cue cannot be used in our work.

Recently, a method was proposed that generates a highlight for basketball based on five cues [14]: "Audio," "Score Differential," "Player Ranking," "Basket Type," and "Motion," described in the following. Audio assesses scenes according to the loudness of spectators and commentators. Score Differential considers that scenes with a narrow gap in score near the end of the game are exciting. Player Ranking selects scenes where a shot is done by a high-ranking player. Basket Type ranks scenes according to their scoring shot types. Motion prioritizes baskets with high camera motion and player motion, measured through optical flow. Due to having no spectators on-site, the Audio cue cannot be used. The concept of Player Ranking is not applicable either since in our work a separate clone of the proposed MCTS is used to control each AI player. The Motion cue is interesting. However, calculating optical flow in real-time for each MCTS simulation is not viable, so this cue is not considered in our work. Worth mentioning is more recent work by Ringer and Nicolaou [15] that takes into account information on streamers' face and audio, which cannot be applied to our work.

Our work differs from the previous studies on highlight detection or generation in that we focus on gameplay generation through highlight analysis with cues inspired by some of those in the aforementioned previous work, especially, the distance cue [13], and both Score Differential and Basket Type [14].

C. FightingICE

FightingICE is a real-time 2D fighting game platform used in the aforementioned FTGAIC and for research [5-10,16-25]. In FightingICE, a round lasts 60 seconds, and the game is rendered at 60 frames per second. Due to receiving a delayed game state from the system, an AI player does not precisely know the timing that it can perform its next action and hence it has to decide and input an action every frame, by which the previous action awaiting execution will be overridden. The HP for both characters is initially set to HP_{max} and decreases when a character of interest is hit. A round ends when the fight is conducted for 60 seconds or the HP of at least one of the two characters becomes 0. The player of the character with the larger remaining HP at the end of a round is the round's winner.

FightingICE has 56 actions of five categories: $basic$, $movement$, $guard$, $recovery$, and $skill$. Because basic actions represent neutral postures and recovery actions are automatically performed after the character has been hit by the opponent or after landing from a jump, only actions of the other three categories are typically used in AI players. Worth noting is that skill actions, used to attack the opponent, have different execution times.

For the energy of each character, it is initially set to 0 and has a maximum value of 300. A certain set of skill actions consumes energy to execute. For some skill actions, the energy of a character executing any of them will be increased when the action hits or is guarded by the opponent. When a character of interest is hit by some skill actions, its energy will also be increased. Please see the relevant information in FTGAIC's site for more details.

TABLE I: *RankAct* List in FightingICE

RankAct	Skill content	*Rank*
STAND_D_DF_FC	Throws a special fire ball	1
STAND_F_D_DFB	A hard uppercut	2
STAND_D_DB_BB	A slide kick	3
STAND_D_DF_FB	Throws a heavy fire ball	4

III. Proposed Method

In this section, we describe our proposed method for generating gameplay to entertain spectators. Our research hypothesis is that a fight between two AI players would be exciting if each AI player selects their actions based on highlight cues. This would result in a fight consisting of many highlights which are generated subject to the amount of energy of both players, causing also up and down tension during the fight.

In this work, MCTS is used, and three highlight cues are introduced to form the evaluation function of MCTS: "Action," "Score Transition," and "Distance." All of the three cues are expected to increase aggression in gameplay, which is what spectators demand in both traditional sports [26] and esports [27]. It is Action that we upgrade in this paper from our previous work [6]. Out of 56 actions available in FightingICE, 40 actions consisting of all the movement, guard, and skill actions are used. A supplementary page[2] is available that contains the list of the actions in use, the link to the source code, and other auxiliary information.

A. Action

In our previous work [6], this cue prioritized certain actions by the AI player and was defined based on our experience in organizing FTGAIC as follows:

$$Ev_{action} = \begin{cases} \frac{1}{2^{Rank-1}} & (belongs\ to\ RankAct) \\ 0 & (otherwise) \end{cases} \tag{3}$$

where $RankAct$ is a list of actions, and $Rank$ is the value associated to each action in the list. The list of actions and their rank are shown in Table I. In our previous work, hinted by the Basket Type cue [14], the actions in the list are those we considered to have high visual effects. The value of an action that is not in the list is 0.

