Motion Planning Algorithms for Tactical Actions in Robot Soccer

Andras Gyorgy, Istvan Harmati

***Abstract*—Robot soccer became a challenging area in computational intelligence and machine learning, including disciplines like real-time image processing, path planning, control and obstacle avoidance. In this uncertain and highly dynamic environment precise and fast actions are required. Defining dominance areas and implement primitives like kick, pass and dribble are crucial with the aspect of computational time reduction. The hard computing analytical solutions presented in this paper allow accurate and rapid actions that are essential for a successful robot soccer strategy.**

I. Introduction

THE development of robots satisfies not only industrial needs but entertainment and research ones as well. A remarkable example is robotic soccer that gained great popularity among scientists in the last two decades.

The Robot World Cup (RoboCup) is a great challenge for scientists of every specific area: multi-agent cooperative systems, computer vision, artificial intelligence, collision detection and obstacle avoidance are all in focus when implementing a robot soccer team. The original goal was to create a team of humanoid players that could defeat the winner of the most recent world cup for human players by the mid-21st century. In this paper, we present precise and time saving methods implemented in MATLAB which were successfully applied in our own robot soccer simulation testbed. By using analytical solutions instead of numerical ones, low-level computations are very fast, hence more time remains for the strategy level which is essential in real-time robot soccer. Another crucial requirement is accuracy which is provided by avoiding approximations.

In the last two decades many papers were published about robot soccer. As for historical aspect, see [1], [2] and [3] that cover the RoboCup Championships and the most successful teams. Some papers like [4] provide a general overview of the subject, while [5] and [6] are helpful for implementing an own robot soccer simulator with teams.

Path planning and collision strategies are described in [7], [8], [9] and [10], while soft computing based methods are presented in [11] and [12]. Interesting game theoretical approach is proposed in [13] and other stimulating ideas can be seen in [14], [15] and [16].

Manuscript received November 2, 2008. This work was supported by the Hungarian Science and Research Fund under grant OTKA K 71762.

Andras Gyorgy is with the Department of Control Engineering and Information Technology, Budapest Univerity of Technology and Econoimcs, Budapest, 1111 Hungary (e-mail: andras.gyorgy2@gmail.com).

Istvan Harmati is with the Department of Control Engineering and Information Technology, Budapest Univerity of Technology and Econoimcs, Budapest, 1111 Hungary (e-mail: harmat@iit.bme.hu).

In this paper, we provide hard computing solutions to important tactical skills such that passing, dribbling and kicking the ball. In order to carry out tactical motion planning, our proposed method intensively relies on Voronoi regions and dominated areas.

This paper is organized as follows: in Section II the proposed method is introduced and explained in detail. Section III and Section IV show some of our latest experimental results and conclusions drawn from them.

II. Proposed Method

A. Simulation Environment

The architecture of the software (Fig. 1) can be naturally divided to six modules. The Environment Parameters unit contains global parameters like pitch size, goal size and ball damping factor. The Central Server is the core that does the computation work and provides interfaces to the Display and the Hardware modules. The Team Shell is the interface between the Central Server and Team Strategy modules whose task is assuring transparency, for example executing coordinate transformation since the teams use their own coordinate systems.

The size of the gamefield is FieldX and FieldY. The applied simulation time quantum is Ts (which means this is the sample period of the simulation), whereas the Central Server asks the Team Shell for control sequences in Tc time ($Tc > Ts$), which means the Team Strategy module has to provide long sequences of output. The ball movement can be modeled by a geometric series, where the quotient q denotes the ball damping factor.

The physical agents are round differential drive robots so the path is realized in three steps: rotation to the target, move to the target and rotation to the desired angle.

As for the hierarchical structure of the Team Strategy module, it contains three subsystems: Low Level Motion Planning in obstacle free space is the level of solving the differential equations of kinematic constraints, while Tactical Motion Planning contains primitives like kick, pass and dribble that are used by High Level Strategy.

