We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

7

Simulated Environment in Robot Soccer

Gregor Klančar and Rihard Karba
University of Ljubljana
Slovenia

1. Introduction

In the last two decades the concept of multi-agent mobile systems has been observed in many computer simulations, laboratory examples and in some practical applications. Among such systems robot soccer has shown to be a very popular research game and has served as a perfect example of multi-agent systems in the last few years (Ferber, 1999; Moss & Davidsson, 2002; Stone & Veloso, 2000).

In this work the mathematical background of the developed robot soccer simulator is presented. 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 & Liu, 2002) 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 could be found in (Baraf, 1997). Another useful contribution in the field of robotic simulator is (Larsen, 2001) 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 (Fremond, 1995) 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

Source: Robotic Soccer, Book edited by: Pedro Lima, ISBN 978-3-902613-21-9,
pp. 598, December 2007, Itech Education and Publishing, Vienna, Austria

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 usege 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 (Baraf, 1997; Larsen, 2001). 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 (Klančar, 2007).

The work 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. Finally some experimental results and conclusions are given.

2. System Overview

The robot soccer set-up (see Fig. 1) consists of ten 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.

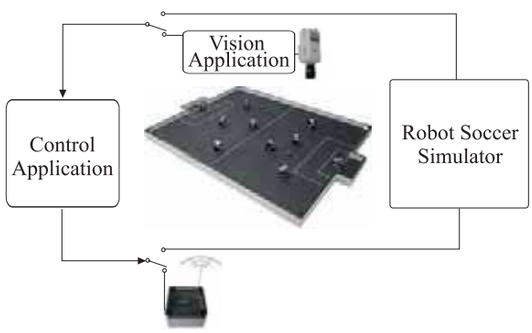

Fig. 1. Robot soccer system overview

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 (Egeland, 2002) 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 (Kolmanovsky & McClamroch, 1995). 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.

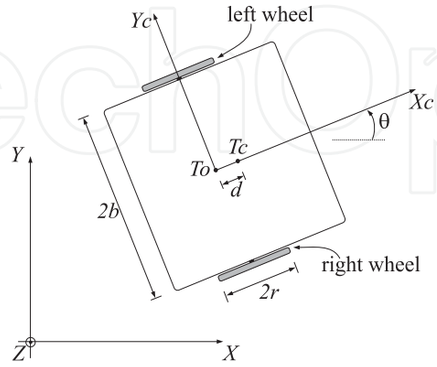

Fig. 2. Symbol description

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 robot orientation, ϕ_r and ϕ_l are angles describing wheels rotation and d is 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:

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

Dynamics motion equation can further be derived using Lagrange formulation (Welles, 1967)

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

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

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

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:

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

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

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

where λ_1, λ_2, λ_3 are Lagrange multiplicators which can effectively be eliminated by the procedure given in (Oriolo et al., 2002; Sarkar, 1994). Brief summary is given in the sequel. Lagrangian formulation (3) can be expressed in matrix form, such as:

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

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 $\mathbf{S}^T$. The part with Lagrangian multiplicators vanish because $\mathbf{S}^T\mathbf{A}^T=\mathbf{0}$.
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.

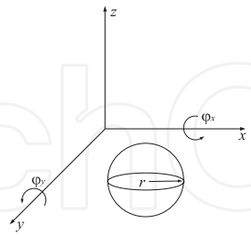

Fig. 3. The ball rolling on the plane

Dynamics motion equation can be derived using Lagrange formulation

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

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), equation (12) is rewritten as

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

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.

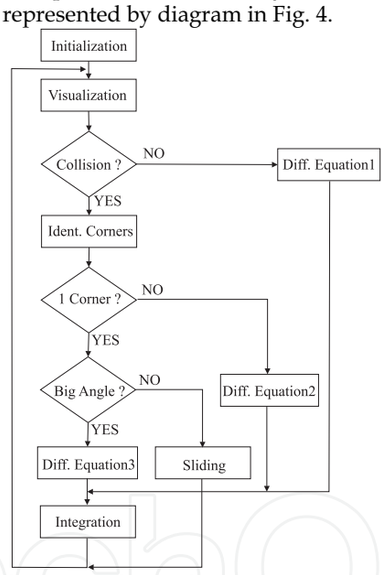

Fig. 4. Robot–boundary collision simulation diagram

The notation Diff. 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 Diff. 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).

