Robust Algorithm for Safety Region Computation and Its Application to Defense Strategy for RoboCup SSL

Taro Inagaki, Akeru Ishikawa, Kazuhito Murakami, and Tadashi Naruse

School of Information Science and Technology, Aichi Prefectural University, Nagakute-cho, Aichi, 480-1198 Japan

Abstract. We have proposed a new concept of "safety region" which we use to measure the position of the defense robots[5]. It is defined as a region that the teammate robot(s) can defend the goal when an opponent robot shoots the ball from the inside of the safety region while teammate robots are positioned according to their defense strategy.

Since it is difficult to obtain the accurate safety region in a short time, we need an algorithm that computes an approximate safety region in real time. We proposed such algorithm in the previous paper[5]. However, the safety region obtained by the algorithm is not accurate enough. Therefore, in this paper, we propose an improved algorithm to compute the approximate safety region. We have achieved 95% accuracy and less than 1 msec of computation time, which is adequate for our RoboCup application. We also propose a defense strategy based on the safety region considering the positions of the opponent robots and the pass direction. The achieved results indicate accurate performance for determining the positions of the defense robots.

1 Introduction

In the recent RoboCup Small Size Robot League(SSL), strategies for attacking and defending are growing higher, and the strategy that dynamically changes the number of defense robots depending on the game's situation is often used. In a typical SSL strategies, the potential field[2] and the playbook[3] are used for the action selection and deciding the number of defense robots. Furthermore, cooperative plays such as a direct play [6] are commonly used in recent SSL games. To defend such plays, it is necessary to compute the situation of the game in real time and to determine the positions of the defense robots.

There are some measures for determining a mark robot[4] and deciding a passing robot[7]. We need such a measure for defense robots. We proposed a new concept of "safety region"[5]. It can be used as a measure for determining the positions of the defense robots. We described an algorithm to compute the safety region in real time when defense robots are placed according to the team's defense algorithm, and also we showed an algorithm to place a new (adding) defense robot robot under the measure of safety region[5]. However, the safety region computed by the algorithm is less accurate than the true one. Therefore, improved algorithm is desirable.

In this paper, we propose an improved algorithm to calculate the approximate safety region. It achieves 95% accuracy with respect to true safety region and less than 1 msec of computation time for SSL applications, which is adequate for our purpose. We also propose an improved defense strategy. This is based on the safety region considering the positions of the opponent robots and the pass direction. The results show that it works well for determining the positions of the defense robots.

2 Safety Region

In this section, we define the "safety region" and describe the algorithm to compute the safety region.

2.1 Definition

A concept of "safety region" is simple. It is defined as a region that the teammate robot(s) can defend the goal when an opponent robot shoots the ball from the inside of the region while teammate robots are positioned according to their defense strategy. Remaining region of the field given by removing the safety region is called "unsafety region".

In the following discussion, we do not consider the chip and curved shots.

2.2 Calculation of Safety Region

The calculation of the safety region depends on how the shot action is taken, i.e. a single or assisted shot, and how the team keeps the goal, i.e. defense strategy and the number of defense robots. It takes much time to compute the accurate safety region. Therefore, we describe procedures to compute the approximate safety region. In the following, we discuss the computation model of the safety region for a direct play, which is a play that the first robot kicks the ball to the second robot and the second robot kicks it directly toward the goal. The single shot by the first robot is modeled as well.

Computation Model

Defense robots will move according to their strategy so that we assume the right positions of the defense robots are given at any time. Let $\mathbf{b}$ and $\mathbf{e}$ be the positions of the ball and the shooting robot at time t, respectively, and $\mathbf{r}_i$ be the position of each defense robot i at time t. Let L_r and L_l be the lines connecting $\mathbf{e}$ and the right goalpost and $\mathbf{e}$ and the left goalpost, respectively. (See Fig. 1.) We assume the goalkeeper stands in the defense area and moves along the border line of the defense area while other defense robots stand outside of the defense area and move along the border line. Then, let d_i be the distance between $\mathbf{r}_i$ and border line (, usually equal to the radius of the robot), and let A_i be the curve along with the border line with distance d_i (dotted line in Fig. 1). Let $\mathbf{p}_{r,i}$ and $\mathbf{p}_{l,i}$ be

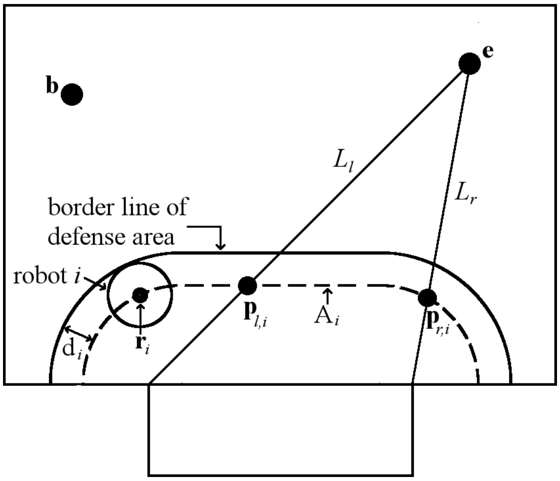

Fig. 1. Definition of symbols

the cross-points between L_r and A_i, and L_l and A_i, respectively. Let $D_{r,i}$ and $D_{l,i}$ be the distance along A_i between $\mathbf{r}_i$ and $\mathbf{p}_{r,i}$, and $\mathbf{r}_i$ and $\mathbf{p}_{l,i}$, respectively.

Assume that the passing robot holds the ball at time t and makes the direct play. Then following equation is obtained for computing the safety region.

Let's define,

$$t_p = \frac{||\mathbf{e} - \mathbf{b}||}{v_p}, \quad t_{j,s} = \frac{||\mathbf{p}_{j,i} - \mathbf{e}||}{v_s} \ (j = r \text{ or } l), \quad t_i = \frac{v_i}{a_i}, \tag{1}$$

where t_p and $t_{j,s}$ are the times for the ball to move from $\mathbf{b}$ to $\mathbf{e}$ with velocity v_p, and from $\mathbf{e}$ to $\mathbf{p}_{j,i}$ with velocity v_s, respectively, and t_i is the time for the robot i to get to its maximal velocity v_i under the condition of maximal acceleration a_i and initial velocity 0.

If $t_i > t_{j,s}$, then compute

$$D_{j,i} < \frac{1}{2}a_i(t_p + t_{j,s})^2 + R_i \quad (j = r, l) \tag{2}$$

otherwise compute

$$D_{j,i} < \frac{1}{2}a_i {t_i}^2 + v_i(t_p + t_{j,s} - t_i) + R_i \quad (j = r, l) \tag{3}$$

where, R_i is the sum of radii of the defense robot i and the ball.

If Eq. (2) or (3) is satisfied for $j = r$ and l or there is no pass line between $\mathbf{b}$ and $\mathbf{e}$, $\mathbf{e}$ is a point in the safety region. In case there are more than one defense robot, $\mathbf{e}$ is a point in the safety region if at least one of them satisfies Eq. (2) or (3) for $j = r$ and l.

For each point on the field, we compute the above equations and obtain the safety region. However, computation time for this process is high and more optimal solutions is desirable for real time performance.

2.3 Reduction of Computation Time

To reduce the computation time, we use the coarse-to-fine method. The algorithm to compute the safety region using the coarse-to-fine method is shown below. First, assume that the field consists of $W \times H$ grids. We assume W and H are 2's powers here. Prepare an array $F[W][H]$ corresponding to the field. The suffix(i, j) of $F[j][i]$ represents the field position. Let N be a given number ($N = 2^n$).

Algorithm. Computing the safety region based on the coarse-to-fine method

1. Put a ball on a given point **b**.(See Fig. 1.)
2. Set $j = 0$, $i = 0$.
3. Set N to 2^n. Compute the algorithm shown in section 2.2 for (j, i) position, where (j, i) is a position **e** in Fig. 1. Set $F[j][i]$ to 1/0 according to the computation result (safety/unsafety). Also compute the algorithm for $(j, i + N - 1)$, $(j + N - 1, i)$ and $(j + N - 1, i + N - 1)$, respectively, and set the corresponding F.
4. If all four points are in the same region (safety/unsafety), then all the points $F[k][l]$ between $j \leq k < j + N$, $i \leq l < i + N$ are set to the safety/unsafety value. Otherwise, set N to $N/2$ and compute steps 3 and 4 recursively for four divided parts until N comes to 1.
5. Set i to $i + N$. If $i < H$, then goto step 3.
6. Set i to 0 and j to $j+N$. If $j < W$, then goto step 3, otherwise, computation finishes.

3 Experiment

In this section, we show the experimental results of the safety region computation for the direct play[6]. We use the defense strategy of RoboDragons[1] since we know all its details.

3.1 Method of Experiment

The safety region should be calculated analytically. However, it is hard to do the analytical computation, therefore, we calculate it on each of the grid points which are given by dividing the field every 40 mm. We compute the approximate safety region using the model discussed in section 2.2.

To evaluate the correctness or preciseness of the approximate safety region, we need a true safety region. It will be given by the experiment using the real robots. However, it requires (ultra) heavy use of the real robots, causing the breakdown of robots. It is not a good strategy to do such experiments. Instead, we use the simulator of the RoboDragons system which simulates the behavior of the Robots based on the physical law. (Final evaluation should be done in the competition using the system implemented by the proposed algorithm, although has not been conducted in this paper.) The simulation procedure for the direct play is given as followings:

1. Divide the field into n grid points, where $n = 14888$ in this experiment. Put the ball on the initial position $\mathbf{b}$, one of the grid points. Also put the attacking robots on the grid points, one around the ball and the other on a given point $\mathbf{e}_i$, a shooting position[1]. Place defense robot(s) on defending position(s) according to the strategy algorithm of the RoboDragons system.
2. At time t, move the ball from $\mathbf{b}$ to $\mathbf{e}$ with the passing velocity v_p.
3. Move the ball on one of two shooting lines, L_r or L_l in Fig. 1, which is the farthest line to the defense robots at time t_e, the time that the ball arrives at shooting position $\mathbf{e}$.
4. Simulate the movement of the defense robots and judge whether a goal is achieved or not. If the goal is achieved, then the point $\mathbf{e}$ is a point in the unsafety region, otherwise a point in the safety region.
5. Repeat steps 2 ... 4 for each grid point $\mathbf{e}$.

We call a safety region obtained by the above procedure a simulated safety region.

The initial position of the ball used in the simulation is selected from the logged data of the 8 games held in RoboCup Japan Open 2009 and RoboCup 2009. Kick off points, and direct and indirect free kick points are the candidates of the initial position of the ball. The initial position is randomly selected from the candidates. Passing velocity and shooting velocity of the ball are 4.0 m/sec and 8.0 m/sec, respectively, which are the typical values of the robots used in the SSL. The acceleration and velocity of the defense robots are 2.0 m/sec^2 and 0.6 m/sec, respectively, which are the measured values.

3.2 Experimental Results

Coincidence Rate

We compared the approximate safety region with the one obtained by the simulation for the two defense robots under the RoboDragons's defense strategy.

Figs. 2 and 3 show the examples of the experimental results. In the figures, an approximate safety region is placed on a simulated safety region, and the red region is an unsafety region in simulation (simulated safety regin) but not in calculation (approximate safety region), the green an unsafety region in calculation but not in simulation, the blue and white an unsafety region and a safety region, respectively, both in simulation and in calculation. From the view of the fail-safe computation, it is desirable that the red area is 0 and the green area to be smallest. To evaluate these areas, we define the following rates:

$$R_c = \frac{A_b + A_w}{A}\ , \quad R_a = \frac{A_g}{A}\ , \quad R_s = \frac{A_r}{A}\ , \quad A = A_b + A_w + A_g + A_r \qquad (4)$$

where A_b, A_w, A_g and A_r are the area of blue, white, green, and red region, respectively. We call R_c the correspondence rate. R_a and R_s are the rate of green and red regions, respectively. Table 1. shows the result of those rates which are

[1] We assume that the robot can kick the ball toward any direction.

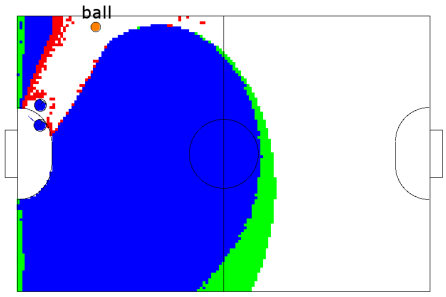

Fig. 2. Safety region: example 1

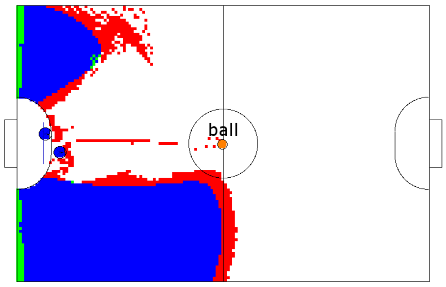

Fig. 3. Safety region: example 2

the average of 10 trials under the condition of 2 defending robots. From Table 1, we think we obtained a considerably accurate algorithm for safety region computation. However, there are still room for improving the algorithm as can be seen in Fig. 3.

Table 1. Rate of the area of blue+white, green and red region

R_c	R_a	R_s
0.951	0.013	0.036

Computation Time

Using the two typical computers, we measured the computation time for the proposed method. Table 2 shows the computation time and the coincidence rate R_c which is the average of 10 trials under the condition of 2 defending robots and using coarse-to-fine algorithm.

Table 2. Computation time and coincidence rate : Coarse-to-fine method

N	computation time(msec)		coincidence rate
	Athron64 X2	Xeon 3.3GHz	R_c
don't use	10.8	6.6	0.950
1	8.5	4.5	0.951
2	3.1	1.4	0.951
4	1.4	0.56	0.951
8	0.94	0.37	0.951
16	0.84	0.32	0.951
32	0.78	0.30	0.951

3.3 Discussion

Judging from our experience to have developed the RoboDragons system, the computation time of the safety region should be less than 1 msec. From Table 2, we can greatly reduce the computation time by using the coarse-to-fine algorithm. We can achieve the time of less than 1 msec without reducing the coincidence rate if $8 \leq N \leq 32$.

Table 2 shows that there are no differences in coincidence rate R_c for various values of N. This is because the initial position of ball is taken from the free kick positions or kick-off positions as described in section 3.1. Note that we should pay attention to the value of N when the initial ball position is around the goal area. Figs. 4 and 5 are examples of resulting safety region. In this case, we cannot get a correct safety region if $N \geq 8$. This fact should be considered when we construct a defense algorithm.

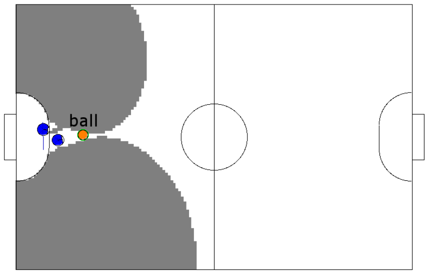

Fig. 4. Safety region obtained by using coarse-to-fine method: N = 2

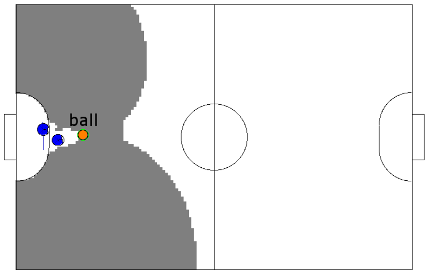

Fig. 5. Safety region obtained by using coarse-to-fine method: N = 8

4 Defense Algorithms Using the Safety Region

4.1 A Defense Algorithm Considering the Position of the Opponent Robot

A defense algorithm (or an algorithm for positioning the defense robots) is proposed in the reference[5]. In the algorithm the unsafety region is weighted by the position of the opponent robots. Example is shown in Figs. 6 through 8. Fig. 6 shows an unsafety region (gray area) where 3 opponent robots (yellow) and 2 defense robots (blue) are placed. Fig. 7 shows a weighted unsafety region. The darker the area, the higher the weight. Then, we would like to add one defense robot. Where should it be placed? The algorithm proposed in [5] gives its position. As a result, Fig.8 is obtained and the unsafety region is reduced considerably. However, is the proposed algorithm good if the shooting robot is the one at upper-left corner? We must consider the pass direction of the passing robot. In the next section, we define such an algorithm.

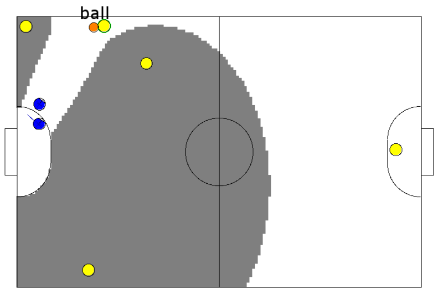

Fig. 6. Safety/Unsafety region

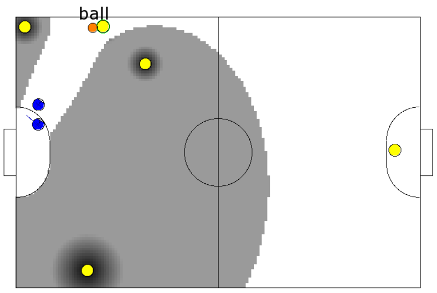

Fig. 7. Weighted unsafety region

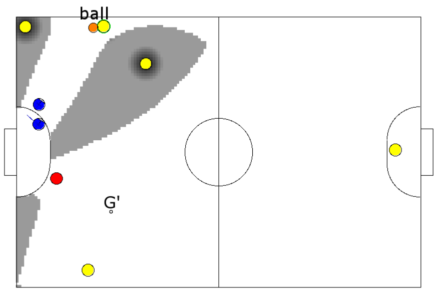

Fig. 8. Safety region: After 3rd defense robot (red circle) is placed

4.2 A Defense Algorithm Considering the Position of the Opponent Robot and the Pass Direction

In a real game, a pass direction is an important factor to evaluate the situation. Therefore, we propose a new algorithm to calculate the safety region considering both the pass direction and a weight function. In the following, we call a line connecting the center of the passing robot and the center of the ball a "pass line".

First, compute safety/unsafety region for already placed defense robots and then compute the weighted unsafety region employing a similar way used in the reference[5]. The weight function $w(\mathbf{e})$ is defined as the following equation for each point $\mathbf{e}$ in the unsafety region.

$$w(\mathbf{e}) = \sum_i max(1, 100 \times (1 - \frac{||\mathbf{r}_i - \mathbf{e}||}{max(M, t_p \times v_i)})) \qquad (5)$$

where, $\mathbf{r}_i$ and v_i are a position and a moving speed of the opponent robot i, respectively, and t_p is a time given by Eq.(1). M is a threshold value to keep the weighting area wide when the value of $t_p \times v_i$ is small. We use $M = 270$ in the experiment. The value of $w(\mathbf{e})$ takes the range between 1 and 100.

Next, the pass line is considered. Let L be the line connecting the ball and the point $\mathbf{e}$. Then, compute the angle θ that the pass line and the line L make. If the angle is small, the value of weight function $w(\mathbf{e})$ should be high. So, we define the following modified weight function $w'(\mathbf{e})$.

$$w'(\mathbf{e}) = \begin{cases} w(\mathbf{e}) \times (max(1, 10 \times \cos(3 \times \theta))) & (|\theta| \leq \pi/6) \\ w(\mathbf{e}) & (|\theta| > \pi/6) \end{cases} \tag{6}$$

The Eq. 6 shows that the weight is higher if the angle $|\theta|$ is less than $\pi/6$ radian. The weight in Eq. 6 ranges from 1 to 1000. The closer the point $\mathbf{e}$ goes to $\mathbf{r}_e$ or the pass line, the higher weight is gained.

We propose the following algorithm that decides the position of (n+1)th defense robot.

Algorithm. Defense robot placement

1. Let $\mathbf{r}_i$ be a position of the defense robot i $(i = 1 \ldots n)$. Calculate safety and unsafety regions. Number each connected component of unsafety region. Let it be $N_k^{'}(k = 1 \ldots)$.
2. For each point $\mathbf{e}$ in the unsafety region, compute the weight $w'(\mathbf{e})$.
3. For each connected component $N_k^{'}$ of the unsafety region, add up all weights of the points in that connected component.
4. Calculate a center of gravity **G'** of the connected component $N_m^{'}$ which has the biggest summation given by step 3.
5. Place the (n+1)th robot at the cross-point $r_{n+1}^{'}$ of the line $L_{g'}$ and the border line of the defense area (a bit outside of the defense area), where $L_{g'}$ is a bisector line of the maximal free angle from **G'** toward the goal.

4.3 Discussion

For the same initial positions of robots and the ball as in Fig. 6, the proposed algorithm in section 4.2 is applied. As a result, the weighted safety region shown in Fig. 9 is obtained. In Fig. 9, the pass line goes toward the robot at the upper-left corner. After placing the 3rd defense robot, which is shown in the red circle in Fig. 10, computation gives a new safety region shown in Fig. 10. The figure shows that the shooting robot is in the safety region.

In Fig. 6, even if the pass line goes toward the other opponent robot, the proposed algorithm provides good placement of the defense robot. Examples are shown in Figs. 11 and 12.

In Fig. 11, the passing robot next to the ball is facing the robot at bottom near **G'**. The defense robot placement algorithm gives the 3rd defense robot position and resulting safety region is shown in Fig. 11. When the passing robot turn to face the robot at upper middle of the field, the algoritnm obtains the

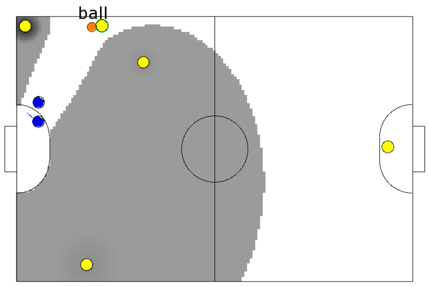

Fig. 9. Weighted unsafety region using the proposed algorithm

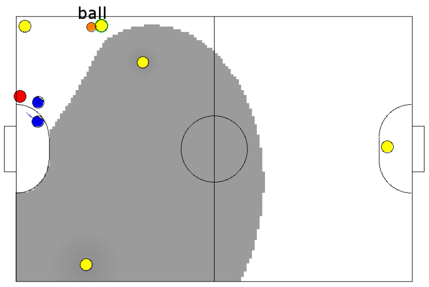

Fig. 10. Safety region: After 3rd defense robot (red circle) is placed by the proposed algorithm

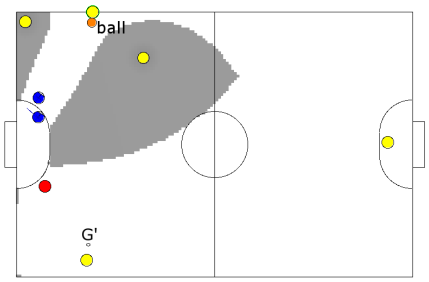

Fig. 11. Safety region: in case the passing robot faces the robot at bottom

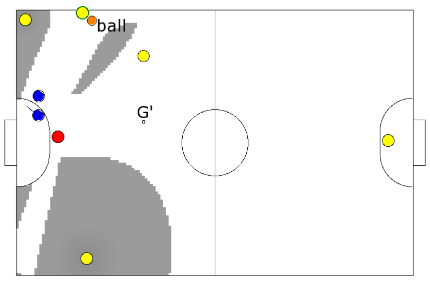

Fig. 12. Safety region: in case the passing robot faces the robot at upper middle

3rd defense robot position as shown in Fig. 12. Resulting safety region is also shown in Fig. 12.

It is important to note that this algorithm decide the position of the defense robot depending on the **G'**. For example, assume that, at first, 3rd defense robot (red circle) is placed on the position shown in Fig. 11, and then the passing robot turns toward another robot shown in Fig. 12. In this case, the 3rd defense robot can move to the position in Fig. 12 in a short time since the moving distance is short because of the gravity center comes between two opponent robots in the connected component of the unsafety region. This is a greate advantage of this algorithm.

5 Concluding Remarks

We proposed the safety region as an index for evaluating the situation of the game in the SSL. The safety region is a region that the defense robots kept the goal when an opponent robot shoots in that region. We proposed an improved algorithm that computes the approximate safety region. We have achieved 95% accuracy and less than 1 msec of computation time, which is adequate for our RoboCup application. We also proposed an improved defense strategy using the safety region index considering the positions of the opponent robots and the pass direction. The results indicates accurate determination of the defense robots position.

Future works are

1. to improve the accuracy of the approximate safety region,
2. to reduce the computation time of approximate safety region,
3. to implement the proposed defense strategy into the RoboDragons system and evaluate the algorithm using the real robots.

Acknowledgement. The authors thank Mr. Kentaro Mizushima for his valuable comments and Dr. Salman Valibeik for his proofreading. A part of this work was supported by the chief director's special study fund of Aichi Prefectural University and the president's special study fund of Aichi Prefectural University.

References

1. Achiwa, H., Maeno, J., Tamaki, J., Suzuki, S., Moribayasi, T., Murakami, K., Naruse, T.: RoboDragons 2009 Extended Team Description (2009), `http://small-size.informatik.uni-bremen.de/tdp/etdp2009/small_robodragons.pdf`
2. Ball, D., Wyeth, G.: UQ RoboRoos 2004: Getting Smarter. In RoboCup 2004 Symposium CDROM, SSL Team Description Papers (2004)
3. Bowling, M., Browning, B., Chang, A., Veloso, M.: Plays as Team Plans for Coordination and Adaptation. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS (LNAI), vol. 3020, pp. 686–693. Springer, Heidelberg (2004)
4. Laue, T., Burchardt, A., Fritsch, S., Hinz, S., Huhn, K., Kirilov, T., Martens, A., Miezal, M., Nehmiz, U., Schwarting, M., Seekircher, A.: B-Smart (Bremen Small Multi Agent Robot Team) Extended Team Description for RoboCup (2009), `http://small-size.informatik.uni-bremen.de/tdp/etdp2009/small_b-smart.pdf`
5. Maeno, J., Ishikawa, A., Murakami, K., Naruse, T.: Safety region: an index for evaluating the situation of RoboCup Soccer game. JSAI Technical Report. SIG-Challenge-B001-2 (2010), `http://winnie.kuis.kyoto-u.ac.jp/sig-challenge/SIG-Challenge-B001/SIG-Challenge-B001-2.pdf`
6. Nakanishi, R., Bruce, J., Murakami, K., Naruse, T., Veloso, M.: Cooperative 3-Robot Passing and Shooting in the RoboCup Small Size League. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006. LNCS (LNAI), vol. 4434, pp. 418–425. Springer, Heidelberg (2007)
7. Zickler, S., Bruce, J., Biswas, J., Licitra, M., Veloso, M.: CMDragons 2009 Extended Team Description (2009), `http://small-size.informatik.uni-bremen.de/tdp/etdp2009/small_cmdragons.pdf`