However, as mentioned earlier, the resulting gameplay was prone to containing only few actions from the list, in particular those with low ranks. We later found out that the AI does not have sufficient energy to execute high-rank actions, especially the rank-1 action in the list, as the energy is consumed by earlier executed lower-rank actions. As a result, we propose a new mechanism for implementing this cue in the following. The basic idea behind the new mechanism is that of promoting execution of actions with high damage values but low energy consumption.

[2]http://www.ice.ci.ritsumei.ac.jp/%7eruck/hlmcts-tog.htm

$$Ev_{action} = \begin{cases} \frac{D_a}{E_a} & (\text{if } E_p \geq \tau_1 \ \&\ E_a \neq 0 \ \&\ D_a > \tau_2) \\ 1 & (\text{if } E_p < \tau_1 \ \&\ E_a = 0) \\ 0 & (otherwise) \end{cases} \tag{4}$$

where D_a and E_a are the damage value and the energy consumption of action a, respectively; E_p is the energy the player has at the end of the current simulation; and τ_1 and τ_2 are thresholds regarding energy consumption and damage, respectively.

B. Score Transition

This cue delays actions with high damage values and is defined as follows:

$$Ev_{score} = RoundTime \times Damage \tag{5}$$

where $RoundTime$ and $Damage$ represent the elapsed fight time and the damage value, respectively, both at the end of the current simulation. According to this term, the AI player prioritizes actions with high damage values, and this kind of an earnest fight is more prominent as the time is closer to the end of the round. The term is based on Score Differential [14], but has been adapted to the fighting game accordingly.

C. Distance

This cue prioritizes a close-distance fight near the center of the screen and is defined as follows:

$$Ev_{distance} = 1 - \left| \frac{center - Xpos}{center} \right| \tag{6}$$

where $center$ is the x coordinate at the center of the screen, and $Xpos$ is the AI player's x coordinate. This term will have a higher value when the AI player is positioned closer to the center of the screen. Note that if the distance between both AI players is only used, they might end up fighting at either edge of the screen, which we consider less exciting.

D. Evaluation Function

Finally, the evaluation function of MCTS in Eq. (2) is concretely defined as follows:

$$Eval_j = \frac{Ev_{action} + Ev_{score} + Ev_{distance}}{3} \tag{7}$$

where during backpropagation the value of Ev_{action} of the first action in the current path is used for the cue at that node; and min-max normalization is applied beforehand to Ev_{score} to make it range between 0 and 1; Note that the value of $Ev_{distance}$ is already in this range by its definition and that the same range can be achieved for Ev_{action} by properly setting the value of τ_2.

TABLE II: Parameters used in the experiments

Notation	Description	Value
C	Balancing parameter	1
N_{max}	Threshold of the number of visits	10
D_{max}	Threshold of the tree depth	10
L_{sim}	Simulation-time budget	60 frames
T_{max}	Execution time of MCTS	16.5 ms
$center$	Center of the game screen	480 pixels
τ_1	The energy-consumption threshold in eqn. (4)	150
τ_2	The damage-value threshold in eqn. (4)	30

IV. Experiments

In this section, we describe experiments conducted to verify the performance of the proposed method. We compare gameplay generated by different pairs of AI players as follows:

- gameplay between two of the proposed AI (newHL)
- gameplay between two of our previous AI (oldHL)
- gameplay between the winning AI (ReiwaThunder) and the runner-up (RHEA PI) of the 2019 FTGAIC.

Results from the conducted gameplay analysis and user study are given.

The last type of gameplay is introduced to examine if high-performance AIs in the competition could be directly used to generate entertaining gameplay, which was not done in our previous work [6]. As a result, their gameplay, empirically found to have the most aggression among gameplay between top entry AIs, is used as a benchmark. ReiwaThunder makes its decision based on Minimax and some predefined rules while RHEA_PI is based on the Rolling Horizon Evolutionary Algorithm with an on-line-learning opponent model. Note that we did not use gameplay of ReiwaThunder vs ReiwaThunder because both AI players were seen to perform the same actions in many situations and thus unnatural.