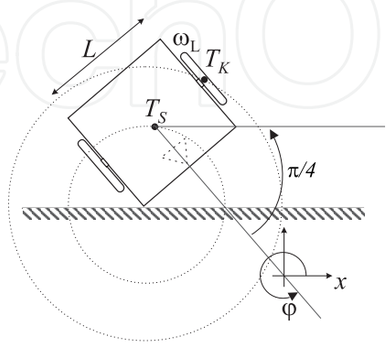

Fig. 5. One-corner collision with the boundary

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 Diff. Equation 3) then becomes:

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

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 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).

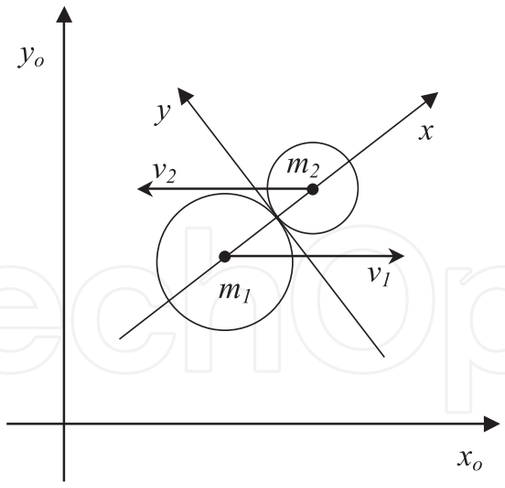

Fig. 6. Collision of two spheres

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:

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

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

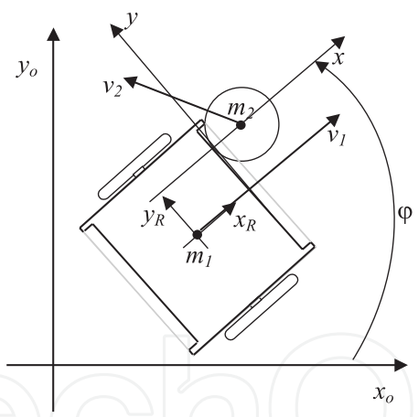

Fig. 7. Robot-ball collision

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 (Egeland, 2002; The Math Works, 1998).
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.

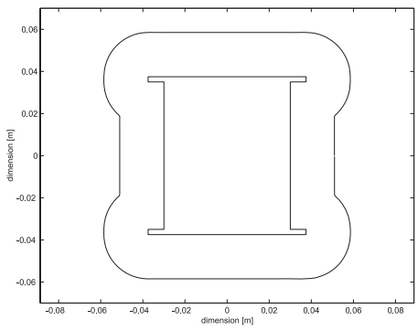

Fig. 8. Shape of the robot (inner) and its rim

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 $tangent(\varphi)$ 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, 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.

Collision Detection

A collision detection algorithm (Klančar et al. ,2003) 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.

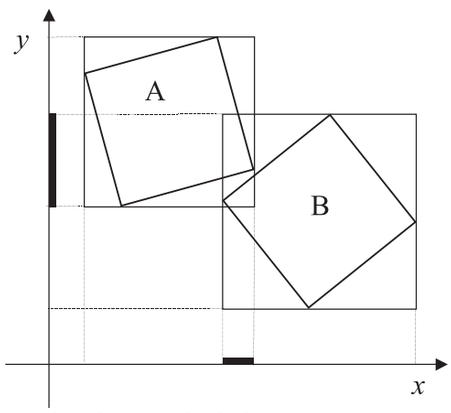

Fig. 9. Overlapping of bounding boxes in both directions

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.

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).