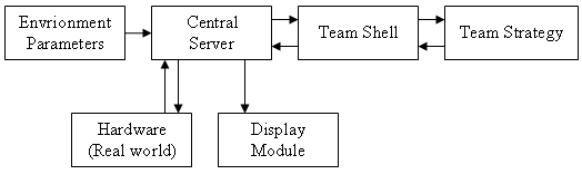

Fig. 1. Software architecture

978-3-9524173-9-3

B. Ball Collision with Wall

When the ball reaches the wall it bounces back with different velocity. In case of elastic collision, magnitude and parallel component remain the same, only normal component changes to its negative. Continuous monitor of possible collisions with the wall requires precious time that should be reduced.

In order to decrease computation capacity requirement we apply a simple and fast transformation (Table I) by adding virtual playzones to the original gamefield (Fig. 2). In this manner, it can be seen that the agents and the ball move in the grown pitch, after that we transform the coordinates back to the real pitch.

C. Agent Collision with Ball

Since agents are not able to catch the ball, they have to manipulate it so that it bounces to the right direction with the desired speed. Consequently, the most important primitive is the kick.

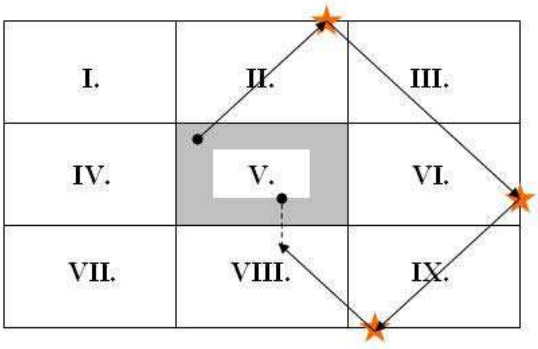

Fig. 2. Extended gamefield. Pitch V. is the original whereas the others are the virtual ones. Ball bounces three times: their detection requires great computation capacity. Instead of this, applying the simple and fast coordinate transformation only once means more time remains for strategy.

TABLE I
TRANSFORMATION WHILE COLLISION WITH WALL

Pitch	Position		Orientation	
	x	y	ϑ	ψ
I.	$-x'$	$2FieldY - y'$	$-\vartheta'$	$-\psi'$
II.	x'	$2FieldY - y'$	ϑ'	$-\psi'$
III.	$2FieldX - x'$	$2FieldY - y'$	$-\vartheta'$	$-\psi'$
IV.	$-x'$	y'	$-\vartheta'$	ψ'
V.	x'	y'	ϑ'	ψ'
VI.	$2FieldX - x'$	y'	$-\vartheta'$	ψ'
VII.	$-x'$	$-y'$	$-\vartheta'$	$-\psi'$
VIII.	x'	$-y'$	ϑ'	$-\psi'$
IX.	$2FieldX - x'$	$-y'$	$-\vartheta'$	$-\psi'$

x and y are x and y coordinates in the original gamefield, while ϑ and ψ are angles to x and y axes

x' and y' are x and y coordinates in the extended gamefield, while ϑ' and ψ' are angles to x and y axes

The problem can be formulated as follows: we know the velocity of the ball before collision that has to be modified to a certain value and orientation. In order to execute this we can use the velocity of the agent before collision. Its velocity after collision is indifferent.

From physical constraints (conservation laws for impulse and energy):

$$mv_{x,1} + M\omega_{x,1} = mv_{x,2} + M\omega_{x,2}, \tag{1}$$

$$mv_{y,1} + M\omega_{y,1} = mv_{y,2} + M\omega_{y,2}, \tag{2}$$

$$\frac{1}{2}m\left(v_{x,1}^2 + v_{y,1}^2\right) + \frac{1}{2}M\left(\omega_{x,1}^2 + \omega_{y,1}^2\right) = \frac{1}{2}m\left(v_{x,2}^2 + v_{y,2}^2\right) + \frac{1}{2}M\left(\omega_{x,2}^2 + \omega_{y,2}^2\right), \tag{3}$$

where x and y in footnote denote orthogonal components of velocity vectors, whereas m and M denote the mass of ball and agent, respectively. As it can be seen in Fig. 3, v and ω represent the linear velocities for the ball and the robot before and after collision.

Considering physical constraints, we have three simple equations but four unknown variables, so we need one more equation, which is very complex in general case. By a rational simplification, the required fourth equation will take an uncomplicated form, so solving the nonlinear system becomes much easier and faster.

By fixing the agent velocity direct to the ball before collision (Fig. 3), we can define variables relative to its moving center. In this manner, it is like the agent stands still while the ball moves with the same velocity component parallel to the axle of collision and increased velocity component perpendicular to it. In the aspect of the ball, it can be seen like collision with the wall, so parallel velocity component remains unchanged while perpendicular turns to its negative. In this way, parallel impulse of the ball remains the same, so the same impulse component of the agent can not change either. This means that after collision the parallel velocity component of the agent is the same as before, which was zero. Consequently, the fourth constraint is

$$\frac{\omega_{x,1}}{\omega_{y,1}} = \frac{\omega_{x,2}}{\omega_{y,2}} \tag{4}$$

in this special case. It is remarkable that the equation is very simple, but the physical simplification is not too great, which is a rather beneficial trade-off.

Now the unknown variables are

$$\omega_{x,2} = \omega_{x,1} + \frac{m}{M}\left(v_{x,1} - v_{x,2}\right), \tag{5}$$

$$\omega_{y,2} = \omega_{y,1} + \frac{m}{M}\left(v_{y,1} - v_{y,2}\right), \tag{6}$$

$$\frac{\omega_{x,1}}{\omega_{y,1}} = \frac{\omega_{x,1} + \frac{m}{M}\left(v_{x,1} - v_{x,2}\right)}{\omega_{y,1} + \frac{m}{M}\left(v_{y,1} - v_{y,2}\right)}, \tag{7}$$

$$\omega_{y,1} = \omega_{x,1}\frac{\left(v_{y,1} - v_{y,2}\right)}{\left(v_{x,1} - v_{x,2}\right)}. \tag{8}$$

Substituting (5)-(8) to (3) we get the final result, the desired agent velocity components before collision:

$$\omega_{x,1} = \frac{M\left[\left(v_{x,1}^2 - v_{x,2}^2\right) + \left(v_{y,1}^2 - v_{y,2}^2\right)\right]}{2\left[\left(v_{x,1} - v_{x,2}\right) + \frac{\left(v_{y,1} - v_{y,2}\right)^2}{v_{x,1} - v_{x,2}}\right]} - \frac{m\left[\left(v_{x,1} - v_{x,2}\right)^2 + \left(v_{y,1} - v_{y,2}\right)^2\right]}{2\left[\left(v_{x,1} - v_{x,2}\right) + \frac{\left(v_{y,1} - v_{y,2}\right)^2}{v_{x,1} - v_{x,2}}\right]}, \tag{9}$$

$$\omega_{y,1} = \frac{M\left[\left(v_{x,1}^2 - v_{x,2}^2\right) + \left(v_{y,1}^2 - v_{y,2}^2\right)\right]}{2\left[\left(v_{y,1} - v_{y,2}\right) + \frac{\left(v_{x,1} - v_{x,2}\right)^2}{v_{y,1} - v_{y,2}}\right]} - \frac{m\left[\left(v_{x,1} - v_{x,2}\right)^2 + \left(v_{y,1} - v_{y,2}\right)^2\right]}{2\left[\left(v_{y,1} - v_{y,2}\right) + \frac{\left(v_{x,1} - v_{x,2}\right)^2}{v_{y,1} - v_{y,2}}\right]}. \tag{10}$$

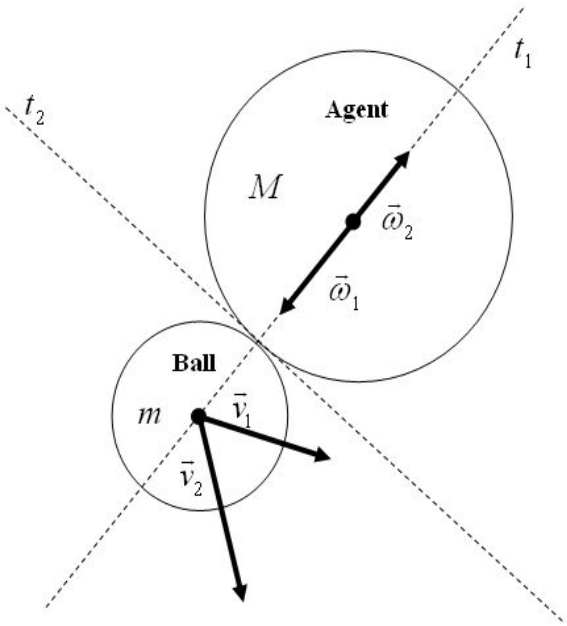

Fig. 3. Agent collision with ball. $\vec{v}_1$ and $\vec{\omega}_1$ denotes ball and agent velocities before collision, whereas $\vec{v}_2$ and $\vec{\omega}_2$ after collision, respectively. t_2 is axle of collision and t_1 is perpendicular to it.

D. Kick Position

In order to avoid unnecessarily complex computations, we define the agent position touching the ball at the end of a simulation time quantum, and the control sequence for the next *Ts* results the required agent velocity magnitude and orientation, so after collision the ball moves as desired (Fig. 4).

Suppose that the ball position *C* is known at the very moment of collision. Let *D* denote the desired target position, whereas *r* and *R* are radius of the ball and agent, respectively. Let denote *DVx* and *DVy* the orthogonal components of ball trajectory.

In this manner, the agent position *K* before collision can be formulated with the velocity components:

$$K_x = C_x - \frac{\omega_{x,1}}{\left\|\vec{\omega}_1\right\|}\left(r + R\right), \tag{11}$$

$$K_y = C_y - \frac{\omega_{y,1}}{\left\|\vec{\omega}_1\right\|}\left(r + R\right). \tag{12}$$

E. Approach

When approaching the ball avoiding collision before desired is essential. In order to fulfill this objective, we apply a slightly suboptimal but collision-safe cornerpoint-based approach sequence, which can be easily defined in the coordinate system of the moving ball.

The main characteristics of agent-ball constellation are the following ones: kick from left or right; kick from front or behind; kick from same or other side; agent in dangerous or in safe zone.

Since we have four binary attributes, sixteen sectors can be defined. Concerning kick position, different cases can be seen in Figure 5. To simplify computation and visualization, we can increase the radius of ball and decrease the radius of agent with *R*, so the agent is point-sized.

F. Direct Kick

Since approach sequence is suboptimal, it is frequently redesigned. In order to avoid useless chasing the ball, defining direct kick is greatly beneficial. It means that if collision-safe approach can be realized without cornerpoints, we should choose this solution.

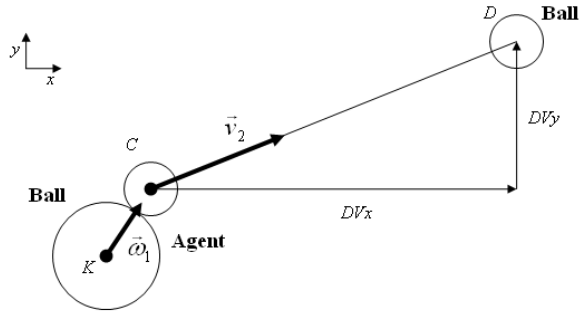

Fig. 4. Ball trajectory after collision.

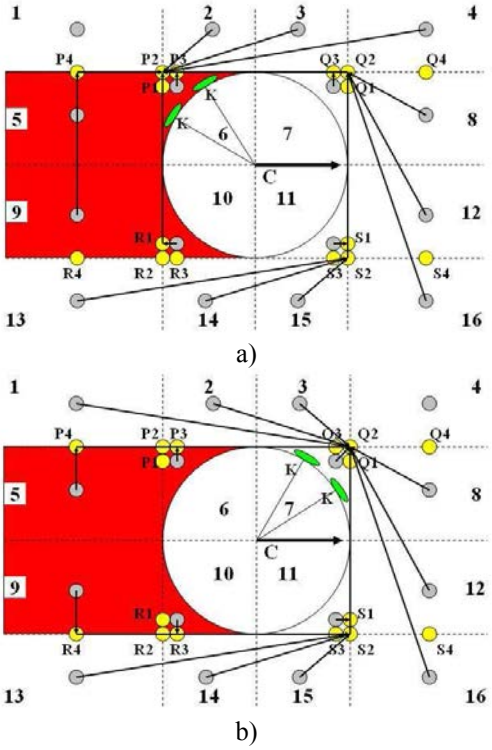

Fig. 5. Approach sequence. a) kick from left-behind, b) kick from left-front, the two other situations are symmetric to these. Grey circles are agent positions, yellow circles are cornerpoints, green ellipses are kick positions and the red area is "dangerous".

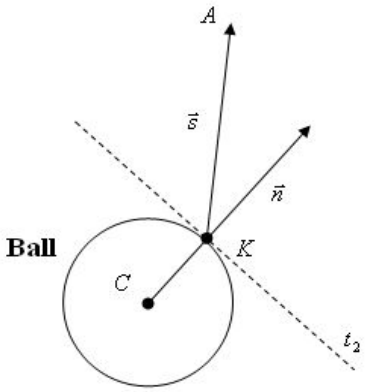

Fig. 6. Direct kick without approach sequence. *A* is agent position and *s* is the vector from *K* to *A*, while *n* is the normal of axle of collision.

Deciding whether the agent can execute direct kick or not, the sign of the scalar product of vectors *s* and *n* has to be computed (Fig. 6). If positive, agent is on the right side, so direct kick can be performed, else cornerpoint-based approach is necessary. Since this is optimal trajectory, the previous problem no longer exists.

G. Coordinate Transformation

Cornerpoints can be easily defined in the coordinate system of the moving ball, but team strategy is described in the team frame, so coordinate transformation is necessary between the two systems (Fig. 7).

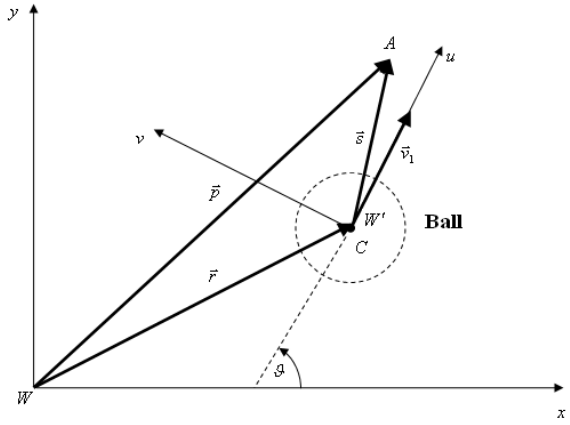

Fig. 7. u and v are axes of the ball coordinate system with W' origin, *x* and *y* are the axes of the coordinate system of the team with W origin. $\vec{r}$ is the vector between the two origins, $\vec{s}$ is the vector between W' and the agent, while $\vec{p}$ is connecting W with the agent.

Coordinate transformation *T* can be seen as

$$\vec{s}^{(W)} = \underline{\underline{T}} \cdot \vec{s}^{(W')}, \tag{13}$$

where

$$\underline{\underline{T}} = \begin{bmatrix} \cos\vartheta & -\sin\vartheta \\ \sin\vartheta & \cos\vartheta \end{bmatrix}. \tag{14}$$

According to this, the transformation from the coordinate system of the ball to the coordinate system of the agent can be seen as a translation and a rotation:

$$\vec{s}^{(W')} = \underline{\underline{T}}^{-1} \cdot \vec{s}^{(W)} = \underline{\underline{T}}^{-1} \cdot \left(\vec{p}^{(W)} - \vec{r}^{(W)}\right). \tag{15}$$

H. Algorithm

After solving the previous subproblems, integration is the next step. By now, we supposed that the kick time is given, but in real situations it is always unknown. Because of the presence of opponents, the most important aspects are accuracy and speed.

Numerical solutions could not be applied since they have to be computed with slow iterations in every cycle which is unacceptable. Consequently, precise and fast analytical solutions are required. Since the model is quite complex (for example ball damping), general approach is not possible, so rational simplifications have to be made. Concerning the requirements, defining possible kick times with varying resolution is a satisfying decision. The earlier the time, the smaller the step size, so the chosen function is exponential-type.

Since agent velocity is limited, not all analytical solutions can be executed. When examining a possible kick, the orientation of the ball after the kick is given, which is defined by the ratio DVy/DVx. Consequently, ball velocity after kick can be set with the scale factor *a*, which defines strength

$$\vec{v}_2 = (a \cdot DVx, a \cdot DVy). \quad (17)$$

For the same reasons as with possible kick times, the resolution of strength is also changing in the same way, so the function is exponential as well.

By decreasing the maximum strength, we force our agents to execute smaller kicks more often to direct the ball to the desired target. In this manner, we can create a very useful and complex primitive, which is dribble.

Summarizing the achieved results, the kick defining algorithm can be seen in Fig. 8.

I. Dominated Regions

After implementing the most important ball manipulation primitives, some strategic aspects have to be taken. When playing a real game, defining dominated areas can be a great advantage, since in this manner blocking the opponent to manipulate the ball can be fulfilled.

A standard method is applying Voronoi regions, which means computing the distances for every point and the nearest agent dominates the examined point. Even with pitch discretization this is a time consuming method, which has to be avoided.

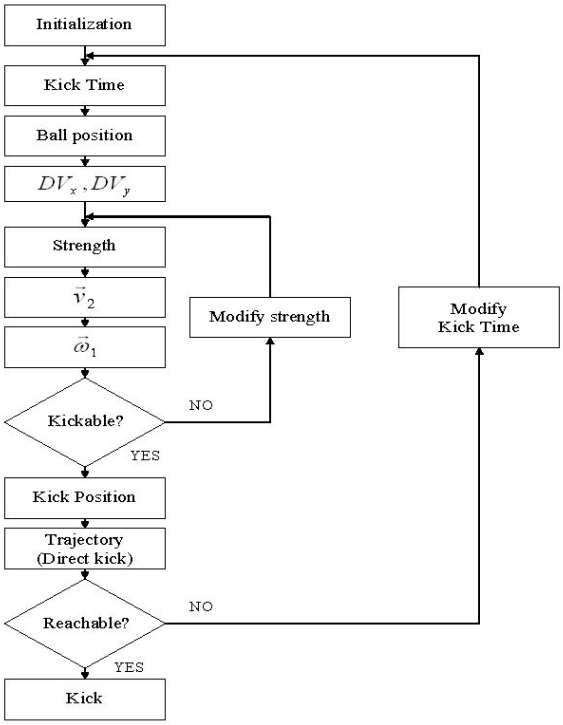

Fig. 8. Kick defining algorithm.

A simple but efficient solution is to compute a distance matrix at initialization, and load the corresponding submatrix to the examined agent position. In this manner, distance computation is required only once.

This time saving method enables us to use more sophisticated techniques than simple distance, because the pitch is not isotropic for the robots: points in front of them can be reached earlier than that are aside. With the proposed solution, several feature matrices can be defined at the initialization phase, for example Time to Reach matrices, which provides information about when the agent can reach the cells. We can apply one degree resolution in agent orientation, so the approximation is very accurate, and during simulation the correct matrix can be applied rapidly. Fig. 9 shows two matrices with the contour of Time to Reach. In this way, complexity does not affect capacity, so more precise region definition can be reached with less time consumption.

III. Simulation Results

We test the proposed method on a quite complex task: players have to manipulate the ball on a triangle trajectory. In Fig. 10 agents have to pass the ball, so their trajectories are limited, while Fig. 11 shows the situation when they have to dribble: in this case, their movements are much more expanded. It can be seen that passing the ball can be achieved in a precise way. Also dribbling is similar to the passing mechanism, however, in this case the ball's velocity is relatively small in comparison with the agent's velocity.

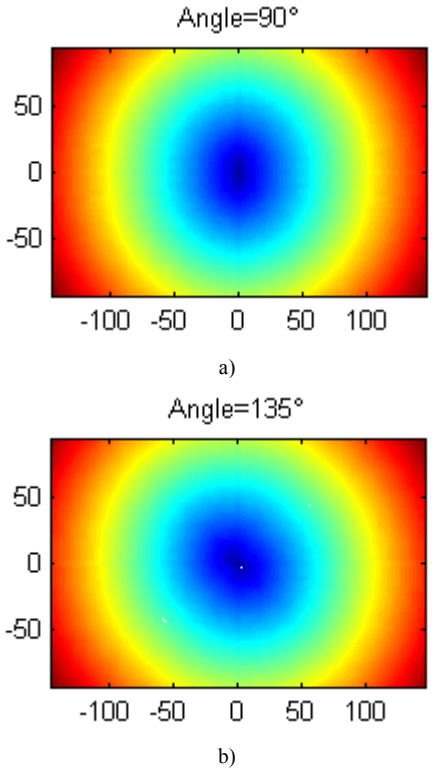

Fig. 9. Time to Reach matrices. a) angle=90° and b) angle=135°.

To demonstrate the use of extended Voronoi regions, Fig. 12 shows the dominated areas at a certain situation. Team strategy based on the presented algorithms are proved to be much more successful than [17] which had been previously used at the laboratory.

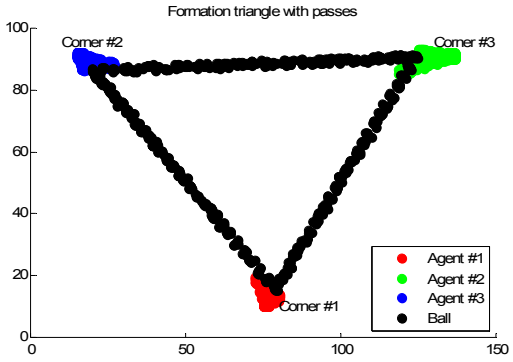

Fig. 10. Formation triangle with passes. Red, green and blue areas are the trajectories of passing players, while black is the trajectory of the ball.

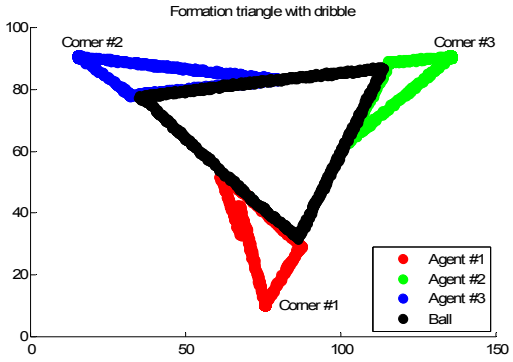

Fig. 11. Formation triangle with dribble. Red, green and blue areas are the trajectories of dribbling players, while black is the trajectory of the ball.

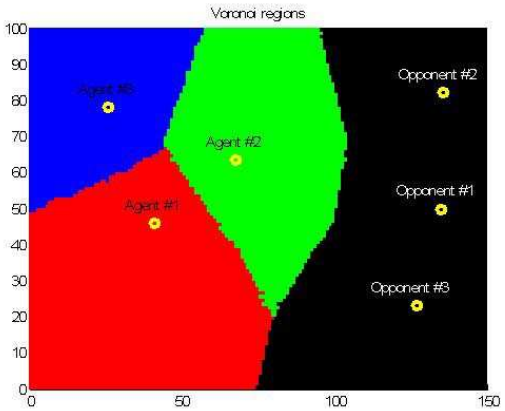

Fig. 12. Voronoi regions. Red, green and blue are safe areas of own players, while black territory is dominated by the opponent team.

IV. Conclusion

In the paper, we presented the architecture and the main characteristics of our robot soccer simulator. We proposed precise and fast methods successfully applied in robot soccer, consequently kick, pass and dribble can be executed in a rapid and accurate manner. In order to implement successful team strategies, a complexity independent dominated region definition was explained. We avoided slow numerical solutions and inexact soft computing based approaches, all of our results are analytical although suboptimal.

References

[1] A.M. Andrew, "RoboCup-97: Robot Soccer World Cup I", Emerald Group Publishing Limited, Volume 28, Issue 5, 1999

[2] G. Lakemeyer, E. Sklar, D.G. Sorrenti, T. Takahashi, "RoboCup 2006: Robot Soccer World Cup X", Springer, Berlin, 2007

[3] U. Visser, F. Ribeiro, T. Ohashi, F. Dellaert, "RoboCup 2007: Robot Soccer World Cup XI", Springer, Berlin, 2007

[4] J-H. Kim, Y-J. Kim, D-H. Kim, K-T. Seow, "Soccer Robotics", Springer, Berlin, Volume 11, 2004

[5] R. de Boer, J.R. Kok, "The incremental development of a synthetic multi-agent system: the UvA Trilearn 2001 robotic soccer simulation team", Master's Thesis, University of Amsterdam, The Netherlands, 2002

[6] V. Korotkich, N. Patson, "On Understanding Global Behavior of Teams in Robot Soccer Simulators", Springer, Berlin, 1999, pp. 429-439.

[7] C-C. Wong, M-F. Chou, C-P. H., C-H. Tsai, S-R. Shyu, " A method for obstacle avoidance and shooting action of the robot soccer", In Proceedings of IEEE International Conference on Robotics and Automation, Volume 4, 2001, pp. 3778 – 3782.

[8] J-S. Liu, T-C. Liang, Y-A. Lin, "Realization of a ball passing strategy for a robot soccer game: a case study of integrated planning and control", Cambridge Journals, Robotica, Cambridge University Press, Volume 22, Issue 3, 2004, pp. 329-338.

[9] A. Tews, G. Wyeth, "Multi-Robot Coordination in the Robot Soccer Environment", Proceedings of the Australian Conference on Robotics and Automation, 1999, pp. 90-95.

[10] G. Klancar, M. Lepetic, B. Zupancic, "Robot soccer collision modelling and validation in multi-agent simulator", Mathematical and Computer Modelling of Dynamical Systems, Volume 9, Number 2, 2003, pp. 137-150.

[11] W-P. Lee, J. Hallam, H.H. Lund, "Applying genetic programming to evolve behavior primitives and arbitrators for mobile robots", In Prodeedings of IEEE 4th International Conference On Evolutionary Computation, 1997

[12] P. Vadakkepat, X. Peng, B.K. Quek, T.H. Lee, "Evolution of fuzzy behaviors for multi-robotic system", Robotics and Autonomous Systems, Volume 55, Issue 2, 2007, pp. 146-161.

[13] S.M. LaValle, "A Game-Theoretic Framework For Robot Motion Planning - LaValle ", PhD Thesis, University of Illinois, 1995

[14] T-C. Liang, J-S. Liu, "Collision-free path planning for mobile robot using cubic spiral", IEEE Conferences on Robotics and Biomimetics, 2004, pp. 671-676.

[15] K. Browne, J. McCune, A. Trost, D. Evans, D. Brogan, "Behavior Combination and Swarm Programming", RoboCup International Symposium, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2001

[16] P-Y. Zhang, T-S. Lu, L-B. Song," Soccer robot path planning on the artificial potential field approach with simulated annealing", Cambridge Journals, Robotica, Cambridge University Press, Volume 22, Issue 5, 2004, pp. 563-566.

[17] P. Gasztonyi, „Heuristiscs-based High-level Strategy for Robot Soccer", 18th International Conference on Artificial Neural Networks, September 2008.