A. Environment

1) FightingICE: In our experiments, the value of HP_{max} was set to 400 according to the rule of the Standard Track of FTGAIC. We used the latest version of FightingICE and its only official character ZEN for the 2020 FTGAIC in this work, while using the 2019 version in our previous work [6].

2) Parameter Settings: In the experiments, the values of the parameters are summarized in Table II. The first six parameters were shared by both newHL and oldHL and set to the same values as those in our previous work [6]. For the last two parameters, τ_1 was set to the energy consumption of STAND_D_DF_FC (the rank-1 action in Table I), which has the highest amount of energy consumption and is the most destructive among the actions in FightingICE, and τ_2 was set to a value higher than the damage value of STAND_D_DF_FB (the rank-4 action in Table I) to suppress the use of it and other similar small-damage actions.

B. Gameplay Analysis

Here, we performed an analysis for each of the three types of gameplay, i.e., newHL vs newHL, oldHL vs oldHL, and ReiwaThunder vs RHEA_PI, which are henceforth called newGP, oldGP, and strongGP, respectively. In particular, we generated 1000 rounds of gameplay for each type and analyzed them using three criteria: "Mean Action Usage" (MAU), "Mean Health Variance" (MHV), and "Average Character Distance" (ACD) defined as follows:

- MAU: the mean of the number of times that each action is executed per one AI player, the left player indicated in each gameplay type, throughout its fight for all *ground* actions (executed when the AI player is on the ground) and all *air* actions (when it is in the air).
- MHV: the mean of the variance of the HP difference between the two AI players among each 100 consecutive frames
- ACD: the mean of the average distance between the two characters throughout their fight

Results and Discussions: Figure 1 shows the MAU results for the ground actions and the air actions. In the left sub-figure, it can be seen that ReiwaThunder in strongGP tends to frequently conduct STAND_B (a light kick) and STAND_FB (a higher hard kick). This is because of heuristics employed in ReiwaThunder that promote execution of these two actions and only make ReiwaThunder execute STAND_D_DF_FC only when it has a sufficient amount of energy to execute the action.

oldGP is prone to have a frequent use of STAND_D_DF_FB (shooting a strong projectile) because it is the action that requires the least amount of energy consumption in $RankAct$ and is often selected by MCTS when there is no sufficient energy to conduct the other higher-rank actions in the list. Because this skill action allows both AI players to execute it while being apart, such frequent execution of the skill action also affects both MHV and ACD of oldGP, as shown later in this sub-section.

On the other hand, newGP has a more variety in executing the actions, including those in $RankAct$, in particular, STAND_D_DF_FC, which is not seen in oldGP. The superiority of newGP over the other gameplay types with respect to the variety in executed actions can also been seen in the Shannon entropy (base-2) value with respect to the execution frequency of each action (the ratio of its MAU over the sum of MAUs for all ground actions): 3.87, 3.21, and 2.79 for newGP, oldGP, and strongGP, respectively.

The right-subfigure of Fig. 1 shows the air-action histograms for the three gameplay types. Although air actions are less frequently executed for all the gameplay types, AIR_A (a light overhead punch in the air) and AIR_B (a light overhead kick in the air) are seen more in both newGP and oldGP than in strongGP. This is because those air actions contribute to the highlight cues. The entropy values of the air actions in newGP, oldGP, and strongGP are 1.69, 1.27, 1.47, respectively, which again shows that newGP outperforms the other types of gameplay with respect to the action variety.

The MHV values are shown in Fig. 2. It can be seen that the values in both newGP and strongGP rise after a certain fight period and then, from around the 800th frame, oscillate toward the round end (the 3600th frame). In addition, both gameplay types have much higher values than that of oldGP. In other words, beginning from around one-fourth of the round, the HP difference between the two AI players fluctuates more strongly in newGP and strongGP than oldGP. The difference