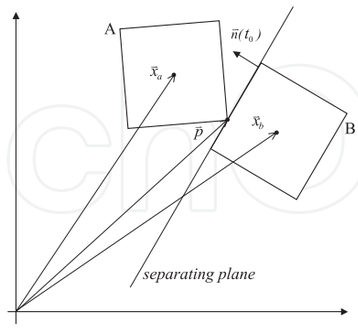

Fig. 10. Collision of two robots

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

$$\Delta\vec{G} = \int\vec{F}dt \tag{26}$$

where $\vec{G}$ stands for conservation of momentum and $\vec{F}dt$ 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 (Baraf, 1997; Klančar et al., 2003). 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.

Force impulse in normal direction $\vec{n}$ of the collision (also normal of the separating plane at the time of the collision, see Fig. 10) of the two frictionless bodies is given by

$$\vec{J} = j\vec{n}(t_0) \tag{27}$$

where t_0 is time of the collision and j is amplitude of the force impulse. For the normal direction of the collision the following relation can be written

$$v_{rel}^{+} = -\varepsilon v_{rel}^{-} \tag{28}$$

meaning that absolute value of relative velocity in normal direction after collision v_{rel}^{+} remains the same or is lowered for energy loss factor ε in comparison with to absolute value of relative velocity in normal direction before collision v_{rel}^{-}. From the property (28) the amplitude of force impulse j in Equation (27) can be estimated according to procedure described in (Baraf, 1997). Let $\dot{\vec{p}}_a^{-}(t_0)$ be the velocity of contact point of robot A before impulse $\vec{J}$ is applied and $\dot{\vec{p}}_a^{+}(t_0)$ velocity of contact point of robot A after applying impulse. Similarly notations $\dot{\vec{p}}_b^{-}(t_0)$, $\dot{\vec{p}}_b^{+}(t_0)$ are used for the second robot B taking part in the collision. Relative velocity in normal direction before applying impulse is thus

$$v_{rel}^{-} = \vec{n}(t_0) \cdot (\dot{\vec{p}}_a^{-}(t_0) - \dot{\vec{p}}_b^{-}(t_0)) \tag{29}$$

and after applying impulse

$$v_{rel}^{+} = \vec{n}(t_0) \cdot (\dot{\vec{p}}_a^{+}(t_0) - \dot{\vec{p}}_b^{+}(t_0)) \tag{30}$$

Defining

$$\vec{r}_a = \vec{p} - \vec{x}_a(t_0) \tag{31}$$

where $\vec{r}_a$ is the displacement vector between mass centre $\vec{x}_a$ of the robot A and collision point $\vec{p}$. Further let $\vec{v}_a^{-}(t_0)$ and $\vec{\omega}_a^{-}(t_0)$ be the liner and angular velocity of robot A before and $\vec{v}_a^{+}(t_0)$ and $\vec{\omega}_a^{+}(t_0)$ after applying force impulse. The following velocities can be written for mass centre of robot A and for the point of collision

$$\vec{v}_a^{+}(t_0) = \vec{v}_a^{-}(t_0) + \frac{j\vec{n}(t_0)}{M_a} \tag{32}$$

$$\vec{\omega}_a^{+}(t_0) = \vec{\omega}_a^{-}(t_0) + I_a^{-1}(\vec{r}_a \times j\vec{n}(t_0)) \tag{33}$$

$$\dot{\vec{p}}_a^{+}(t_0) = \vec{v}_a^{+}(t_0) + \vec{\omega}_a^{+}(t_0) \times \vec{r}_a \tag{34}$$

Here M_a stands for mass of robot A and I is the corresponding moment of inertia. The same notation is used for robot B. Inserting Equations (32) and (33) to Equation (34), the following relation is obtained

$$\dot{\vec{p}}_a^{+}(t_0) = \dot{\vec{p}}_a^{-}(t_0) + j \cdot (\frac{\vec{n}(t_0)}{M_a} + I_a^{-1}(\vec{r}_a \times \vec{n}(t_0))) \times \vec{r}_a \tag{35}$$

The velocity in the contact point of robot B considering opposite direction of impulse force is thus

$$\dot{\vec{p}}_b^{+}(t_0) = \dot{\vec{p}}_b^{-}(t_0) - j \cdot (\frac{\vec{n}(t_0)}{M_b} + I_b^{-1}(\vec{r}_b \times \vec{n}(t_0))) \times \vec{r}_b \tag{36}$$

