MR-Simulator: A Simulator for the Mixed Reality Competition of RoboCup*

Marco A.C. Simões[1], Josemar Rodrigues de Souza[1,2], Fagner de Assis Moura Pimentel[1], and Diego Frias[1]

[1] Bahia State University (ACSO/UNEB), Salvador, BA, Brazil
[2] Integrated Center of Manufacture and Technology (SENAI-CIMATEC), Salvador, BA, Brazil
{msimoes,josemar}@uneb.br,
{diegofriass,fagnerpimentel}@gmail.com

Abstract. Simulations play an important roll in the development of intelligent and collaborative robots. In this paper we describe the development of a simulator for the RoboCup Mixed Reality (MR) competition. MR uses a mix of simulated and real world to foster intelligent robotics development. Our simulator "virtualizes" the players within the MR hardware and software framework, providing the game server with the player-positional information usually supplied by the central vision system. To do that the simulator, after receiving the action commands from each agent (player) must track its expected trajectory and behavior in collision events for all the players. We developed fast algorithms for simulating the robot motion in most usual cases. The simulator was validated comparing its results against the real environment and proved to be realistic. This tool is important for setting-up, training and testing MR competition teams and could help to overcome current difficulties with robot hardware acquisition and maintenance.

1 Introduction

Simulators are important tools for development and support of RoboCup. Can be used in various leagues with the purpose of testing the performance of teams prior to competition in the real environment. The RoboCup simulation league is divided into three distinct competitions: 2D, 3D and Mixed-Reality (MR). The first two competitions are fully simulated while the last has a virtual environment (simulated) in which physical robots interact [1]. The virtual game environment comprising the field, the ball and football beams is simulated by a server which communicates with software agents [1] providing to them the position of each player in the field and trajectory of the ball. Each software agent processes that information and take decisions for the best movement and behavior of the corresponding player in each time step. Action commands are then transmitted by infrared interface to the physical robots. This paper presents the substitution of physical robots by a software module that we have called MR-simulator.

* This project is partially funded by UNEB, and CNPQ. The authors thank the great help and provision of Juliana Fajardini Reichow in elaboration of this article.

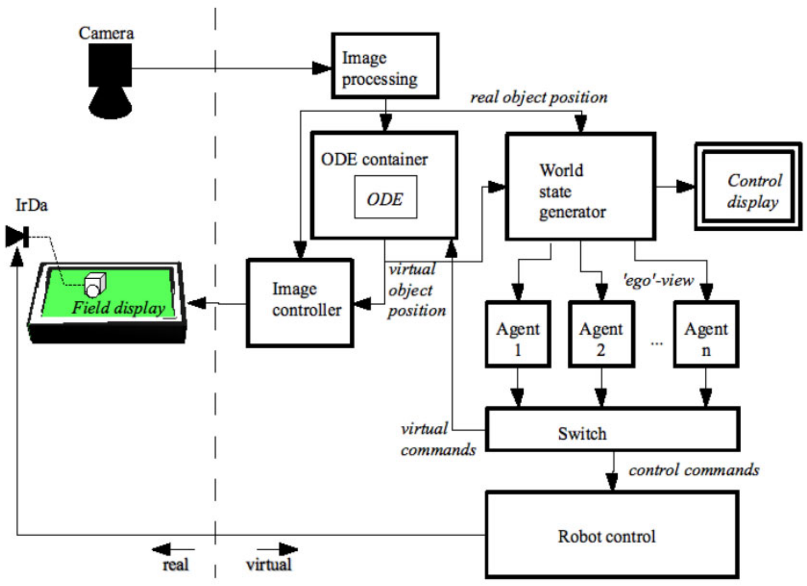


Fig. 1. Structure of RoboCup Mixed Reality system. Figure taken from [1].

The infrastructure of MR competition consists of: (a) camera and vision-tracking (VT) system which guarantee the capture of robots position and movement direction in the field, (b) infra-red (IR) transmitters located in the corners of the field driven by a Robot Control (RC) system and receivers in the robots which guarantee the command of the robots, and (c) a projection system comprised by an LCD monitor and software that projects the soccer field, as well as the moving objects: ball and robots, in the LCD screen. The LCD screen is put in horizontal position allowing the physical robots to move over its flat surface. The real robots interact each other and with virtual objects displayed on the screen [1]. In this infrastructure, the server (MR-SoccerServer [2]) is responsible for the simulated environment and virtual objects, and supervises the soccer game. The robots are controlled by software agents, the infrared transmitters and the camera are the actuators and sensors in real environment (Fig. 1). MR-Simulator communicates with the MR-Soccerserver emulating the IR-RC and VT tracking systems.

