Believable Team Behavior: Towards Behavior Capture AI for the Game of Soccer

Maxim Mozgovoy
University of Aizu
Aizu-Wakamatsu, Japan
mozgovoy@u-aizu.ac.jp

Iskander Umarov
TruSoft Int'l Inc.
St. Petersburg, Florida, USA
umarov@trusoft.com

This paper describes our approach used to build a practical AI solution for a simplified 2D soccer simulation game. The two main features of the designed AI-controlled team are believability (human-likeness of players' behavior) and effectiveness (capability of players to reach own goals). We show how learning by observation and case-based reasoning techniques are applied to create believable behavior. The key feature of our experiment is team behavior: individual agents (soccer players) are independent but aware of their partners' actions, and thus exhibit synchronized behavioral patterns.

1 Introduction

Most traditional AI methods are aimed at creation of efficient agents, ready to act in complex domains. The quality of an agent can be characterized in terms like robustness, reliability, and effectiveness (the capability of an agent to reach own goals). However, in the domain of computer simulations and video games, the factor of *believability* turns out to be one of the key factors of successful AI. Defined in [1] as the "one that provides the illusion of life, and thus permits the audience's

suspension of disbelief", a believable agent possesses human-like characteristics, such as capabilities to learn, to show doubts (by delaying decision making), to make mistakes, and to adjust own strategy in response to opponent's actions. Furthermore, a believable agent can exhibit its own unique behavioral style, distinct from other (both believable and non-believable) agents.

The importance of believability factor is emphasized both by researchers and by game developers [2; 3]. A clear example of AI system with high believability requirements is computer-controlled player or team in a sport game. An opponent in such a game usually exhibits own distinctive playing style in addition to pure game-playing skills. In team-based games, such as soccer, the whole team shows own "team style" [4], distinct from other teams' behavioral characteristics.

Consequently, a successful AI agent for a sport game should be able to demonstrate different playing styles in order to be believable and fun opponent to contend with. Furthermore, if a virtual computer-controlled player has a real prototype (i.e., a real athlete), an AI system should be able to reproduce his/her personal playing style.

In our recent works [5; 6], we have demonstrated an approach used to create a believable and effective agent for a 3D boxing video game. Our AI system uses a combination of learning by observation and case-based reasoning technologies. A computer system first observes a human expert, who demonstrates desired behavior by actually playing the game, then acts according to the formed knowledgebase.

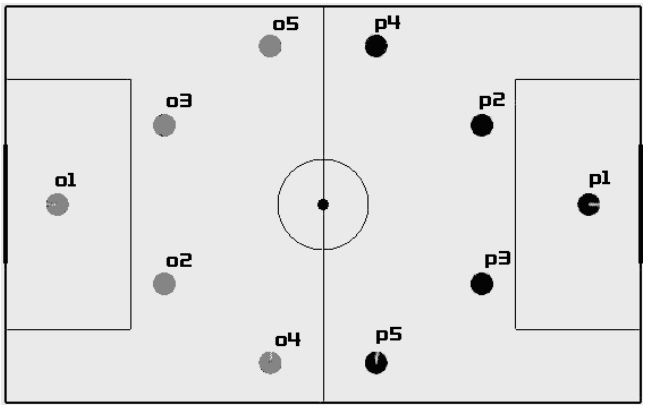


Figure 1. Five-a-side 2D Soccer Simulator

We have shown that the obtained AI-produced behavior is believable and shares distinctive behavioral patterns with its prototype (a human player). Next, we have applied reinforcement learning to increase the agent's skill level, still preserving believability of agent's behavior [6].

After these experiments, we wanted to generalize our approach to teams of AI agents in order to obtain distinct believable team styles. However, the game of boxing cannot serve as an appropriate environment for testing team behavior, since the participants of a boxing match are individual opponents.

Thus, we had to select another testbed for experimenting. Currently we use a simplified five-a-side 2D soccer simulator (see Figure 1), similar to the one used in the RoboCup Initiative (2D Simulation League) [7]. While our simulator cannot be considered as a full-fledged commercial-level game environment, its capabilities are sufficient to test the quality of the proposed AI solution.

