# Team: *Kasuga-bitos* with Modulation of Playing

Tomoo Inden and Tomoichi Takahashi

Chubu University, 1200 Matsumoto Kasugai-shi Aichi 487, JAPAN

**Abstract.** Two points are paid attention in implementing our team *Kasuga-bitos*. One is an agent changes its playing style as a game proceeds. The other is that agents cooperate each other using only visual information without any programmed protocols. Their effects are estimated by games with Ogaltes, the champion team in PreRoboCup 96.

## 1 Introduction

A soccer game is a kicking game which object is to score more goals than an opponent team. A soccer game is played with two teams. A team is composed of 11 players. One player is a goalkeeper. The goalkeeper is a special player who is in front of the goal and defends his goal from the opponent team's attack. The other 10 players play in the field to get a goal. The game is mainly made up by the players in the field.

The techniques of players are classified into several levels.

**fundamental techniques:** Each player should master techniques of controlling the ball. The controlling techniques are kicking, dribbling, passing, intercepting, etc.

**advanced techniques:** Good players can decide immediately which play is most suitable to changing situations. They shift from one technique to another smoothly, for example from dribbling to passing.

**tactics of game:** Players should cooperate each other to get a goal and to defend their goal. Coaches dictate players how to move cooperatively during the game.

The techniques in the first two levels are players' abilities. The last level refers to the abilities as a team. These techniques are common to real games and simulation games.

This paper discusses fundamental and advanced techniques in section 2. In real soccer games, coaches outside of the field suggest players to move in a better way. Playing style of a soccer agent also should reflect the situations which change as a game proceeds. Player's style modulation are discussed in section 3. In section 4, the agent movement is discussed in terms of computer resource allocation problem. Our playing style modulation methods are discussed using games between our team, *Kasuga-bitos*, and Ogalets. Ogalets is the champion team in PreRoboCup. [1]

## 2 Player's techniques

Agents in simulation track are provided with fundamental techniques as control commands such as `kick`, `dash`, etc. The agent should be programmed

- to select suitable commands and to set their parameters,
- to combine the fundamental techniques.

### 2.1 Players basic model

Each player decides these procedures in primitive (`sense - plan - action`) cycle. The planning procedure is modeled as a finite state automaton ($Q$, $\Sigma$, $\delta$, $d$, $g$). Fig 1 shows state transition diagram, $\delta$. Circles indicate player's states, $Q = (a, b, c, d, g)$, where $d$ is an initial state and $g$ is a final state.

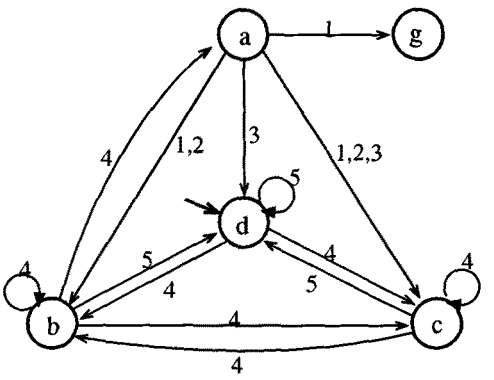


**Fig. 1.** state transition diagram

**a:** The agent can kick the ball. The parameters of `kick` - direction, power - are calculated by the player's position.
**b:** The ball is near the agent. The agent should `dash` to control the ball.
**c:** The ball is out of the region which is assigned to the agent. The agent returns to and stays the default position.
**d:** The ball is out of the agent's sight. The agent `turns` to look for the ball.
**g:** The ball is in the goal.

Table 1 shows input alphabet $\Sigma$. Input alphabet is sensor information edited in a five-tuple form - (shoot condition, pass condition, obstacle-in -sight condition, ball-in-sight condition, ball-in-field condition). The agent can shoot or pass if shoot condition or pass condition is satisfied. Obstacle-in-sight and ball-in-sight conditions indicate agents of the opponent team or the ball is in the sight.