Inserting Equations (35) and (36) into Equation (30) and then combining obtained equation with Equation (28) the amplitude of impulse is finally calculated as

$$j = \frac{-(1+\varepsilon)v_{rel}^{-}}{\frac{1}{M_a} + \frac{1}{M_b} + \vec{n}(t_0) \cdot \left(I_a^{-1}(\vec{r}_a \times \vec{n}(t_0))\right) \times \vec{r}_a + \vec{n}(t_0) \cdot \left(I_b^{-1}(\vec{r}_b \times \vec{n}(t_0))\right) \times \vec{r}_b} \tag{37}$$

Having estimated the impulse, linear velocity $\vec{v}^{+}$ and angular velocity $\vec{\omega}^{+}$ for robot mass centre can be calculated by using relations (32), (33). It is namely equivalent to impulse force because of collision simulation but more suitable and accurate for realization. To obtain accurate t_0 zero crossing algorithm implemented in Matlab Simulink could be used in order to assure accurate integration of discontinuous velocities signals. This algorithm simply changes integration step by bisection, according to some input variable (distance between robots multiplied by a sign which is negative if robots penetrate), until exact time of discontinuity is achieved. 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 step size of simulation becomes very small which results in halting of the simulation. Thus a better solution is to check for correspondingly small distance between one robot corner and the separating plane belonging to another robot. If separating plane does not exist, the time before penetration of the simulated robots 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.

5. Experimental Validation

In the sequel a few examples of simulator operation will be compared to the operation of a real set-up. In these comparisons similar conditions (initial pose, velocities and situations) are ensured. These visual comparisons give some impression about the capability of the simulator to realistically describe the real set-up.

The situation where the ball collides with the wall and the robot is presented in Fig. 11. Here the robot stands still while the ball starts to move with initial velocity v= 0.8 m/s. In the left graph of Fig. 11 the experiment result from the real set-up is presented while the right one shows a similar simulated experiment. In both figures the object shapes are drawn with 165 ms resolution (simulation sampling time is 33 ms).

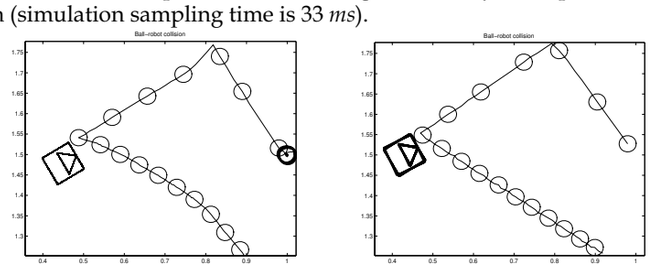

Fig. 11. Comparison of collision between the ball and the wall and between the ball and the robot on a real set-up (left) and on the simulator (right)

In both cases a similar ball motion is recorded. More interesting is the second ball collision where the ball hits the robot and rebounds from the robot specific shape presented in Figs. 7 and 8. The difference between both of thecompared figures is the course of the ball which is supposed to be a straight line on the simulator but in a real set-up it has a slight deviation from the straight line motion. This might happen because of the ball spinning effect after the collision and some other (stochastic) effects such us uneven terrain, dirt on the ground which were not considered in the simulator.

In Fig. 12 the simulated and real robot hits the boundary at the 45° angle relative to the boundary. In both cases the robot starts with constant velocity (v=0.5 m/s).

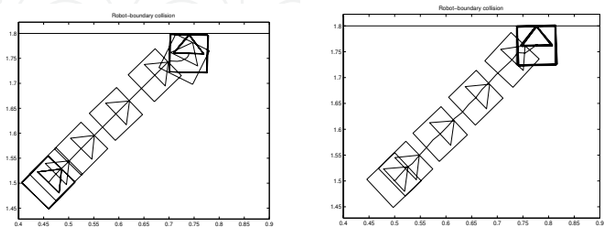

Fig. 12. Comparison of collision between the robot and the wall and between the ball and the robot on a real set-up (left) and on the simulator (right)

It can be observed that both examples in Fig. 12 (real and simulated) are almost identical.
In Fig. 13 comparisons between robots from a real set-up (first column) and simulated robots (second column) for three different collision situations (rows in Figure 13) are given. The experiments were performed with the same initial conditions (starting positions, orientations and velocities).

