UDC 004.8

Kimura K., Tu Y., Tanji R., Mozgovoy M.

Identifying Winning Actions in a 3D Tennis Game with Monte-Carlo Tree Search

1. Abstract. AI nowadays plays an important role in many areas including video games. AI-controlled agents have become an essential part of video game worlds, inhabited by both intelligent opponents and friendly characters that make games more interesting and interactive. In certain types of games, notably sports games, it is important to provide opponents of different skill levels. One way to achieve such tunable AI is to employ an automated optimization method. Monte-Carlo tree search (MCTS) has been successfully used for this purpose in card and board games, such as chess and poker. We explore the possibility to apply MCTS in an action sports game of 3D tennis, and show how a dataset of pre-recorded tennis games can be used to train MCTS to overcome an entry-level built-in tennis AI system.

2. Introduction. Among methods of building game AI, the most common one is Rule-Base AI. It is simple and powerful. But it requires the implementer to understand the game logic well and be able to think about all game patterns, which is difficult.

This disadvantage can be overcome by using heuristic algorithms such as MCTS [1] to select sequentially plausible actions. In general, MCTS-family algorithms represent game process as a tree with nodes corresponding to game states. Then a certain procedure is used to explore the tree and find the most effective actions.

In this study we rely on Monte-Carlo tree search to built an AI system for a 3D tennis game.

Kimura Kaito – Undergraduate student
Yuan Tu – Undergraduate student
Tanji Riku – Undergraduate student
Mozgovoy Maxim – Ph.D., Associate professor
The University of Aizu;
e-mail: {s1250131, s1252005, s1270139, mozgovoy}@u-aizu.ac.jp;
phone: +81-242-37-2664

3. Monte-Carlo tree search method. Monte-Carlo tree search consists of the following four steps:

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

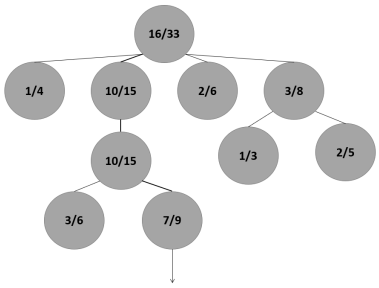

Figure 1. MCTS steps

By repeating these steps, MCTS is able to build a game tree and find branches corresponding to efficient decisions (see Fig. 2). The root node represents the current game state. From there, valid moves are explored. Each node has a record of the number of all attempts and scored attempts. The node with the highest ratio of scored attempts to all attempts is considered the most effective.

For the selection step, the UCT (Upper Confidence Trees) [4] value is commonly used. AI selects a node that has the highest UCT value from the root node to the leaf node. UCT uses the UCB1 algorithm [5] for action selection. UCB1 calculates the score of a node as

$$UCT = X_i + 2C_p\sqrt{\frac{2\ln N_i}{n_i}}$$

The first term x_i is a reward.

$$X_i = p_i + (1 - p_i)\frac{w_i}{n_i}$$

In this formula, p_i stands for the potential of the move (pattern) score directly, w_i stands for the number of wins for the node considered after the i-th move, and n_i stands for the number of simulations for the node considered after the i-th move.

The value of w_i/n_i represents a possibility of winning of this move. In general, w_i/n_i is used as the reward term. In the game of tennis, however, every shot has the potential to score directly. Therefore, we adapt Equation (2) as the reward term. The scoring possibility p_i is calculated from extracted shot pattern in the process of the simulation

The subsequent term is for biasing the search towards nodes that have not been explored enough. An abnormally high value is calculated when the number of child nodes n_i is too much smaller than the number of simulations for parent node N_i. C_p is a constant that specifies degree of bias toward unexplored nodes. If there are not enough searches for valid nodes, the searching for the node with smaller value is finished. Otherwise, the node with the highest winning rate is searched in priority.

In the expansion step, the game tree is expanded by adding child nodes if a leaf node has been explored beyond a certain threshold.

In the simulation step, the result of the game from the leaf nodes is simulated. The result of the match is reflected in all the nodes selected in the back propagation step.

By repeating these steps, the AI judges the node with the highest X_i value of reward as the valid move and acts on it.

4. Adopting MCTS for tennis. Our work aims to build an AI system for the game "World of Tennis" (see Fig. 1). Currently this game has two built-in system: one based on rules, and another based on case-based reasoning [3, 2]. Three types of basic movements are provided in the game:

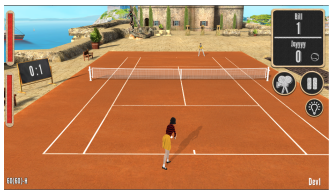

Figure 2. World of Tennis

- Serve
- Return the shot
- Recovery movement

The player and the opponent are able to serve, make shots of different types, and move to a new position after returning a shot.