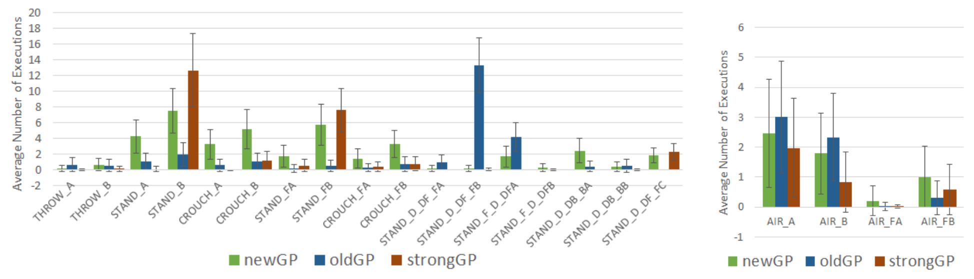

Fig. 1: Mean action usage of ground (left) and air (right) actions, where the error bars represent standard deviations

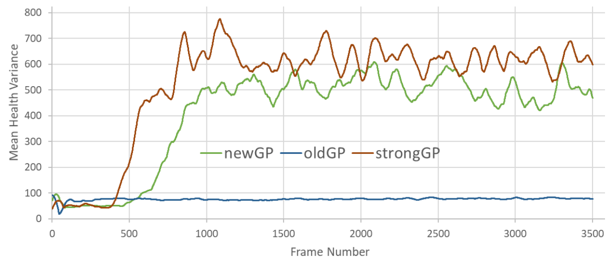

Fig. 2: Mean health variance

TABLE III: Average Distance

Gameplay	Average Distance
newGP	170.46 (15.45)
oldGP	419.37 (57.46)
strongGP	168.81 (13.22)

between newGP and oldGP stems from the difference in implementation of *Action*, by which the former has a more variety in executed actions as discussed earlier in this subsection.

ACD is shown in Table III, including standard deviation in parentheses. It can be clearly seen that close-distance fights are more pronounced in both newGP and strongGP than oldGP. Since in both newGP and oldGP the same mechanism is used to implement *Distance*, their difference also stems from the way that *Action* is implemented. In newGP, the average position (and its standard deviation) of AI Player 1, who starts from the left, and AI Player 2, who starts from the right, are 477.61 (114.22), and 482.11 (111.08), respectively. Although not shown in this paper, their histograms of the x-coordinate (cf. the supplementary page whose link has been introduced in sec. III) show that both AI players do not always position themselves near the center.

C. User Study

In this user study, we compared the fun of newGP and strongGP, selected here because of being the most promising two gameplay types from the analysis results in the previous subsection. There were 40 participants (35 male and 5 female college students, with an average age of 24.1 and a standard deviation of 1.8). The participants were asked to watch each of three pairs (cf. the aforementioned supplementary page) of two gameplay video clips (one round per clip) from newGP and strongGP. Then, given no information on how each gameplay was generated, they were asked to individually answer "Please select the most fun video" and to comment their reasons for selection. The user study was conducted on an online survey site which each time displayed a pair of gameplay video clips of the same but randomly selected round number to a participant.

TABLE IV: Votes per round

Gameplay	Round1	Round2	Round3
newGP	28	31	25
strongGP	12	9	15

TABLE V: Reasons for selection

GP	Reason
newGP	· Because there were many fights in close proximity and I could not read the development of the game · Because of flashy movement · Variation in attacks, better defense of both players · Match dominance has changed many times along the way
strongGP	· Beat the opponent in less time · The way of defense seems more reasonable · I thought that thorough fighting was good

Results and Discussions: Table IV shows the number of votes between newGP and strongGP for each of the three rounds displayed to the participants. As can be seen from this figure, newGP outperforms strongGP for all the three rounds. Representative comments by the participants are shown in Table V. In addition, according to the exact binomial test, although not statistically significant for round 3 ($p = 0.15$), the difference in the number of votes between the two gameplay types is statistically significant at the 5% level of significance for round 1 ($p = 0.017$) and at the 1% level of significance for round 2 ($p = 0.00068$). As a result, it can be said that the newGP is more fun than strongGP.

V. Conclusions and Future Work

We aim at generating entertaining gameplay for a fighting game. Our approach is based on existing cue-based highlight generation methods. In this paper, we improved our previous method, a Monte-Carlo tree search (MCTS) AI utilizing three cues forming the evaluation function of MCTS. In particular, the cue dealing with execution of actions was modified by prioritizing those actions with high damage values and low energy consumption.