2 Basics of the Behavior-Capture Engine

Similarly to our 3D-boxing virtual player, the soccer AI system follows the philosophy of TruSoft's Artificial Contender AI middleware (www.trusoft.com), and makes use of TruSoft's AI SDK. Thus, the soccer agent acts according to the general scheme described in [5].

2.1 Learning and Acting

The game engine can operate in two different modes: learning mode and acting mode. In learning mode, a behavior-capture agent (referred to below as *BC Agent*) observes the actions of the "trainer", in most cases represented by a human expert. Every time the observed character makes an action (including "do nothing" case), BC Agent stores the executed action, paired with the representation of the current state of the game world, in its knowledgebase. In our soccer simulator, the actions of individual players are restricted to movements in eight possible directions, passes, and kicks towards the goal line.

In acting mode, BC Agent uses its knowledgebase to retrieve the most suitable action for a given game world state upon request from the game engine.

2.2 Identifying BC Agents

A BC Agent for the 3D boxing game was attached directly to one of the competing players. In team-based games, such as soccer, it is necessary to decide first what will be considered as a single decision-making entity. We have examined the following scenarios in our experiments:

Player with the ball. A BC Agent is attached only to the player, currently possessing the ball. When the ball moves to another player, the BC Agent takes control over it. All other players are controlled by an alternative (non behavior capture-based) AI system.

"Puppet master." A BC Agent controls the soccer team as a whole. An atomic action to be stored in the knowledgebase incorporates all actions of the individual players.

Players as separate agents. A separate BC Agent is attached to every player on the game field.

The first scenario is handy for smooth transition from a single-player game (boxing) to a multi-player game (soccer) environment. We used "Player with the ball" to test the applicability of our behavior-capture approach to the game of soccer, avoiding complications, related to team behavior.

The "Puppet master" approach was rejected as being potentially fragile. The case-based reasoning subsystem tries to find the best approximation for the current game

world state in its knowledgebase, so by increasing the number of parameters in the game state description, we decrease chances of a successful search. If BC Agent controls just one soccer player, its game state description may include multiple precise attributes for the player's close surroundings, while storing only several rough values for the distant game field locations. If all players are equally represented in the game state description, as in case of "puppet master", the number of stored attributes may increase significantly. Moreover, in real computer games this approach requires more training sessions, including the ones that expose unlikely scenarios, such as the absence of one of the players (due to the red card).

Thus, the most important experiments were conducted within "Players as separate agents" scenario. It gave us the possibility to train the players independently and to analyze how successfully they can achieve team goals.

2.3 Knowledge Representation

Game world states and player actions are stored in plain data structures GameSituation and Action. GameSituation structure stores the values of the most important game-field numeric attributes, selected by the developers. Game state is described "through the BC Agent's eyes", so the most important attributes for the BC Agent are stored with higher precision, while other attributes (such as faraway players' positions) are represented with rough estimations or even omitted.

Each Action object stores the parameters needed to perform the corresponding game action. Since we experiment with "player with the ball" and "players as separate agents" models, each Action refers to the activities of an individual soccer player.

When learning new cases, a BC Agent memorizes incoming (GameSituation, Action) pairs and creates the links between subsequent records, forming a directed graph-like knowledgebase that contains action chains.

Each action chain corresponds to a certain individual game session. In addition to core action parameters, each Action object contains a usage counter that is increased every time when the same (GameSituation, Action) pair arrives at the input. This counter is later used for a weighted action choice (more frequent actions are selected more often, when possible).

The decision making mechanism is considerably more complex than the learning subsystem, because it contains an additional heuristic search routine. Ideally, when a virtual agent receives the next GameSituation object from the game engine, it should extract the same GameSituation from the knowledgebase. However, in order to obtain reliable decision making, we cannot assume that this search is always successful. Therefore, the agent should be able to extract the closest (if not perfect) match from the knowledgebase and act in accordance with the found (GameSituation, Action) pair.

In our system, approximate matching is achieved through a series of knowledgebase polls with sequentially relaxed search conditions. We have two types of such relaxations: (a) excluding certain developer-specified GameState attributes from comparison; (b) performing approximate comparison (the attributes are considered equal if their difference is less than a certain developer-specified value).

Thus, BC Agent usually finds several suitable actions, and selects the most reliable one, found under the strictest search criteria. When several actions have the same degree of reliability, the agent selects one of them randomly (using weighted random choice [8] with action usage counters as weights).

The advantages of this graph-based scheme in comparison with other approaches are discussed in [6].

2.4 Action Filters Subsystem and Planner Filter

As explained in [5], the set of relevant actions, retrieved by the case-based reasoning module during decision making, is further narrowed by the system of action filters.

Action filters increase decision making quality by providing simple hints on game's basic principles to the agent. A single action filter analyzes all actions, extracted by action selection subsystem and rejects an action (marks as unacceptable) or lowers its weight if it does not satisfy certain preprogrammed criteria.

For example, the case-based reasoning system may consider the current game situation GS_1 to be similar to the game situation GS_2, found in the knowledgebase, while the action "pass the ball", successfully performed in GS_2, can turn out to be too risky in GS_1. Even when the differences between the game situations can be minor, they can be crucial for the given action: small coordinate change of one of the defending players can make the pass easy to intercept.

Most filters for the game of boxing were relatively simple and straightforward, such as "try not to stay in the corner of the ring", "try to avoid clinch with the opponent." For the game of soccer, we decided to develop a special *planner filter* that introduces basic planning capabilities to our system.

The game of boxing did not require any planning; our experiments showed that acting only on the basis of the current game situation is sufficient to create effective agents. In contrast, soccer requires some advance planning and awareness of possible outcomes of the chosen action.

The planner filter works as follows. First it constructs the game situation GS_{new}, representing the probable outcome of the currently analyzed action A. In other words, it tries to guess the situation on the game field after A is applied. Since A contains all the information about the BC Agent's intended action, the system can predict the changes in the agent's attributes. The new positions of the opponents and the ball are simply extrapolated using their current speeds and directions. Then the filter checks whether the case-based reasoning subsystem can extract actions for the GS_{new}. If all these actions are rejected by the action filters subsystem, the predicted game situation GS_{new} is considered unfavorable, and the planner lowers the weight of A.

Since the game graph stored in the knowledgebase contains agent's future action sequences, the planner filter can use this information to predict game situation that may occur after a whole chain of agent's actions. In actual experiments, we limit the planner's horizon with a certain time-based value (typically, 2-3 seconds).

3 Experiments

To test our system, we followed the same scenario, as in case of 3D boxing game [5]. We have identified 16 parameters, describing an individual player's view of the game world. The most important among them are:

- player's position (coordinates);
- the "danger of moving forward" value that represents the probability of losing the ball on the next move towards the opponent's goal;
- the direction and distance towards the opponent's closest player;
- the player's current movement direction;
- the ball's state (in the air / possessed by a player);
- the location of the player, possessing the ball.

This parameter set forms our GameSituation data structure. The first experiments conducted in "player with the ball" mode were aimed at proving the ability of our system to play soccer. The human expert, controlling player with the ball, played 8000 frames of the game time (around 400 secs). Then two different BC Agents (BC_1 and BC_2) were trained on 2000, 4000, 6000, and 8000-framed samples of the human-provided data. Then we compared the performance of the human-controlled player with BC_1 and BC_2 using three indicators:

- the average number of goals scored per 2000 frames of play;
- the average number of player's passes, intercepted by the opponent's team per 2000 frames of play;
- the average percentage of player's team ball possession per 2000 frames of play.

The remaining team members as well as opponent's players were controlled by a simple rule-based AI system. The results (see Figure 2) show high abilities of BC Agents to reproduce human-generated behavior patterns.

