Available online at www.sciencedirect.com

ScienceDirect

ELSEVIER

Simulation Modelling Practice and Theory 15 (2007) 647–658

SIMULATION MODELLING PRACTICE AND THEORY

www.elsevier.com/locate/simpat

Modelling and simulation of a group of mobile robots

Gregor Klančar *, Borut Zupančič, Rihard Karba

Faculty of Electrical Engineering, University of Ljubljana, Tržaška 25, 1000 Ljubljana, Slovenia

Received 7 February 2005; received in revised form 7 February 2007; accepted 8 February 2007
Available online 16 February 2007

Abstract

In this paper the mathematical background of the developed robot soccer simulator is presented. It involves robot and ball dynamic behaviour and focuses mainly on their collisions study. Vital parts of the simulator are explained and modelled in more detail, beginning with the simple model of ball and robot motion and continuing with a more complex approximate collisions models, where the real robot shape is taken into consideration. Some new ideas of collision formulation, realization and real robot shape inclusion are used. The implementation of the simulator is described and advantages for the usage of the realistic simulator are stated.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Simulator; Multi-agent system; Collision detection; Modelling; Discontinuous simulation

1. Introduction

In this paper the mathematical background of the developed robot soccer simulator is presented. Robot soccer has been a very popular research game and has served as a perfect example of multi-agent systems in the last few years [3,11,12,16]. The main purpose of the simulator design procedure is to obtain a realistic simulator which would be used as a tool in the process of strategy and control algorithms design for real world robot soccer as well as for other mobile-robotics related topics. To assure transferability to the real system the obtained strategy algorithms have to be designed on a realistic simulator.

The main motivation for robot soccer simulator development was to design and study multi-agent control and strategy algorithms in FIRA Middle or Large League MiroSot category (5 against 5 or 11 against 11 robots). However, on FIRA's (Federation of International Robot Soccer Association) official website (www.fira.net) there exists a simulator for SimuroSot league, which could only be used in Middle League MiroSot (5 against 5 robots). A similar simulator was built by Liang and Liu [9] where robot motion is simulated by dynamic model, collisions remaining oversimplified. There also exist a number of other simulator applications but not many papers are available. An important part of every realistic robot soccer simulator is collision modelling and simulation. Good mathematical background in rigid body collisions modelling and simulation

* Corresponding author. Tel.: +386 1 4768701.
E-mail address: gregor.klancar@fe.uni-lj.si (G. Klančar).

1569-190X/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.simpat.2007.02.002

could be found in [1,2]. Another useful contribution in the field of robotic simulator is [8] where collisions are treated by spring-dumper approach rather than by impulse force only. The use of spring-dumper linkage in collisions makes velocities changes continuous, which is less problematic for simulation than discontinuous change of velocities [4] obtained by impulse usage. However, spring and dumper coefficients are not easy to identify. Moreover, when observed from macroscopic time scale (as it is in simulation) collisions are indeed discontinuous events.

Simulated robots should have a realistic shape, which should not be represented simply with a square (the real shape of the robot is not a square) otherwise the simulation of ball guidance and other collisions becomes unrealistic. Furthermore, some of the available robot soccer simulators do not treat collisions well, especially the collisions among robots (robot corners), collisions between robot and boundary and situations where the ball is in-between two robots or robot and boundary. Algorithms on such simulators are also not transferable enough to the real system. A majority of them is used for competitions in simulation league and these simulators do not need to be realistic.

With a rapid progress of computer graphics used in computer games, animated movies and other purposes a number of physics engines have appeared which can realistically simulate rigid body dynamics considering variables such as mass, inertia, velocity, friction, etc. Some of available physics engines are ODE – Open Dynamics Engine, Ageia physX, AERO, Karma in Unreal Engine and many others. Their usage enables computer simulations, animations and games such as racing games to appear more realistic. Depending on their usage there exist two types of physics engines, namely real-time and high precision. When dealing with interactive computing (e.g. video games), the physics engines are simplified in order to perform in real-time. On the other hand high precision physics engines require more processing power to be able to calculate very precise physics and are usually used by scientists and computer animated movies. Some of physics engines are free and open source. As such they can also be used to simulate physics in different research oriented experiments. These packages are usually comprehensive and therefore quite difficult to manage, use and modify. When constructing the mobile robot its mathematical background was completely developed by our team, which enabled us to get a better insight into the problem domain and gave us the possibility to efficiently solve some simulator specifics as mentioned in the sequel.