Among the features of MR-Simulator there are: simulation of IR-RC and VT (connection and messages transmission) systems, real robots simulation (calculation of trajectories and collisions) and repositioning of the robots in the field. The main features are explained in section 3. Section 2 contains a brief review of the RoboCup Mixed Reality competition history and describes some mixed reality aspects. In section 4 we discuss the results of tests with the MR-Simulator. The conclusions and future works are addressed in section 5.

2 The RoboCup Mixed Reality Competition

The Mixed Reality competition is part of the simulation league of RoboCup. It was proposed in RoboCup 2006 under the name Physical Visualization (PV) [3]. Its goal is to provide a bridge between simulated and physical leagues, using the concepts of mixed reality [1,4,5].

Mixed reality is defined as a mix between real and virtual, or the overlap of the virtual world with the physical. It is divided into augmented reality and augmented virtuality in Milgram's real-virtual continuum [6,7]. Augmented reality is the insertion of virtual objects in the real world, while augmented virtuality is the insertion of real objects into the virtual world. The mixed reality that occurs in the competition MR is augmented virtuality, where robots (real objects) are inserted in a virtual environment composed by field, ball and football beams simulated by the server of the competition.

The MR complete environment set is formed by a high resolution camera, infra-red transmitters, a flat screen or projector, where field and virtual objects are seen, and the robots [1]. Robots are identified on the field by the camera, and their positions and orientations are sent to the VT, and then to the server. The server sends data about all real objects on the field to the clients, which process their decisions and pass the desired wheel velocities (varying from 0 to 130mm/s, positive, negative) back to the server, so it can send them as commands to the robots on the field through the RC system. Inside the RC, commands are discretized based on Table 1, and sent to robots through the infra-red transmitters. The micro-robot currently used in MR competitions has small size (28x25x27 mm) having an almost cube shape (Fig. 2).

Since its creation the MR competition has been gradually evolving, both in software and hardware. However, still remain some platform development challenges. For example VT, RC and even the robots used are too much unstable. We have observed that VT can lose robot's positions due to calibration errors or improper lighting. In some cases

Table 1. Speeds's Discretization. Values taken from [8].

Code	speed in mm/s	Code	speed in mm/s
0	0	16	44.80
1	25.61	17	47.15
2	27.17	18	49.77
3	28.54	19	52.65
4	29.72	20	55.81
5	30.76	21	59.24
6	31.71	22	62.95
7	32.59	23	66.96
8	33.48	24	71.33
9	34.39	25	76.19
10	35.39	26	81.78
11	36.51	27	88.59
12	37.77	28	97.48
13	39.21	29	110.16
14	40.84	30	130.43
15	42.70	-	-

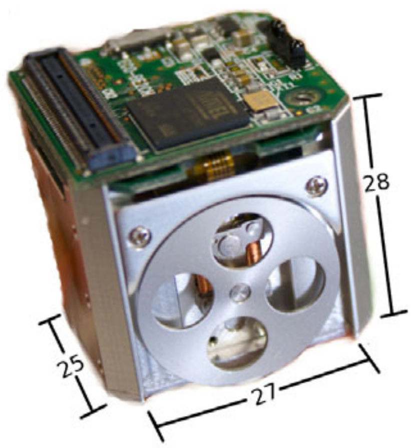


Fig. 2. MR micro-robot and its dimensions (in milimeters)

RC it was not able to pass commands to all robots due to infra-red interference. However, this problem has been solved by increasing the number of infra-red transmitters around the field. Moreover, robots have both hardware and software problems: robots can translate and/or execute erroneously sent commands. In spite of the effort of the participating teams in solving those problems, since 2009 the MR was presented only as a Demo Competition on RoboCup.

3 The MR-Simulator

The motivation for developing the MR-Simulator emerged from the difficulties to work with such still unstable mixed platform. MR soccer team developers need to test tactics and strategies in the field in order to improve the defensive and offensive performance of their teams, and that is not really possible with the current infrastructure. Authors believe that MR-simulator provides an stable environment for team training and testing, as well as could allow the expansion of the competition providing a cheaper environment. Using MR-Simulator it is possible to dispense robots, infra-red transmitters (RC), camera (VT) and big sized monitors or projectors, which can be an alternative for beginners and/or for preliminary tests. Besides the relatively high cost of such infrastructural items there is also an economy in time and resources needed for support and maintenance of such infrastructure, when using MR-Simulator. The MR fully simulated environment makes possible a faster spread of the competition, widening educational applications possibilities, one of the league's primary goals - together with gradual platform development [9,10,11].

Nowadays, we can find good simulators that could be used in the MR competition, for example Webots [12] and Spark [13]. Both are based on the same simulation engine Open Dynamics Engine [14] used by MR-SoccerServer. ODE provides software objects to model physical objects and joints and to handle collisions which makes it suitable for simulating also 3D competition. However, in our case we decided to develop a simulator free of dependencies with any simulation library. Two main reasons justified that decision: (1) ODE is still incomplete and needs substantial improvements in the documentation and, (2) we aimed to construct simplified kinetic models that better simulates the movement and collisions of the robot-players in the real MR soccer environment.

MR-simulator is more focused on competitor's needs and expectations. Spark, for instance, offers many features that are unnecessary for MR, and is in constant evolution, which requires frequent readjustments of the competition structure in order to maintain the compatibility. It should be said that a first attempt was done trying to use the 2D competition server (Soccer Server [15]) for MR simulations, however it was difficult to control agent's behavior under the classical noisy environment of the 2D competition.

3.1 Simulation of the Environment

In order to simulate VT and RC, virtual modules were created for message exchange with the server: the virtual vision tracking (V-VT) and the virtual Robot Control (V-RC). Messages sent by V-VT are based on an initial robot position and on trajectory and collision models, explained in section 3.2. V-VT sends messages containing virtual position and orientation of robots to the server, which passes client commands to the V-RC. V-RC discretizes the velocities according with Table 1 and sends them to the virtual robots. Trajectory and collision models are used to compute robot's next positions and restarts the cycle, until the game ends, as can be seen on Algorithm 1. In our implementation the MR-SoccerServer interface was not modified, that is, the server behaves in the same way that when communicates with real VT and RC.

3.2 The Robot's Simulation

As the camera is located on top of the field, the height of the robot is neglected in the simulation. The robot is modeled as a flat figure of 25 x 27 mm.

The robots are simulated with virtual robots which have name, ID, orientation, size and wheels speed, based on data passed by the agents. The robots simulation is made with the trajectory and collision models.

Algorithm 1. Main Loop of MR-Simulator

```
while $!EndGame()$ do
  if $RC.ReceiveRobotsVel()$ then
    for $Robots.begin : Robots.end$ do
      $UpdateVelRobots()$
      $ModelCollisionRobot - Robot()$
      $ModelCollisionRobot - Wall()$
      $ModelTrajectory()$
    end for
  end if
end while
```

Trajectory Model. Our model considers two discretized and coupled time scales for generality and numerical stability purposes. Consider first a succession of cycle times $t_i = t_0, t_1, ..., t_n$, such that $t_{i+1} = t_i + \varDelta T_c$, at which the velocities $v_{L,i}$ (left) and $v_{R,i}$ (right) of the wheels of the robot can be updated via the RC system. Here $\varDelta T_c$ stands for the "cycle time" which depends on the server configuration. Within a time interval $I_i = [t_i, t_{i+1})$ the velocities of both wheels are kept constant and equal to the velocity set at $t = t_i$, that is, $v_L(t) = v_{L,i}$ and $v_R(t) = v_{R,i}$ for $t \in I_i$. However, the position of the wheels are updated at smaller time steps $\tau_k = \tau_1, \tau_2, \ldots, \tau_m$ such that $\tau_1 = t_i$ and $\tau_m = t_{i+1}$. That is, we set $\tau_{k+1} = \tau_k + \varDelta T_s$, for $k = 1, 2, \ldots, m$ where $m = \varDelta T_c / \varDelta T_s$ is an integer greater or equal 1. $\varDelta T_s$ is a prefixed "simulation time step" adjusted for obtaining the desired precision in the simulation and is subject to a natural constraint $\varDelta T_s \leq \varDelta T_c$. Therefore, during a cycle time interval m simulated displacements of the robot must be calculated. Let's denote as $(x_{L,k}, y_{L,k})$ and $(x_{R,k}, y_{R,k})$ the position of the left and right wheels, respectively, of the robot at a given simulated time τ_k in a x, y plane domain representing the football field. Let a_0 be the distance between wheels and f a correction factor. At the beginning of each server time interval I_i, $i = 1, 2, n$ set $(x_{L,1}, y_{L,1})$ and $(x_{R,1}, y_{R,1})$ according with the current position of the left and right wheel, respectively, sent by the V-VT system. Then, using the velocities $v_{L,i}$ and $v_{R,i}$ sent by the agent for this cycle, compute

$$e_y = y_{R,1} - y_{L,1}$$
$$e_x = x_{R,1} - x_{L,1}$$
$$c_t = e_y / a_0$$
$$s_t = -e_x / a_0$$
$$m = \varDelta T_c / \varDelta T_s$$

and then for $k = 1, 2, ..., m$ do:

1. Calculate predicted wheel position in the next simulation time step
   - $x_{L,k+1} = x_{L,k} + v_{L,i} c_t \varDelta T_s$
   - $x_{R,k+1} = x_{R,k} + v_{R,i} c_t \varDelta T_s$
   - $y_{L,k+1} = y_{L,k} + v_{L,i} s_t \varDelta T_s$
   - $y_{R,k+1} = y_{R,k} + v_{R,i} s_t \varDelta T_s$
2. Compute correction terms
   - $e_y = y_{R,k+1} - y_{L,k+1}$
   - $s_y = sign(e_y)$
   - $e_x = x_{R,k+1} - x_{L,k+1}$
   - $s_x = sign(e_x)$
   - $e_a = \sqrt{e_y^2 + e_x^2} - a_0$
   - For velocity dependent correction set $f_R = v_{R,i} / (v_{R,i} + v_{L,i})$ else $f_R = f$
   - $f_L = 1 - f_R$
3. Calculate corrected wheel position in the next simulation time step
   - $x_{R,k+1} = x_{R,k+1} - s_x e_a f_R$
   - $x_{L,k+1} = x_{L,k+1} + s_x e_a f_L$
   - $y_{R,k+1} = y_{R,k+1} - s_y e_a f_R$
   - $y_{L,k+1} = y_{L,k+1} + s_y e_a f_L$

After each cycle MR-simulator plots the virtual robot placing the left and right wheels at coordinates $(x_{L,k}, y_{L,k})$ and $(x_{R,k}, y_{R,k})$, respectively. When the last simulation time interval ends, send to the V-VT the position of the centroid of each robot calculated as the mean of the wheel coordinates, that is $X = 0.5 * (x_{L,m} + x_{R,m})$ and $Y = 0.5 * (y_{L,m} + y_{R,m})$.

Collision Model. The collision model is divided into robot-wall collisions and robot-robot collision. In both types of collision it uses the concept of slip, a reduction or increase in the actual speed of the wheels depending on the type of collision. When MR-simulator detects an immediate future collision scenario, that is, when the body of a robot-player is expected to be not entirely placed within the field (robot-wall collision case) or when it is expected to intersect other robot body (robot-robot collision case) in the next simulation time τ_{k+1}, the kinetics is altered by using modified wheel velocities $v^*_{L/R,k}$ instead of $v_{L/R,k}$, that are the velocities set by the agent without taking into account the collision. To do that, the actual velocity of the wheels are calculated as $v^*_{L/R,k} = (1 - s_{R/L,k})v_{L/R,k}$ where $s_{R/L}$ is an slip factor that formally varies between a negative lower bound and an upper bound greater than or equal to one and is different for each wheel depending on the collision conditions.