Our next goal was to obtain believable team behavior by experimenting with "players as separate agents" scenario. Probably, the most important indicator to be checked is the algorithm's robustness, i.e. its capability to reproduce team behavior patterns under changing conditions.

In its most realistic form, our experiments were conducted as follows. Five independent human experts control respective soccer team players via network interface. Each computer runs the local copy of the same soccer simulator, and the games are synchronized across the local network.

We have played several games (5-10 minutes each), then analyzed team behavior of the BC Agents in the acting mode. It worth mentioning technical difficulties of such experiments: even human experts need some pre-training before they are able to act effectively as a team. They also need some preliminary agreement on tactics and carefully planned combinations to be used in various scenarios of the game.

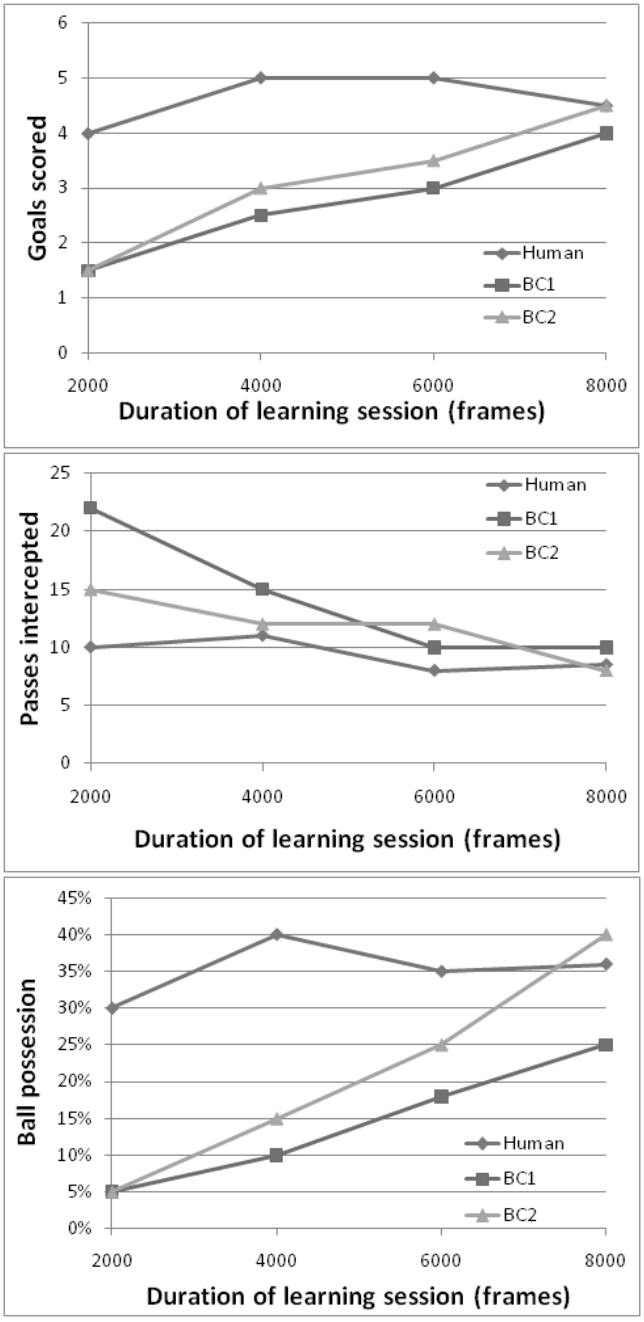


Figure 2. BC Agent's learning process

Currently, we can report the following preliminary results: the teams of independent BC Agents are able to reproduce relatively complex combinations

during the game. While players do not communicate directly, their actions are synchronized by the case-based reasoning system, since every player's GameSituation includes the location of the player possessing the ball. If the team of BC Agents performs an attack, it has the ball in possession, so all BC Agents act in synchrony with the player with the ball.

One of the most complex combinations, reproducible by the team of BC Agents (players P1-P5, see Figure 3), includes seven ball kicks:

1. P5 charges O5 and gets the ball.
2. P5 passes the ball to P4 in order to avoid O5's tackle.
3. P4 passes the ball to the defender P2, completely avoiding the danger of being tackled by O5.
4. P2 makes a short run, and passes the ball to P3. Meanwhile, P4 moves towards the center of the game field, and P5 runs to the opponent's goal.
5. P3 returns the ball to P2.
6. At this time, O5 is no longer a threat to P4, so P2 passes the ball to P4.
7. P4 quickly transfers the ball further to P5.
8. P5 scores a goal.

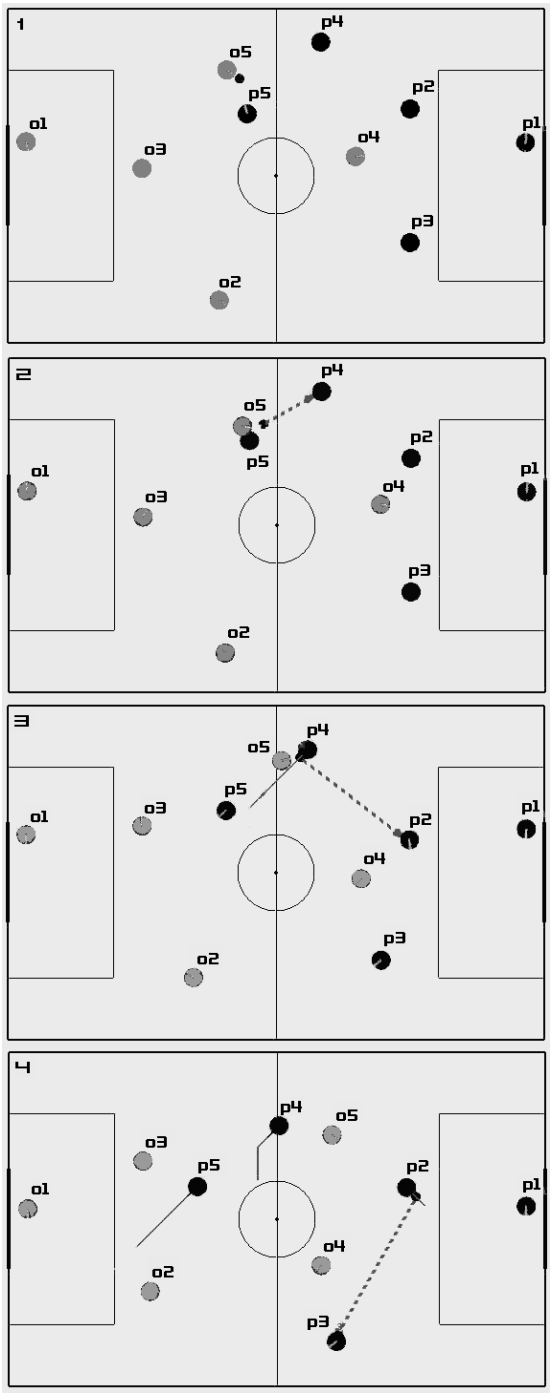
As it can be seen, scoring a goal requires considerable team efforts, including actions, performed in strict synchrony.

4 Conclusion

We have adapted our earlier behavior capture AI system for the game of 2D soccer and performed basic experiments with team behavior. The previous version of the system was tested in a simpler environment of boxing game, without any requirements of team acting and advance planning. The purpose of the current research is to prove that our approach is feasible for the task of team-based AI development.

First, we have developed the AI system that controls the player, currently possessing the ball, while all other players are controlled by the simple rule-based AI mechanism. This system was able to provide believable and effective behavior, comparable to performance of the human expert, acting under the same conditions.

Next, we have performed a series of experiments with a team of independent but team-aware BC Agents. While the results are still preliminary, they clearly show that the team of BC Agents is able to act in synchrony, reproducing complex game combinations and reaching common goals.