The presented simulator is mainly used as a tool in control and strategy design of multi-agent system in real game and therefore needs to be realistic. Strategy design could be developed also on a real plant but there are some important reasons which benefit the usage of realistic simulator as stated in the paper. Some vital parts of the simulator are explained and modelled in more detail, beginning with the kinematics and dynamic motion modelling considering kinematics constraints and, further on dealing with different collisions modelling. The stress is given to the motion modelling where the assumptions of pure rolling conditions are made and dynamic properties are included. The results of this part are motion models of the ball and the robot with differential drive. Some new ideas of collision formulation and realization (taking into account the real robot shape) are used as well. Collisions are simply solved by mathematically correct discontinuous change of velocities (states of the velocity integrators), which is more convenient for realization than simulating collisions by applying impulse force [1,8]. However, collisions are only described by approximate models, which are sufficient enough for realistic behaviour of the obtained simulator. Precise collisions modelling is usually very demanding because of many factors, which should be considered during collision. When simulating a realistic game a precise collision modelling is less important than motion modelling. This is because the game strategy is designed to play a good game where different collisions are undesired and we want to avoid them. Nevertheless collisions still happen and have to be handled. The problems of collision detection and the method of finding the exact time of the collision are exposed too. For the latter the existing algorithms in Matlab Simulink are used.

The system presented in this paper is available for other researchers. It can be used for mobile-robot related experiments, such as multi-agent strategy design, agent behaviour analysis, robot motion planning, cooperation, collision avoidance, motion planning, control and the like. The presented simulation is available at our website [6].

The paper is organized as follows. First, a brief system overview is revealed, followed by the mathematical model derivation of basic agents (robots and ball). Then some new ideas of collisions modelling considering complex robot shape are presented in more detail. The conclusions are given in the final part of the paper.

2. System overview

The robot soccer set-up (see Fig. 1) consists of 10 Middle League MiroSot category robots (generating two teams) of size 7.5 cm cubed, orange golf ball, rectangular playground of size 2.2×1.8 m, colour camera and personal computer. Colour camera is mounted above playground (each team has its own) and is used as a global motion sensor. The objects are identified from their colour information; orange ball and colour dresses of robots. The agent-based control part of the programme calculates commands for each agent (robot) and sends them to the robot by a radio connection. The robots are then driven by two powerful DC motors; one for each wheel.

The role of the simulator developed in the paper is to replace the real playground, camera, robots and ball, which is expensive and needs a large place to be set up. Therefore the simulator must include mathematical models of motion as well as collisions which happen on the playground.

3. Mathematical modelling

To simulate robot soccer game mathematic motion equations should be derived first. The playground activities consist of two kinds of moving objects: robot and ball. Therefore their motion modelling [10] is presented in the sequel.

3.1. Robot model

The robot has a two-wheel differential drive located at the geometric centre, which allows zero turn radius and omni-directional steering because of nonholonomic constraint [7]. It is an active object in the robot soccer game. Its appearance is given in Fig. 2 and its motion is described in the sequel by kinematics and dynamic motion equations.

Where $T_o = (x_o, y_o)$ is robot geometric centre, $T_c = (x_c, y_c)$ is its mass centre, m_c is body mass, m_k is wheel mass and J_c, J_k, J_m are moments of inertia for robot body around axis Z, for wheel around its axle and wheel around axis Z, respectively. Supposing pure rolling conditions of the wheels, the following kinematics constraints can be written:

$$\begin{aligned} &\dot{y}_c \cos\theta - \dot{x}_c \sin\theta - \dot{\theta} d = 0 \\ &\dot{x}_c \cos\theta + \dot{y}_c \sin\theta + b\dot{\theta} = r\dot{\phi}_r \\ &\dot{x}_c \cos\theta + \dot{y}_c \sin\theta - b\dot{\theta} = r\dot{\phi}_l \end{aligned} \tag{1}$$

where θ is the robot orientation, ϕ_r and ϕ_l are the angles describing wheels rotation and d is the distance between mass centre and geometric centre. According to the first constraint in Eq. (1), the robot cannot slide in the sideways, while the second and the third constraints describe pure rolling of the wheels. The null space of kinematics constraints (1) defines robot kinematics motion equation, given as:

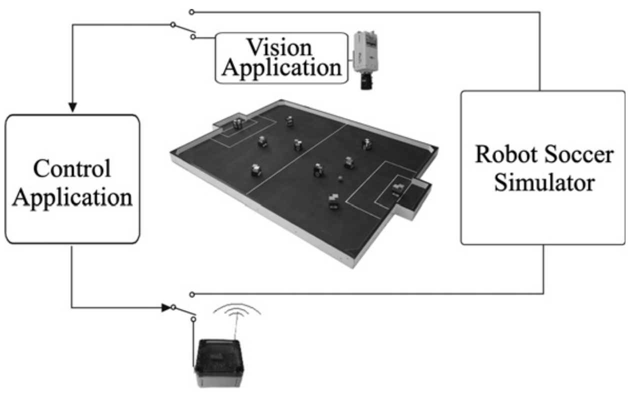

Fig. 1. Robot soccer system overview.

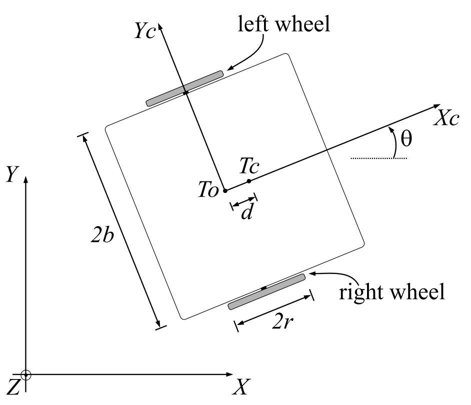

Fig. 2. Symbol description.

$$\begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\theta} \\ \dot{\phi}_r \\ \dot{\phi}_l \end{bmatrix} = \begin{bmatrix} \frac{r}{2b}(b\cos(\theta) - d\sin(\theta)) & \frac{r}{2b}(b\cos(\theta) + d\sin(\theta)) \\ \frac{r}{2b}(b\sin(\theta) + d\cos(\theta)) & \frac{r}{2b}(b\sin(\theta) - d\cos(\theta)) \\ \frac{r}{2b} & -\frac{r}{2b} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \dot{\phi}_r \\ \dot{\phi}_l \end{bmatrix} \tag{2}$$

Dynamics motion equation can further be derived using Lagrange formulation [17]

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}_k}\right) - \frac{\partial L}{\partial q_k} + \frac{\partial P}{\partial \dot{q}_k} = f_k - \sum_{j=1}^{m} \lambda_j a_{jk} \tag{3}$$

the last part of Eq. (3), λ_j are Lagrange multiplicators associated with jth ($j = 1,\ldots,3$) constraint equation and a_{jk} is kth ($k = 1,\ldots,5$) coefficient of kth constraint equation. Lagrangian is defined as:

$$L = \frac{m_c}{2}(\dot{x}_c^2 + \dot{y}_c^2) + \frac{m_k}{2}(\dot{x}_{k_r}^2 + \dot{y}_{k_r}^2) + \frac{m_k}{2}(\dot{x}_{k_l}^2 + \dot{y}_{k_l}^2) + +\frac{J_c}{2}\dot{\theta}^2 + 2\frac{J_m}{2}\dot{\theta}^2 + \frac{J_k}{2}\dot{\phi}_r^2 + \frac{J_k}{2}\dot{\phi}_l^2 \tag{4}$$

Defining $m = m_c + 2m_k$, $J = J_c + 2J_m + 2m_k(d^2 + b^2)$ and expressing (4) by robot mass centre variables the following is obtained:

$$L = \frac{m}{2}(\dot{x}_c^2 + \dot{y}_c^2) + \frac{J}{2}\dot{\theta}^2 + \frac{J_k}{2}\dot{\phi}_r^2 + \frac{J_k}{2}\dot{\phi}_l^2 + 2m_k\,\mathrm{d}\dot{\theta}(\dot{x}_c\sin\theta - \dot{y}_c\cos\theta) \tag{5}$$

According to (3) the dynamic model is written as:

$$\begin{aligned} &m\ddot{x}_c + 2m_k d(\ddot{\theta}\sin\theta + \dot{\theta}^2\cos\theta) - \lambda_1\sin\theta + (\lambda_2 + \lambda_3)\cos\theta = 0 \\ &m\ddot{y}_c - 2m_k d(\ddot{\theta}\cos\theta - \dot{\theta}^2\sin\theta) + \lambda_1\cos\theta + (\lambda_2 + \lambda_3)\sin\theta = 0 \\ &J\ddot{\theta} + 2m_k d(\ddot{x}_c\sin\theta - \ddot{y}_c\cos\theta) - \lambda_1 d + (\lambda_2 - \lambda_3)b = 0 \\ &J_k\ddot{\phi}_r + \mu\dot{\phi}_r - \lambda_2 r = \tau_r \\ &J_k\ddot{\phi}_l + \mu\dot{\phi}_l - \lambda_3 r = \tau_l \end{aligned} \tag{6}$$

where λ_1, λ_2, λ_3 are Lagrange multiplicators which can effectively be eliminated by the procedure given in [14,15]. Brief summary is given in the sequel. Lagrangian formulation (3) can be expressed in matrix form, such as:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{V}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{F}(\dot{\mathbf{q}}) = \mathbf{E}(\mathbf{q})\mathbf{u} - \mathbf{A}^{\mathrm{T}}(\mathbf{q})\boldsymbol{\lambda} \tag{7}$$

where $\mathbf{M}(\mathbf{q})$ is inertia matrix, $\mathbf{V}(\mathbf{q},\dot{\mathbf{q}})$ is vector of position and velocity dependent forces, $\mathbf{F}(\dot{\mathbf{q}})$ is vector of friction or dumping forces, $\mathbf{E}(\mathbf{q})$ is input transformation matrix, $\mathbf{u}$ is input vector of actuator forces and torques and $\mathbf{A}(\mathbf{q})$ is the matrix of kinematics constraints. System kinematics from Eq. (2) expressed in matrix form reads:

$$\dot{\mathbf{q}} = \mathbf{S}(\mathbf{q})\mathbf{v}(t) \tag{8}$$

and matrix form of kinematics constraints from Eq. (1) is

$$\mathbf{A}(\mathbf{q})\dot{\mathbf{q}} = 0 \tag{9}$$

Calculating first derivative of (8) gives

$$\ddot{\mathbf{q}} = \dot{\mathbf{S}}\mathbf{v} + \mathbf{S}\dot{\mathbf{v}} \tag{10}$$

Lagrange multiplicators can finally be eliminated by substituting (8) and (10) in Eq. (7) and pre-multiplying by $\mathbf{S}^{\mathrm{T}}$. The part with Lagrangian multiplicators vanish because $\mathbf{S}^{\mathrm{T}}\ \mathbf{A}^{\mathrm{T}} = \mathbf{0}$.