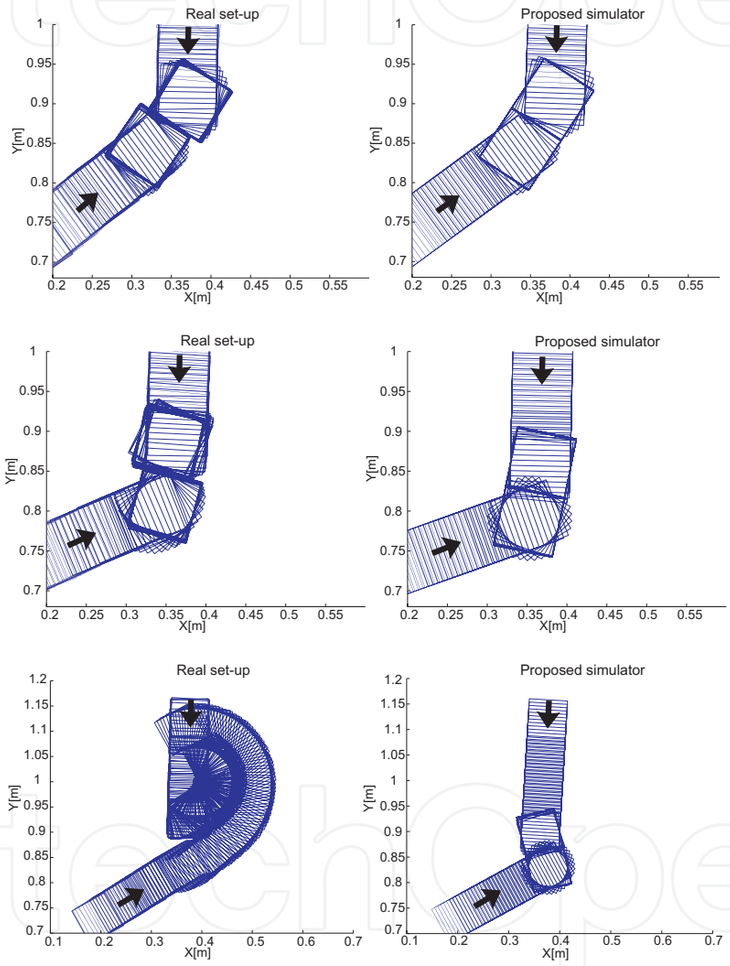

Fig. 13. Comparison of collisions between real robots and simulated robots

From the proposed representation also the estimation of robots course and their speeds in certain time (sample time is 33 *ms*) can be observed. The first and second row of Fig. 13 show the situation where compared subjects are relatively equal. The real situation where

robots wheels slide on the real set-up is shown in the third row in Fig. 13. Here of course simulator gives wrong results. It is evident that the proposed robots collision model captures the behaviour of the real robots to the reasonable extent, which means that simulated situations cover a vast majority of collisions in real game sufficiently well.
Presented collision models give sufficient representation of real situation. However, a lot of factors in real set-up are of significantly stochastic character what means that their modelling is not justifiable from the usable simulator point of view (fast enough on available personal computers, simple enough, etc.). The mentioned factors are: nonuniform friction, dirt or dust on the playground or wheels, shape of the robot, robot strength which depends on battery status, wheel sliding, friction is different for the direction along or perpendicular to the direction of wheels, etc. If comparison is performed over longer time interval shown results are useless due to above reasons. Main goal of the work however is to present reasonably accurate motion and collision models and thus contributes to obtain more realistic simulator, which would be used as a tool in the process of strategy and control algorithms design. Therefore, the validation of the simulator as a whole should be done through transferability of obtained strategy algorithms to the real system. It can be confirmed that the behaviour of the simulator is similar enough to the real setup which means that the designed algorithms on a simulator (strategy and low level control) can be without modifications directly used also in real games. The simulator was tested in a number of European and World competitions in FIRA MiroSot league (real robots) category. There the game strategies used in real competitions were mostly developed by using the presented simulator.