1
p4
o5
p5
p2
o1
o3
o4
p1
p3
o2
2
p4
o5
p2
p5
o1
o3
o4
p1
p3
o2
3
p4
o5
p5
o1
o3
p2
p1
o4
p3
o2
4
p4
o5
o3
p5
p2
o1
p1
o4
o2
p3

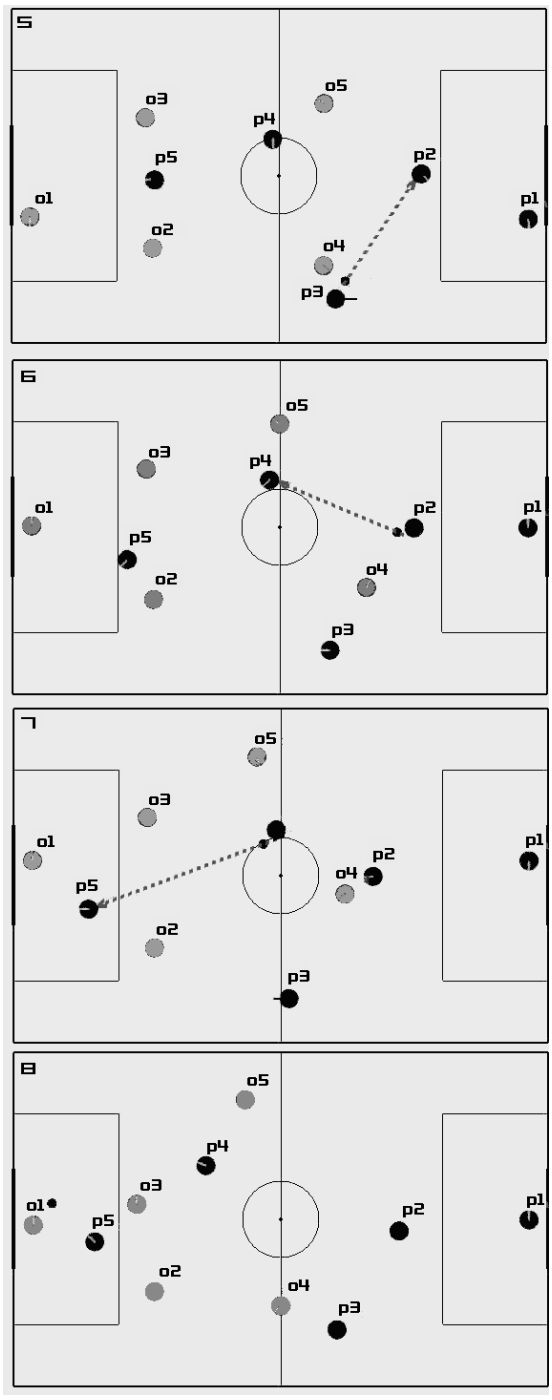


Figure 3. Combination performed by a team of BC Agents

Bibliography

[1] J. Bates, "The role of emotion in believable characters," *Communications of the ACM*, 1994, v. 37, pp. 122-125.

[2] G. Yannakakis and J. Hallam, "Towards optimizing entertainment in computer games," *Applied Artificial Intelligence*, 2007, v. 21, pp. 933-972.

[3] J. Orkin, "Three states and a plan: the AI of FEAR," *Game Developers Conference*, 2006.

[4] ExpertFootball.com. Soccer Styles of Play. *http://expertfootball.com/coaching/styles.php*

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

[6] M. Mozgovoy and I. Umarov, "Building a Believable and Effective Agent for a 3D Boxing Simulation Game," *Proc. of the 3rd IEEE International Conference on Computer Science and Information Technology*, 2010, v. 3, pp. 14-18.

[7] RoboCup. Soccer Simulation League. *http://wiki.robocup.org/wiki/Soccer_Simulation_League*

[8] F. Olken and D. Rotem, "Simple random sampling from relational databases," *Proceedings of the 12th International Conference on Very Large Data Bases*, 1986, pp. 160-169.