The dynamics of electric part (the motors) can usually be neglected, as electrical time constants are usually significantly smaller than mechanical time constants.

3.2. Ball model

The ball is a passive object whose motion across the playground can be described by five generalized coordinates as shown in Fig. 3.

Dynamics motion equation can be derived using Lagrange formulation

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[\frac{\partial L}{\partial \dot{q}_k}\right] - \frac{\partial L}{\partial q_k} + \frac{\partial P}{\partial \dot{q}_k} = f(t) \tag{11}$$

where L stands for difference between kinetic and potential energy, P stands for power function (dissipation function), q_k stands for generalized coordinate and $f(t)$ is external force respectively and is nonzero when the ball collides. Lagrangian is defined as

$$L = \frac{1}{2}m(\dot{x}^2 + \dot{y}^2) + \frac{1}{2}J(\dot{\varphi}_x^2 + \dot{\varphi}_y^2 + \dot{\varphi}_z^2) \tag{12}$$

where m is the ball mass and J is moment of inertia. Supposing pure rolling conditions the following kinematics constraints follow:

$$\begin{aligned} \dot{x} + r\dot{\varphi}_y &= 0 \\ \dot{y} - r\dot{\varphi}_x &= 0 \end{aligned} \tag{13}$$

where r is ball radius. Both conditions in Eq. (13) give perfect rolling of the ball, i.e. motion with no slipping. Constraints in Eq. (13) are holonomic (integrable) and can be used to eliminate two generalized coordinates. Further on, by neglecting rotation around z axis $\omega_z = 0$ and using constraints (13), Eq. (12) is rewritten as

$$L = \frac{m + \frac{J}{r^2}}{2}(\dot{x}^2 + \dot{y}^2) \tag{14}$$

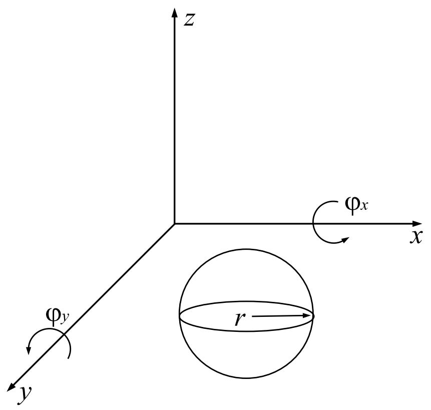

Fig. 3. The ball rolling on the plane.

The power function is

$$P = \frac{1}{2} f_D \dot{x}^2 + \frac{1}{2} f_D \dot{y}^2 \tag{15}$$

where f_D is dumping coefficient. Considering (11) the final motion equation of the ball are as follows:

$$\begin{aligned} \ddot{x} &= \frac{F(t) - \dot{x} \cdot f_D}{m + J/r^2} \\ \ddot{y} &= \frac{F(t) - \dot{y} \cdot f_D}{m + J/r^2} \end{aligned} \tag{16}$$

4. Collisions modelling

During the motion of the robots and the ball on the playground several collisions between them are possible. They are given as submodels and describe the collision between moving objects: the robot–ball collision model, the robot–boundary collision model, the ball–boundary collision model and the collision between robots model. When simulating a realistic game, a precise collision modelling is less important than motion modelling. This is because the game strategy is designed to play a good game where different collisions are undesired and we want to avoid them. Nevertheless collisions still happen and have to be handled. However, in the sequel the collision models only approximately describe real situations. Most of the presented models are therefore relatively simple for realization in a simulator.

4.1. Robot–boundary collision

When modelling collision of the robot to the boundary, the test whether all robot corners are inside the playground must be performed first. If they are, this means that there is no such collision. The procedure is represented by diagram in Fig. 4.

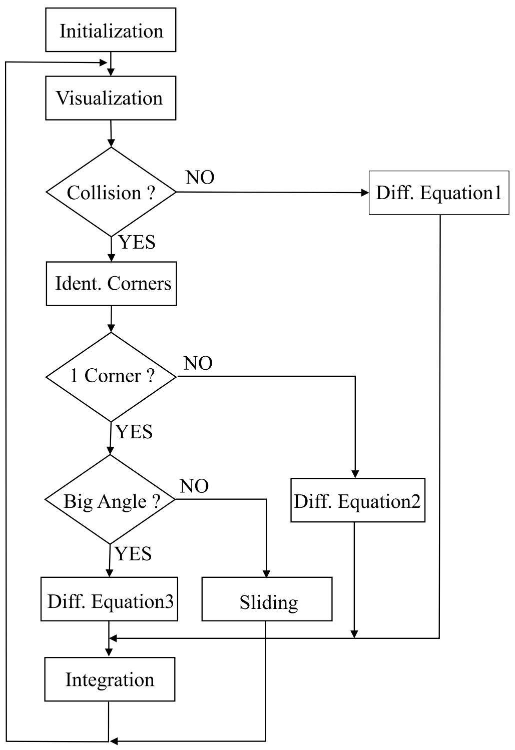

Fig. 4. Robot–boundary collision simulation diagram.

The notation differential equation (1) in Fig. 4 stands for Eq. (3). When the robot hits the boundary with two corners, it stops and so robot kinematics equation (in Fig. 4 marked as differential equation (2)) becomes:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{17}$$

More demanding case appears when the robot hits the boundary with one corner only. If the angle between the robot and the boundary is greater than the proposed threshold value, the robot starts to rotate around the corner (see Fig. 5).

The velocity in point T_K with tangential direction to the outer circle in Fig. 5 is obtained by a transformation of the left wheel rim velocity ($\omega_L \cdot r$). Angular velocity ω_{TK} in point T_K is thus

$$\omega_{T_K} = \frac{\omega_L \cdot r \cdot \cos(\alpha)}{\sqrt{L^2 + L^2/4}} \tag{18}$$

where angle α is

$$\alpha = \operatorname{arctg}\left(\frac{L/2}{L}\right) \tag{19}$$

and linear velocity of the robot centre (v_{Ts}) is:

$$v_{T_S} = \omega_{T_K}\sqrt{\frac{L^2}{2}} = \omega_L r \sqrt{\frac{2}{5}}\cos(\alpha) \tag{20}$$

Robot kinematics equation (in Fig. 4 marked as differential equation (3)) then becomes:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \sqrt{\frac{2}{5}} \cdot r \cdot \cos(\alpha) \cdot \cos(\varphi + \frac{\pi}{4}) & 0 \\ \sqrt{\frac{2}{5}} \cdot r \cdot \cos(\alpha) \cdot \sin(\varphi + \frac{\pi}{4}) & 0 \\ -\sqrt{\frac{4}{5}} \cdot \frac{r}{L} \cdot \cos(\alpha) & 0 \end{bmatrix} \cdot \begin{bmatrix} \omega_L \\ \omega_R \end{bmatrix} \tag{21}$$

If the angle between the robot and the boundary is less than the mentioned threshold, the robot slides along the boundary (see Fig. 4).

4.2. Ball–boundary collision

In the ball–boundary collision elastic collision is supposed. The velocity component parallel to the boundary remains the same, while the perpendicular velocity component changes sign and is multiplied by a factor

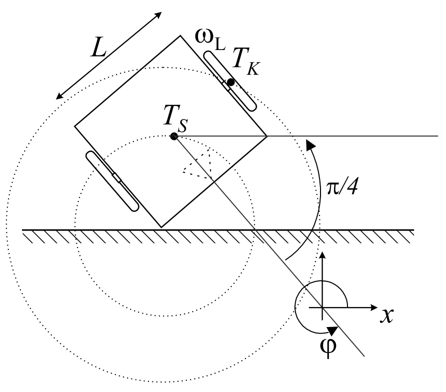

Fig. 5. One-corner collision with the boundary.

less than one, representing energy loss. To assure proper rebound without penetration, zero crossing algorithm implemented in Matlab Simulink environment is used to treat the problem of integration over discontinuities correctly and efficiently. This algorithm simply changes the integration step by bisection, according to some input variable (distance between ball and boundary multiplied by sign which is negative if the ball is outside the playground), until the exact time of discontinuity appears.

4.3. Robot–ball collision

Mutual impact of the robot and the ball can be described with collision model of two spheres (Fig. 6). Mathematically the model is based on kinetic energy and momentum balance equations as follows:

$$\begin{aligned} &m_1 v_{x_1}^2 + m_2 v_{x_2}^2 + m_1 v_{y_1}^2 + m_2 v_{y_2}^2 = m_1 w_{x_1}^2 + m_2 w_{x_2}^2 + m_1 w_{y_1}^2 + m_2 w_{y_2}^2 \\ &m_1 v_{x_1} + m_2 v_{x_2} = m_1 w_{x_1} + m_2 w_{x_2} \\ &m_1 v_{y_1} + m_2 v_{y_2} = m_1 w_{y_1} + m_2 w_{y_2} \end{aligned} \tag{22}$$

where indexes 1 and 2 stand for the first and second sphere, v represents the velocities before and w the velocities after the collision, while m_1 is robot and m_2 ball mass respectively.

The playground coordinate system is rotated so that axis x connects mass centres of the spheres (see Fig. 6).

Because of the coordinate system rotation the impact force is different from zero only in normal direction of the collision, i.e. direction x. Thus the velocities in direction y remain the same. Final non-trivial velocities after the collision are then given by:

$$\begin{aligned} w_{x_1} &= \frac{-m_2 v_{x_1} + m_1 v_{x_1} + 2m_2 v_{x_2}}{m_1 + m_2} \\ w_{x_2} &= \frac{2m_1 v_{x_1} + m_2 v_{x_2} - m_1 v_{x_2}}{m_1 + m_2} \\ w_{y_1} &= v_{y_1} \\ w_{y_2} &= v_{y_2} \end{aligned} \tag{23}$$

where index 1 stands for the robot and index 2 stands for the ball. If m_2 is very small in comparison with m_1, a simplification of Eq. (23) is justified. Some manipulations give:

$$\begin{aligned} w_{x_1} &= v_{x_1} \\ w_{x_2} &= v_{x_1} + k(v_{x_1} - v_{x_2}) \\ w_{y_1} &= v_{y_1} \\ w_{y_2} &= v_{y_2} \end{aligned} \tag{24}$$

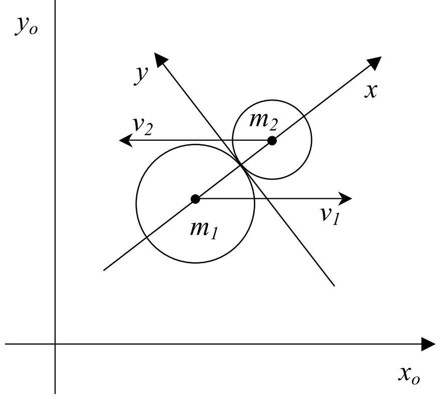

Fig. 6. Collision of two spheres.

Furthermore, energy loss is realized by multiplying the part of Eq. (24) inside the brackets by factor k less than one.

Calculated velocities after the collision are then used as new initial states of the integrators in the simulator. This is equivalent to applying and simulating impulse force caused by collision but is less suitable for realization [10,13].

However to assure a realistic collision of the robot and the ball, a concrete robot shape has to be modelled. The actual robot shape is shown in collision situation in Fig. 7 and the idea of how to include the real robot shape into the model is given in Fig. 8.

The outer shape is the rim of the robot obtained if the ball is rolled around the robot and its positions are recorded. With the proposed reshaping the collision of the robot with the ball can be treated as a collision between two points (ball centre and point on robot rim). Because linear and angular velocities of the robot are given for geometrical centre, the following transformations have to be done in order to obtain the velocities in the point of the rim where the collision with the ball occurs:

$$\begin{aligned} v_{x_1} &= v - \omega r(\varphi)\sin\varphi \\ v_{y_1} &= \omega r(\varphi)\cos\varphi \end{aligned} \tag{25}$$

Function $r(\varphi)$ is the distance from the robot centre to the collision point on the rim and φ is the angle from the local robot axis x to the line connecting the robot centre and the collision point. To solve Eq. (23) the playground coordinates are rotated first so that axis x is in tangential direction of the rim (in the point of collision). After that the collision results are transformed to the global coordinates.

The shape of the robot is described with two look-up tables (distance $r(\varphi)$ and tangent(φ) of the rim), which are addressed with angle φ. To detect if the ball hits the robot, a check of the distance between their centres must be performed. If the distance is less than the one obtained from look-up table $r(\varphi)$, the ball hits the robot. The accurate time of the collision is again obtained by zero crossing algorithm. So proper collision without penetration (within machine precision) and accurate integration over velocities are assured.

4.4. Collisions between robots

The collision of two or even more robots is undoubtedly problematic from the modelling point of view. However, the complexity of the model must be strongly dependent on the demands of the realistic simulator, where the compromise between reality approximation and simulation precision must be found according to the simulation usage aims. During simulator design a few more or less approximate solutions were tested until finally the best one was implemented. When designing the control strategy of the robot soccer game, it seems that collisions between robots are not so important because one focuses mainly on shots on goal, on passes,

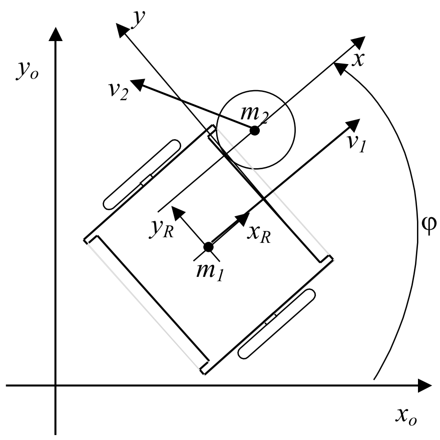

Fig. 7. Robot–ball collision.

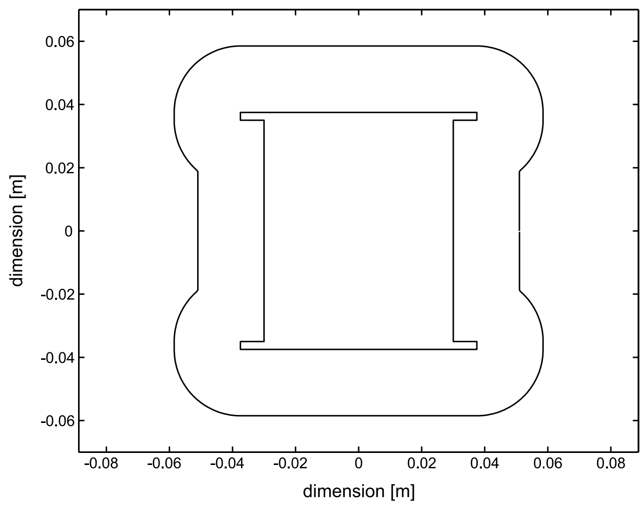

Fig. 8. Shape of the robot (inner) and its rim.

organizing defence and similar actions, while collisions between robots are more or less undesired. However, collisions between robots are quite frequent in the game and in the case of defence also very important. Therefore they must be treated correspondingly in a realistic simulator.

4.4.1. Collision detection

A collision detection algorithm [5] consists of two steps. In the first step only the information about a possible collision is obtained. The second step is then performed only if the possibility obtained from the first step exists. In the second step a separating plane between objects is found. The reason for performing collision detection in two steps is only due to lower computational burden. Thus, the second step is performed only in situations where collision is almost inevitable.

The first step is performed by analyzing bounding boxes of all robots. The latter have their sides parallel to the global coordinate axes, thus representing the rectangle in which robot in its current position is included (see Fig. 9). The possibility of two objects colliding exists only if the bounding boxes overlap. The overlapping between two bounding boxes is determined by checking if their sides overlap in both axis directions (x and y) at the same time.

As mentioned before the second step is performed only if the overlapping of bounded boxes from the first step exists. The separating plane is calculated so that one object (convex polyhedrons) is on one side of the plane and the other on another side of the separating plane. The separating plane always exists if two objects do not invade.

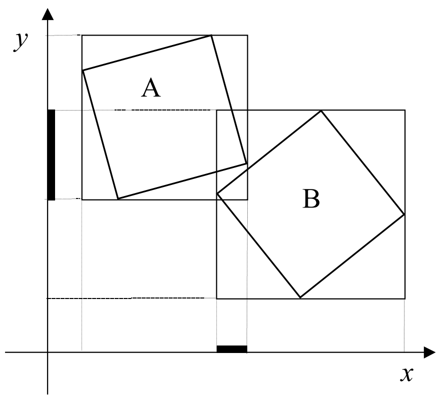

Fig. 9. Overlapping of bounding boxes in both directions.

4.4.2. Collision realization

In a two-dimensional space the separating plane is a straight line. It is convenient that the separating plane has a normal in the same direction as is the normal direction of collision. A separating plane should thus contain the side of one of the two objects which are involved in collision (see Fig. 10).

When a collision of two robots appears, the following holds:

$$\Delta\vec{G} = \int \vec{F}\,\mathrm{d}t \tag{26}$$

where $\vec{G}$ stands for conservation of momentum and $\vec{F}\,\mathrm{d}t$ is force impulse acting at the time of collision. Because of the force impulse a sudden change in velocities of the two robots occurs. Force impulse acts only in normal direction of the collision. Thus only the velocity components in the normal direction of the collision change while perpendicular components remain the same. To calculate the new velocities of the robots after collision the force impulse $\vec{J} = \vec{F}\Delta t$ has to be calculated. The detailed procedure to estimate the velocities of two rigid bodies after collision is described in [1,5]. The idea is to calculate the relative velocities in the collision point $\vec{p}$ (see Fig. 10) before and after the collision in normal direction. It is always true that the absolute value of the relative velocity in normal direction after the collision remains the same compared to the absolute value of the relative velocity in normal direction before collision in point $\vec{p}$. From that property the amplitude of force impulse can be calculated. Having estimated the impulse, linear velocity $\vec{v}^+$ and angular velocity $\vec{\omega}^+$ for robot mass centre can be calculated by using relations:

$$\begin{aligned}\vec{v}^+(t_0) &= \vec{v}^-(t_0) + \frac{\vec{J}(t_0)}{M}\\ \vec{\omega}^+(t_0) &= \vec{\omega}^-(t_0) + I^{-1}(\vec{r}\times\vec{J}(t_0))\end{aligned} \tag{27}$$

where t_0 is time of the collision, M is mass of the robot, I is corresponding moment of inertia and $\vec{r}$ is a displacement vector between mass centre $\vec{x}$ and point of collision $\vec{p}$ (see Fig. 10), while the sign in the superscript denotes time instant of the collision (− before and + after collision). To obtain accurate t_0 again, a zero crossing algorithm could be used in order to assure accurate integration of discontinuous velocities signals. However, the problem of high frequency oscillations around a discontinuity (chattering) appears when two or more robots stay in contact (robots pushing each other). Therefore sampling time of the simulation becomes very small, which results in halting of the simulation. Thus, a better solution is to check for a correspondingly small distance between one robot corner and the separating plane belonging to another robot. If the separating plane does not exist, the time of the penetration must be taken into account. The obtained velocities after the collision are then used to determine new initial states of the integrators in the simulator, which is equivalent to simulating impulse force because of the collision. The former is more suitable and accurate for realization, though.