Our results from the conducted analysis showed the effectiveness of the proposed method over our previous method. In addition, the conducted user study confirmed that generated gameplay by two AI players using the proposed method was more fun to watch than gameplay generated by the winning AI and the runner-up AI of the 2019 Fighting Game AI Competition. All participants in the user study were students from a graduate-level game-AI class and the authors' laboratory, leading to limitation in the population size and diversity (gender, game-playing experience, and knowledge of game AI). This is the main limitation of this work and will be addressed in the future together with other plans described below.

At present, the weights of the cue terms are equal. As future work, it might be interesting to tune them, using for example active learning [28], for certain types of spectators. We are also interested in analyses of the frequency of alternation of the leading player and the frequency of combo usage, both during gameplay, and in generation of believable gameplay [24], [29]. It is also interesting to examine how the proposed highlight-cue terms work or can be modified to work with another type of evaluation function that implements a pre-defined persona [5], a pre-defined play-arc [25], or gameplay pacing. There are other alternatives to cope with the real-time property of fighting games such as treating the game as a turn-based game like Go or as a simultaneous move game [30] as well as the mechanism employed in [25]. We plan to investigate them in future. Extension of this work to other games where MCTS-based AIs dominate is also worth exploration by introducing highlight-cue terms to their evaluation function.

References

[1] T. Smith, M. Obrist and P. Wright, "Live-Streaming Changes the (Video) Game," in *Proc. 11th European Conference on Interactive TV and Video*, ACM, pp. 131–138, 2013.

[2] R. Thawonmas and T. Harada, "AI for Game Spectators: Rise of PPG," in *Proc. AAAI 2017 Workshop on What's next for AI in games*, San Francisco, USA, pp. 1032–1033, 2017.

[3] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *Proc. International Conference on Computers and Games*, pp. 72–83, 2006.

[4] C.B. Browne, E. Powley, D. Whitehouse, S.M. Lucas, P.I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

[5] R. Ishii, S. Ito, M. Ishihara, T. Harada and R. Thawonmas, "Monte-Carlo Tree Search Implementation of Fighting Game AIs with Personas," in *Proc. 2018 IEEE Conference on Computational Intelligence and Games*, Maastricht, The Netherlands, pp. 54–61, 2018.

[6] R. Ishii, S. Ito, M. Ishihara, T. Harada and R. Thawonmas, "A Fighting Game AI Using Highlight Cues for Generation of Entertaining Gameplay," in *Proc. 2019 IEEE Conference on Games*, London, UK, 6 pages, 2019.

[7] Y. Takano, W. Ouyangy, S. Ito, T. Harada and R. Thawonmas, "Applying Hybrid Reward Architecture to a Fighting Game AI," in *Proc. 2018 IEEE Conference on Computational Intelligence and Games*, Maastricht, The Netherlands, pp. 433–436, 2018.

[8] S. Y. and K.-J. Kim, "Deep Q Networks for Visual Fighting Game AI," in *Proc. 2017 IEEE Conference on Computational Intelligence and Games*, New York City, USA, 3 pages, 2017.

[9] D.T.T. Nguyen, V. Quang and K. Ikeda, "Optimized Non-visual Information for Deep Neural Network in Fighting Game," in *Proc. 9th International Conference on Agents and Artificial Intelligence*, Porto, Portugal, pp. 676–680, Feb. 2017.

[10] S. Yoshida, M. Ishihara, T. Miyazaki, Y. Nakagawa, T. Harada and R. Thawonmas, "Application of Monte-Carlo Tree Search in a Fighting Game AI," in *Proc. IEEE 5th Global Conference on Consumer Electronics*, pp. 623–624, 2016.

[11] D.P. Liebana, J. Dieskau, M. Hunermund, S. Mostaghim, S. Lucas, "Open Loop Search for General Video Game Playing," in *Proc. the 2015 Annual Conference on Genetic and Evolutionary Computation*, pp. 337–344, 2015.

[12] P. Auer, N. Cesa-Bianchi, P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2–3, pp. 235-256, 2002.

[13] B. Nunnapus, C. Nagul, J. Chuleerat, "Flashlight and player detection in fighting sport for video summarization," in *Proc. 2005 IEEE International Symposium on Communications and Information Technology*, pp. 441–444.

[14] B. Vinay, P. Caroline, E. Irfan, "Leveraging contextual cues for generating basketball highlights", in *Proc. 2016 ACM on Multimedia Conference*, pp. 908–917.

[15] C. Ringer and M. A. Nicolaou, "Deep unsupervised multi-view detection of video game stream highlights," in *Proc. the 13th International Conference on the Foundations of Digital Games*, article no. 15, 6 pages, 2018.

[16] F. Lu, K. Yamamoto, L. H. Nomura, S. Mizuno, Y. Lee and R. Thawonmas, "Fighting Game Artificial Intelligence Competition Platform," in *Proc. IEEE 2nd Global Conference on Consumer Electronics*, pp. 320–323, 2013.

[17] X. Neufeld, S. Mostaghim, and D. Perez-Liebana, "HTN fighter: Planning in a highly-dynamic game," in *Proc. 2017 Computer Science and Electronic Engineering*, Colchester, pp. 189–194, Sep. 2017.

[18] S. Demediuk, M. Tamassia, Wi. Raffe, F. Zambetta, X. Li and F.F. Mueller, "Monte Carlo Tree Search Based Algorithms for Dynamic Difficulty Adjustment," in *Proc. 2017 IEEE Conference on Computational Intelligence and Games*, pp. 53–59, 2017.

[19] M.-J. Kim and K.-J. Kim, "Opponent Modeling based on Action Table for MCTS-based Fighting Game AI," in *Proc. 2017 IEEE Conference on Computational Intelligence and Games*, pp. 178–180, 2017.

[20] K. Majchrzak, J. Quadflieg, and G. Rudolph, "Advanced Dynamic Scripting for Fighting Game AI," in *Proc. Entertainment Computing*, pp. 86–99, 2015.

[21] K. Asayama, K. Moriyama, K. Fukui, and M. Numao, "Prediction as Faster Perception in a Real-time Fighting Video Game," in *Proc. 2015 IEEE Conference on Computational Intelligence and Games* , pp. 517–522, 2015.

[22] N. Sato, S. Temsiririkkul, S. Sone. and K. Ikeda, "Adaptive Fighting Game Computer Player by Switching Multiple Rule-based Controllers," in *Proc. 3rd International Conference on Applied Computing and Information Technology*, pp. 52–59, 2015.

[23] H. Park and K.J. Kim, "Learning to Play Fighting Game using Massive Play Data," in *Proc. 2014 IEEE Conference on Computational Intelligence and Games*, pp. 458–459, 2014.

[24] M. Ishihara, S. Ito, R. Ishii, T. Harada and R. Thawonmas, "Monte-Carlo Tree Search for Implementation of Dynamic Difficulty Adjustment Fighting Game AIs Having Believable Behaviors," in *Proc. 2018 IEEE Conference on Computational Intelligence and Games*, pp. 46–53, 2018.

[25] S. Ito, M. Ishihara, M. Tamassia, T. Harada, R. Thawonmas, and F. Zambetta, "Procedural Play Generation According to Play Arcs Using Monte-Carlo Tree Search," in *Proc. of the 18th International Conference on Intelligent Games and Simulation*, pp. 67–71, 2017.

[26] R.T. Jewell, A. Moti, and D. Coates, "A Brief History of Violence and Aggression in Spectator Sports," in *Sports Economics, Management and Policy*, volume 4, pp. 11–26, 2012.

[27] J. Hamari and M. Sjöblom, "What is eSports and why do people watch it?," in *Internet Research*, vol. 27, issue: 2, pp. 211–232, 2017.

[28] A. Zook, E. Fruchter, M. O. Riedl, "Automatic playtesting for game parameter tuning via active learning," in *Proc. the 9th International Conference on the Foundations of Digital Games 2014*, 8 pages, 2014.

[29] S. Devlin, A. Anspoka, N. Sephton, P.I. Cowling, "Combining Gameplay Data with Monte Carlo Tree Search to Emulate Human Play," in *Proc. Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*, pp. 16–22, 2016.

[30] M.J.W. Tak, M. Lanctot, and M.H.M. Winands, "Monte Carlo Tree Search Variants for Simultaneous Move Games," in *Proc. 2014 IEEE Conference on Computational Intelligence and Games*, Dortmund, Germany, 8 pages, 2014.