**Table 1.** sensor conditioning input alphabets

| condition | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| *shoot* | o | × | × | × | - |
| *pass* | - | o | × | × | - |
| *obstacle-in-sight* | - | - | o | - | - |
| *ball-in-sight* | o | o | o | o | × |
| *ball-in-field* | o | o | o | o | o |

o: condition satisfied.
×: condition not satisfied.

Ball-in-field condition indicates whether the ball is dead or not. The condition in right columns constrain more globally than the condition in left columns.

The finite state automaton decides an agent's action sequence as a real player predicts consequences of his actions. The other players move autonomously. As a result, the decided action is not suitable for the new situation. The player's action sequence does not necessarily follow the transition diagram of the automaton.

## 2.2 Implementation of advanced techniques

Pass is to kick a ball to another player of own side. Pass' success is judged by whether another player receives the ball or not. Communication between players is required to succeed passes. There are two ways for agents to communicate with each other.

- using voice(`say-command`), one agent send information to other agents.
- using vision(`see-information`), agents know the situations.

Using `say-command` as acknowledge procedure is a reliable procedure, but it takes time and other players may be farther than voice can reach. We think that most of kicking and receiving a ball in real game are done see-information. [1]

The other technique of pass is the direction of `kick`. Real players kick the ball with predicating the next position of the other players. In addition to the kick player's prediction, receiving players expect that the ball will be passed when the other player's state is the same one as their kick state. These predictions are similar to the agents movements in the simulation game. Fig.2 shows one of such situations. There are three agents ,`a1`, `a2`, `a3`. They dash to a ball and one of them (for example, `a1`) holds it. Then the other two players should stop the dash and change other actions. They may turn and dash to the goal, excepting the ball will be passed to oneself.

[1] We admit this is disputable. According to Mr. Ando, the runner-up in RoboCup-97, *Andhill* use `say-command` as to inform other players the next action.

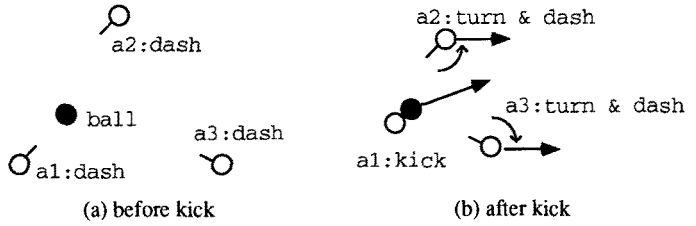


**Fig. 2.** Example of advanced techniques

## 3 Playing style modulation

Game tactics vary in the process of a game. The tactics as a team controls playing style of a player who moves autonomously.

### 3.1 Factors of playing style

There are several factors to change the playing style. The followings are examples of such factors.

- distance to ball:
  The player near the ball should play quickly to get a goal or to defend his goal. On the other hand, the players far from the ball may move more slowly. They keep their positions for the next coming situations.
- gap between score of teams:
  When score gap increases as the game proceeds, coaches give suggestions to the players from outside of the field in the real games. The suggestions are "mark the player number X", or "change attack or defense formation to Y", etc. The coach suggestion model should be given in advance and the agents change their own playing styles using the suggestion model.
- elapsed time of game:
  When games are coming to end and the score of our side is over the opponent, the players are inclined to play defensively to keep the lead. To the contrary, the players play offensively in a case that the score is vice versa.

### 3.2 playing styles

There are several playing styles. Table 2 shows examples of styles.

The first position style is determined by the player's position. The second play style reflects player's characteristics. These two styles are the same ones as real players have. They are assigned to the client when it is called.

The last style indicates how quickly players move. In real games, quick movement is one of good player's characters [2]. It is related to each player's physical

[2] soccer server version 3.05 later has a parameter, indicating an agent's stamina.

**Table 2.** examples of playing style

| *factor* | playing style | |
|---|---|---|
| *position* | forwards | backs |
| *play type* | offensive | defensive |
| *response* | quickly | slowly |

strength, while the agent's quick movement is also related to communication rare to the soccer server in the RoboCup games.

The last two styles change during the game. They are modeled as a function of distance to the ball and elapsed time. The combination of the modes control the agent's global movement.

# 4 Resource Allocation Problem

## 4.1 Players Sensing-Action Cycle

Including an opponent team, 22 clients join the game. The agent knows its position and game situation through communications with the soccer server. The communication are done in a form of `(sense - plan - action)` cycles with following conditions.

**c1:** `sense` and `action` are associated with communication with the soccer server.
**c2:** queuing to the soccer server is equal to all clients.
**c3:** CPU time allocation is equal to all clients.

The player's speed is proportional to communication rate between the corresponding client and the server. Under amount of communication is restricted, the game is resource allocation game among clients as well as a game of kicking. [2]

## 4.2 Tactics to allocation problem

Response factor indicates one tactics that players nearer to the ball move more quickly. In order to play quickly, the following requirements are satisfied.

**a1:** Agents near the ball should communicate with server more frequently than other agents.
(Agents near the ball should have higher priority than other agents.)
**a2:** Our team should communicate more frequently than the opponent.

Our idea is to allocate more resource than other agents by controlling the agent's CPU running time. Fig 3 shows the heuristic function between sleep command parameter and distance to ball. If the distance between agent and the ball is below fileld_with/6, then the agent is not slept. The distance is greater than fileld_with, one sleep unit in plan cycle is allocated to the agent.

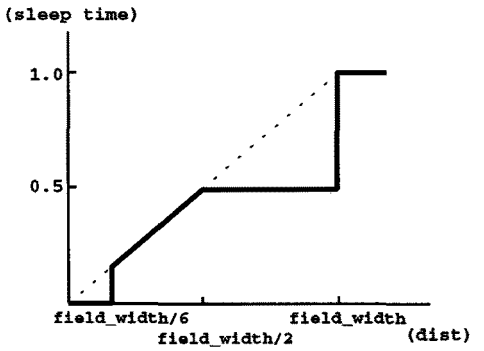


**Fig. 3.** sleep ratio vs. distance.

## 5 Experiment Result

Table 3 shows the games history of between *Kasuga-bitos* and Ogalets.[1] Ogalets is a champion team in PreRoboCup in 1996. Clients for game 1 are programmed by using only the automaton in Fig 1. Response mode changing by controlling CPU running time is installed in clients for game 2. Clients in game 3 are capable of passing the ball.

**Table 3.** Scores of games between Ogalets

| *No.* | score | *implemented skills* |
|---|---|---|
| *1.* | 0-15 | only fundamental ones. |
| *2.* | 0-9 | response mode. |
| *3.* | 4-3 | positioning, passing between agents. |

The game records show that

1. *Kasuga-bitos* can decrease the score gap in No.2 match. This score imply that response mode is effective,
2. In No.3 match, the agents can pass a ball and try to keep its position in a better point. This makes *Kasuga-bitos* a team comparable to Ogalets.

## 6 Summary and conclusion

Ball games players move differently as the situation changes. Our principle in making *Kasuga-bitos* is to imitate real player's motion, - its play style changes

in a process of game and the information comes mainly from its eyes. In this paper, a player's techniques is classified into three levels and the relationship between response of player's movement and resources allocation are discussed. Movement response modulation and passing skill are implemented. Their effects are estimated in games with Ogalets.

The scores show that they make *Kasuga-bitos* a good team, however it is not sufficient to be an excellent team. In the first stage of RoboCup-97, *Kasuga-bitos* belong to Group D , where the champion team *AT Humboldt* and the runner-up *Andhill* are. Matching them makes the difference to the excellent teams. Modifying finite state automaton model during games is our next purpose.

Finally, we would like express our gratitude to the RoboCup's committee for holding this Cup.

## References

1. http://ci.etl.go.jp/ noda/soccer/client.html
2. Noda Itsuki, Kuniyoshi Yasuo. *Simulator track and Soccer Server.* bit, Vol.28, No.5, pp.28-34.