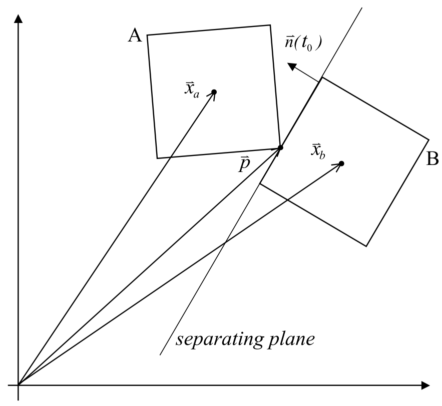

Fig. 10. Collision of two robots.

5. Conclusion

The introduced simulator is mostly used as a tool in the process of strategy and control design for real robot soccer game. Therefore, its verification is done through transferability of the obtained strategy algorithms to the real system. The verification shows that the behaviour of the simulator is similar enough to the real setup, which means that the designed algorithms (strategy and low level control) can directly be used without modifications in real games as well.

The designed simulator has significant improvements in comparison with the available simulator in MiroSot leagues (simulator for SimuroSot) and other available simulators; the advantages being dynamics motion modelling and a realistic shape of the robots, which contributes to a more realistic simulation of robot ball interactions, collisions with robots, robots and boundary interactions and the situations where the ball is captured between two objects (it cannot invade any object). The presented simulator proved to be a good approximation of the real system. The motion models as well as collision models give realistic descriptions, which enable the simulator designed algorithms to be used on the real system.

References

[1] D. Baraf, An introduction to physically based modeling: rigid body simulation II – nonpenetration constraints, in: SIGGRAPH '97 Course Notes, Carnegie Mellon University, 1997.

[2] O. Egeland, J.T. Gravdahl, Modeling and Simulation for Automatic Control, Marine Cybernetics, Trondheim, Norway, 2002.

[3] J. Ferber, Multi-Agent Systems, An Introduction to Distributed Artificial Intelligence, Addison-Wesley, Essex, England, 1999.

[4] M. Fremond, Rigid bodies collisions, Physical Letters A 1 (1995) 34–41.

[5] G. Klančar, M. Lepetič, R. Karba, B. Zupančič, Robot soccer collision modelling and validation in multi-agent simulator, Mathematical and computer modelling of dynamical systems 9 (2) (2003) 137–150.

[6] G. Klančar, Mobile Robot Simulator, <http://msc.fe.uni-lj.si/PublicWWW/Klancar/RobotSimulator.html>.

[7] I. Kolmanovsky, N.H. McClamroch, Developments in nonholonomic control problems, IEEE Control Systems 15 (1995) 20–36.

[8] E. Larsen, A Robot Soccer Simulator: A Case Study for Rigid Body Contact, Sony Computer Entertainment America R&D, March 2001.

[9] T.C. Liang, J.S. Liu, A distributed mobile robot simulator and a ball passing strategy, Technical Report TR-IIS-02-007, Institute of Information Science, Academia Sinica, Nankang, Taiwan, 2002.

[10] D. Matko, R. Karba, B. Zupančič, Simulation and Modelling of Continuous Systems, A Case Study Approach, Prentice-Hall, Englewood Cliffs, USA, 1992.

[11] A. Minoru, R. D'Andrea, A. Birk, H. Kitano, M. Veloso, Robotics in edutainment, in: Proceedings of the 2000 IEEE International Conference on Robotics and Automation (ICRA'2000), San Francisco, USA, 2000, pp. 795–800.

[12] S. Moss, P. Davidsson, Multi-Agent-Based Simulation, Springer-Verlag, New York, 2002.

[13] The Math Works, Inc., Simulink, Dynamic System Simulation for Matlab, Natick, USA, 1998.

[14] G. Oriolo, A. Luca, M. Vandittelli, WMR control via dynamic feedback linearization: design, implementation, and experimental validation, IEEE Transactions on Control Systems Technology 10 (6) (2002) 835–852.

[15] N. Sarkar, X. Yun, V. Kumar, Control of mechanical systems with rolling constraints: application to dynamic control of mobile robot, The International Journal of Robotic Research 13 (1) (1994) 55–69.

[16] P. Stone, M. Veloso, Multiagent systems: a survey from a machine learning perspective, Autonomous Robots 8 (2000) 345–383.

[17] D.A. Welles, Lagrangian Dynamics, Schaum's Outline Series, McGraw Hill Book Company, 1967.