6. 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.

7. Acknowledgment

The authors would like to acknowledge the Slovenian Research Agency under CRP MIR M2-0116 project for partly funding this work.

7. References

Baraf, D. (1997). An Introduction to Physically Based Modeling: Rigid Body Simulation II – Nonpenetration Constraints, In: *SIGGRAPH '97 Course note*s, Carnegie Mellon University.

Egeland, O. & Gravdahl , J. T. (2002). *Modeling and Simulation for Automatic Control*, Marine Cybernetics, Trondheim, Norway.

Ferber, J. (1999). *Multi-Agent Systems, An Introduction to Distributed Artificial Intelligence*, Addison-Wesley, Essex, England.

Fremond, M. (1995). Rigid bodies collisions. *Physical Letters*, Vol. A, No. 1, 34-41.

Klančar, G.; Lepetič, M.; Karba, R. & Zupančič, B. (2003). Robot soccer collision modelling and validation in multi-agent simulator. *Mathematical and computer modelling of dynamical systems*, Vol. 9, No. 2, 137-150.

Klančar, G. (2007). Mobile Robot Simulator, available at: *http://msc.fe.uni-lj.si/PublicWWW/Klancar/RobotSimulator.html*.

Kolmanovsky, I. & McClamroch, N. H. (1995). Developments in Nonholonomic Control Problems, *IEEE Control Systems*, Vol. 15, 20-36.

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

Liang, T. C. & Liu, J.S. (2002). A Distributed Mobile Robot Simulator and a Ball Passing Strategy, *Technical Report TR-IIS-02-007*, Institute of Information Science, Academia Sinica, Nankang, Taiwan.

Moss, S. & Davidsson, P. (2002). *Multi-Agent-Based Simulation*, Springer-Verlag, New York.

The Math Works, Inc., Simulink (1998). *Dynamic System Simulation for Matlab*, Natick, USA.

Oriolo, G.; Luca, A. & Vandittelli, M. (2002). WMR Control Via Dynamic Feedback Linearization: Design, Implementation, and Experimental Validation. *IEEE Transactions on Control Systems Technology*, Vol. 10, No. 6, 835-852.

Sarkar, N.; Yun, X. & Kumar, V. (1994). Control of mechanical systems with rolling constraints: Application to dynamic control of mobile robot. *The International Journal of Robotic Research*, Vol. 13, No. 1, 55-69.

Stone, P. & Veloso, M. (2000). Multiagent Systems: A Survey from a Machine Learning Perspective. *Autonomous Robots*, Vol. 8, 345-383.

Welles, D. A. (1967). *Lagrangian Dynamics*, Schaum's Outline Series, McGraw Hill Book Company.

Robotic Soccer

Edited by Pedro Lima

ISBN 978-3-902613-21-9

Hard cover, 598 pages

Publisher I-Tech Education and Publishing

Published online 01, December, 2007

Published in print edition December, 2007

Many papers in the book concern advanced research on (multi-)robot subsystems, naturally motivated by the challenges posed by robot soccer, but certainly applicable to other domains: reasoning, multi-criteria decision-making, behavior and team coordination, cooperative perception, localization, mobility systems (namely omni-directional wheeled motion, as well as quadruped and biped locomotion, all strongly developed within RoboCup), and even a couple of papers on a topic apparently solved before Soccer Robotics - color segmentation - but for which several new algorithms were introduced since the mid-nineties by researchers on the field, to solve dynamic illumination and fast color segmentation problems, among others. This book is certainly a small sample of the research activity on Soccer Robotics going on around the globe as you read it, but it surely covers a good deal of what has been done in the field recently, and as such it works as a valuable source for researchers interested in the involved subjects, whether they are currently "soccer roboticists" or not.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen