Imitation Learning for Combat System in RTS Games with Application to StarCraft

In-Seok Oh
Dept. of Computer Science and Engineering,
Sejong University,
Seoul, South Korea
ohinsuk@naver.com

Ho-Chul Cho
Dept. of Computer Science and Engineering,
Sejong University,
Seoul, South Korea
chc2212@naver.com

Kyung-Joong Kim[1]
Dept. of Computer Science and Engineering,
Sejong University,
Seoul, South Korea
kimkj@sejong.ac.kr

***Abstract*—Unlike the situation with regard to board games, artificial intelligence (AI) for real-time strategy (RTS) games usually suffers from an infinite number of possible future states. Furthermore, it must handle the complexity quickly. This constraint makes it difficult to build AI for RTS games with current state-of-the-art intelligent techniques. This paper proposes the use of imitation learning based on a human player's replays, which allows the AI to mimic the behaviors. During game play, the AI exploits the replay repository to search for the best similar moment from an influence map representation. This work focuses on combat in RTS games, considering the spatial configuration and unit types. Experimental results show that the proposed AI can defeat well-known competition entries a large percentage of the time.**

Keywords—Imitation; StarCraft; Combat; Unit Control;

I. INTRODUCTION

Imitation learning has been used widely for various types of video game (e.g., car racing, Super Mario, and Unreal Tournament) [1]. The idea is to collect the game logs of human players and use them to have bots play the games by imitation. Although game artificial intelligence (AI) developers find it attractive, it is not a trivial task to generalize this to new situations effectively. It has been reported that the imitation usually performs poorly because of the mismatch between human perception and the low-level sensory inputs to game AI bots.

In recent years, there has been much progress in the development of AI for real-time strategy (RTS) games, such as StarCraft. However, the level of AI players is still lower than that of human players and they have weaknesses in terms of handling uncertainty, build-order adaptation, strategy prediction, and long-term strategic decision-making. In particular, most bots suffer from real-time constraints complicating the use of state-of-the-art techniques for the game bots.

Previously, we applied high-level strategy prediction based on human replays with the goal of predicting the opponent's strategy from the observed information using machine learning [2]. The system learns to map units and buildings into a pre-defined high-level strategy. Although it is very accurate, it is difficult to change the build order based on the estimated one.

In this study, we propose using human replays as a source of imitation for StarCraft AI bots. The bot searches the replay repository for the most similar game scene and then follows the movements of units during that segment. In the search, we measure the similarity of influence maps between two game scenes, rather than the closeness of individual units. This reduces the computational cost of the comparison and mimics high-level human perception of the game situation.

Initially, we focus on combat in the game StarCraft. Although a human can control units very effectively, it is difficult to implement these skills in AI bots. It is necessary to determine how to coordinate the efficient movement of units while considering various values, such as the types, numbers, and locations of units. In particular, the terrain and rapidly changing situation further complicate the design of control. We evaluated the performance of our proposed bot against well-known entries in StarCraft AI competitions: FreSCBot, Skynet, and Sherbrooke.

II. PROPOSED METHOD

Our model collects human replay files of combat situations. Using replay analysis software, we can extract relevant game events from each frame. For each segment, the potential field value of the current game scene is calculated to create an influence map. The total number of searchable influence maps equals the number of replays times the average number of frames per game. These maps should be prepared before playing the game.

During the game, the AI bot asks for the influence maps stored in the replay repository that are the most similar to the current situation. Next, it imitates the movement of individual units in the most similar game situation. In this way, the programming of the AI bots can be simplified to the problem of building a replay repository and implementing imitation algorithms. To satisfy the real-time constraints, the search process should be efficient (for example, making use of indexing, hashing, and clustering). The proposed method is depicted in Figure 1.

[1] Corresponding Author
This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (2013 R1A2A2A01016589, 2010-0018950).

978-1-4799-3547-5/14/1.00

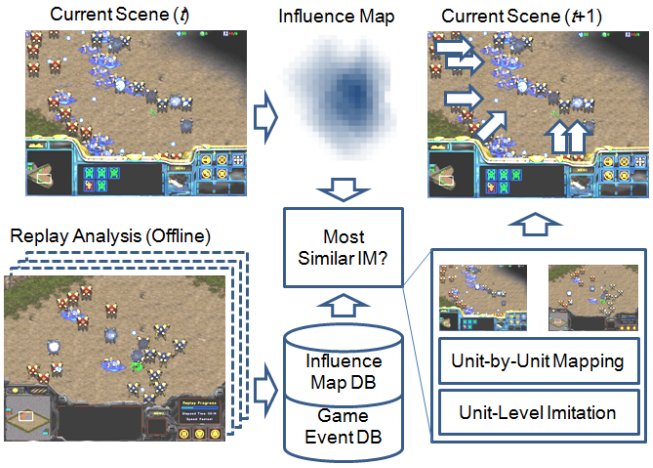

Figure 1. Overview of the proposed method.