In the tennis game, the game flow is a repetition of each player's action which can be represented as a game tree. In this case, each node of the game tree should contain the following information:

- Shot from: AI or opponent
- AI position

- Opponent position
- Target position

Each node contains three coordinates: the AI's, the opponent's, and the target point's coordinates. All of which are indexed by dividing the court into 48 sections (see Fig. 3) to simplify game representation with a tree.

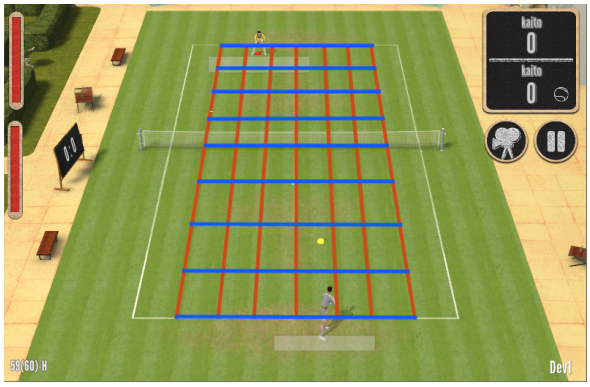

Figure 3. Tennis court divided into sections

The opponent will be in the upper court (0-23), and the AI will be in the lower court (24-47). The root node represents the current state where the opponent in position 3 is hitting the ball towards position 39. The following child nodes represent two possible effective moves

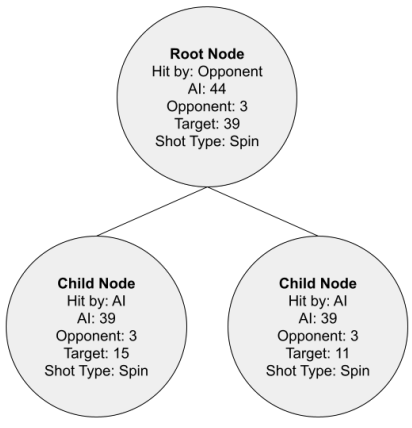

Figure 4. Node states for MCTS

for the next move,
i.e., after the AI returns the ball. Two possible next moves are both move to position 39, then return the shot to different target positions. We only showed a part of the game tree in Fig. 4, in our case, there are 24 return positions, each position has two types of shots. In total 48 nodes should be generated. In this way, the state of the tennis game is appropriately represented by the game tree.

5. Simulation logic. In MCTS, the AI simulates the game with a random selection from leaf nodes. In each simulation, the AI needs to judge whether the game will eventually result in a score or a loss of points. The accuracy of these simulations is very important in building a strong AI because it is the only standard for distinguishing strong and weak moves of AI.

In this work, more than 1400 scoring patterns from statistical data of 10,000 actual games data were extracted. The coordinates of each player and the ball, the shot types are stored as time-series data. From each match data, we extract the following information for simulation:

- Attacker (player's) position
- Defender (opponent's) position
- Target position
- Shot type
- Whether this shot is scoring or not

Player and opponent positions, shot target, intended shot type, and shot outcome directly determine the information for each game pattern.

6. Results. As mentioned above, World of Tennis has two built-in AI engines. One of them was used to create an entry-level "Coach AI" system aimed to introduce basic game patterns to beginners. We played a series of games against the Coach AI to estimate MCTS performance. As a result, MCTS-based AI won 7 games out of 10. The total score was 59:44. The scores of the matches show that the AI is able to take rational actions.

7. Conclusion. In this paper, we have created an AI for a 3D tennis game using a heuristic algorithm Monte Carlo tree search. By properly

modeling complex game states, the AI can represent game process as tree and perform the search. MCTS learns whether a move at a node is valid or not by repeatedly simulating the game state of the node. Therefore, the accuracy of the simulation is important. In this study, we conducted simulations by extracting patterns that are likely to be scored from actual game data. As a result, we were able to simulate with a certain level of accuracy and construct a working agent.

References

1. C. Browne et al. "A survey of Monte Carlo tree search methods." IEEE Transactions on Computational Intelligence and AI in games 4.1 (2012): 1-43.

2. M. Mozgovoy. Context-Awareness and Anticipation in a Tennis Video Game AI System. Proceedings of the 2018 IEEE International Conference on Systems, Man, and Cybernetics, Miyazaki, Japan, 2018.

3. Shinsei Kinouchi, "Identifying Key Elements of Successful Behavior in a Video Game of Tennis". The University of Aizu, 2019.

4. L. Kocsis and C. Szepesvari, "Bandit based monte-carlo planning," in European conference on machine learning. Springer, 2006, pp. 282–293.

5. P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," Mach. Learn., vol. 47, no. 2, pp. 235–256, 2002.