A. *Replay Pre-processing (offline processing)*

We extract unit-level information from each scene in replays using the Brood War application programming interface (BWAPI). Table 1 shows an example of raw game events extracted from a StarCraft replay file. In addition to the basic information, it includes the current command assigned to each unit and the movement direction.

Table 1. Raw game events extracted from replays

ID	Type	Position	Health	Command	Direction
0	Dragoon	(113,1134)	180	Stop	Right
1	Dragoon	(113,1218)	180	Move	Right-Up
2	Dragoon	(147,1217)	180	Attack	Left-Up
⋮	⋮	⋮	⋮	⋮	⋮

Influence maps (IMs) have been used for RTS games and Ms. Pac-Man. They analyze the influence of units and the terrains spatially. In this process, (x, y) is the actual position of the unit and (i, j) is its position in the IM. If the unit is close to a specific point, the unit has a marked impact on it. In addition, the IM value is proportional to the unit's health. k is the number of units.

$$IM(i,j) = \begin{cases} \sum_{n=1}^{k} \left(\frac{U_n.CurrentHP}{U_n.FullHP} \right) \times \alpha, & if(\alpha \geq 0) \\ 0 & otherwise \end{cases}$$

$$\alpha = 1 - (|i - x| + |j - y|) \times 0.1$$

B. *Imitation (online processing)*

In this step, the AI bot produces an influence map of the current game situation. Next, it compares the current IM with all of the IMs stored in the database. In this work, we used a simple Euclidean distance to compare two IMs. To speed up the comparison, all of the IMs are loaded into memory. In addition, a hashing technique is used to reduce the search time. If the closest IM is found, the bot loads the associated raw game events to the IM. The raw data include all of the commands and movement directions of the units.

Naturally, the current game scene situation (the number of units, their types, and positions) is not exactly the same as the closest scene from the repository. It is necessary to map and compare the units in the current scene to the ones from the closest scene. Then, the AI bot imitates the behaviors of the closest units if the distance is within a pre-defined threshold.

III. Experimental Results

We adopted the 2010 Artificial Intelligence and Interactive Digital Entertainment (AIIDE) format of StarCraft AI micro-management tournaments. Although this is not a full-game setting, it can focus on combat mode. Human replays for imitation were collected from 40 matches between two expert human players (20 games from 12 vs. 12 dragoon battles and 20 games from 24 vs. 24 dragoon battles). The opponents were Skynet (winner of the CIG 2013 StarCraft AI competition), FreSCBot, and Sherbrooke (winner and 2nd place finisher in the AIIDE 2010 micro-management competition, respectively), and the built-in AI from the original StarCraft game. For the imitation, the bot searched 14,867 scenes stored in the DB and hashing significantly reduced the number of comparisons.

Table 2 shows the winning percentages of our imitation bot against opponents. On average, it won 84% of the games against the AI bots. It is a very promising result that bots can perform well just by imitating the behavior from expert human replays.

Table 2. Winning ratio of imitation bots against well-known AI entries (%)

Opponent	12 vs. 12 (20 games)	24 vs. 24 (20 games)
FreSCBot	80	65
sherbrooke	95	100
Skynet	85	95
Built-in AI	70	75
Avg	83	84

IV. Conclusion And Future Works

In this paper proposes an imitation learning model for StarCraft and applies it to control battle units. This imitation learning model has a substantial win rate against other well-known AI bots. It is promising because the AI bot is designed to just use replay files of human experts. This can save much time with regard to designing specific skills for the bot players. To speed up the game scene search, we introduced the use of influence map representation, and hashing techniques.

This study showed the potential of the imitation approach for StarCraft combat. It is necessary to expand the technique to include other skills, such as unit production, defense, scouting, and build-order. In addition, there is room to develop efficient new search techniques for comparing game scenes.

Reference

[1] L. Cardamone, et al., "Learning drivers for TORCS through imitation using supervised methods," *IEEE Symposium on Computational Intelligence and Games*, pp. 148-155, 2009.

[2] H.-C. Cho, K.-J. Kim, and S.-B. Cho, "Replay-based strategy prediction and build order adaptation for StarCraft AI bots," *IEEE Conference on Computational Intelligence and Games*, pp. 1-7, 2013.