Negative slip causes wheel acceleration with respect to its velocity prior collision due to impulse transfer from a colliding object. This occurs when the colliding object was moving faster that the collided object in a non-opposite direction. For example for a slip value equal to -1, the actual wheel velocity is the double of the expected velocity of the wheel prior collision. A null slip does not change the velocity of the wheel, but as the slip increases toward one the wheel velocity decreases, stopping when slip is equal to one. This happens when the robot collides with an object moving slower in the same direction or with an object moving and colliding from an opposite direction. Therefore, when a wheel is expected to stop its slip factor is set to one, independent of the velocity set by the agent at the beginning of the cycle. That is the case when a robot collides with a virtual field limiting fence. This way the robot is limited to be within the field. Finally when the sleep exceeds one it implies a reversion of the expected rotation of the wheel. The actual reverse velocity depends on the magnitude of the slip. For example, when the slip is equal to two, the actual velocity of the wheel has the same expected value but in opposite direction.

The total slip (sum of the left and right wheel slips $s_k = s_{L,k} + s_{R,k}$) was set as a function of the absolute value of the resultant velocity after collision, which is proportional to the remaining kinetic energy of the colliding objects assumed to stay together forming a single object after collision. In the case of robot-wall collision it depends only on the velocity of the robot. The total slip is then distributed between left and right wheels which depends on the collision geometrical conditions, considering torque.

During collisions the velocities of the wheels of the colliding objects are updated at each simulation time τ_k, that is the wheel velocity is set equal to the modified velocity. Such modified velocity is then used for the next simulation step and a collision status is sent to the agents of the colliding robots to update de finite state machines. The non-collision status is reestablished when the collision algorithm does not predict physical contact of the robots in the next time step.

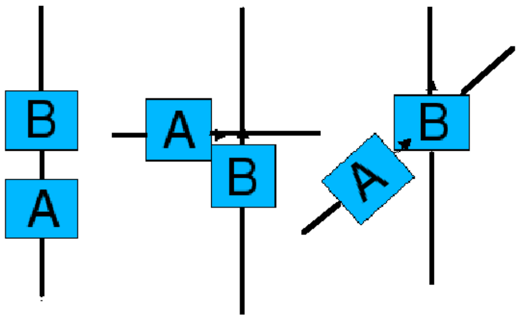


Fig. 3. a) Parallel b) Perpendicular c) Oblique

When simulating robot-robot collision we firstly identify the type of collision considering three categories: (1) parallel, when the robot collides with the front or back side, (2) perpendicular, when collides with a lateral (left or right) side, and (3) oblique when collide with one of its corners, as illustrated in Figure 3. The latter case is currently treated as a parallel collision when the colliding corner bump with another robot's front or back side, and as perpendicular when it collides with the left or right side of another robot. The parallel and perpendicular collisions are treated separately, always analyzing two robots a time: a colliding robot (A) and a collided robot (B).

For example, to find the collision point at robot A we must verify if any of the four vertex of robot B (considering the orthogonal projection of the 3-dimensional robot onto the x, y plane) will be located within the region occupied by the robot A in the next simulation time step. This should be done also for robot B with respect to robot A. In Figure 4 we show an schematic representation of a collision. $V = (x_v, y_v)$ is the colliding vertex and (x_r, y_r) are the coordinates of the collision point, with respect to a reference system located in the center of mass of the collided robot. In that figure $b = a_0$ is the size of the robot and θ is the angle of rotation of the reference system. Therefore a vertex (x_v, y_v) is inside a robot with centroid located at x_c, y_c if it satisfies equations 1 and 2 [16].

$$-a_0/2 \leq x_r \leq a_0/2 \tag{1}$$

$$-a_0/2 \leq y_r \leq a_0/2 \tag{2}$$

where

$$x_r = \sqrt{(x_v - x_c)^2 + (y_v - y_c)^2}\ cos(tan^{-1}(\frac{y_v - y_c}{x_v - x_c}) - \theta)$$

and

$$y_r = \sqrt{(x_v - x_c)^2 + (y_v - y_c)^2}\ sin(tan^{-1}(\frac{y_v - y_c}{x_v - x_c}) - \theta)$$

With these formula we check whether some vertex of A collides with B and/or some vertex of B collides with A, defining the collision type point using the rules in Table 2.

Table 2. Defining the collision point

Vertex of A bumps into B	Vertex of B bumps into A	Collision point
True	False	where A intersects B
False	True	where B intersects A
True	True	middle point of A and B vertices
False	False	no collision case

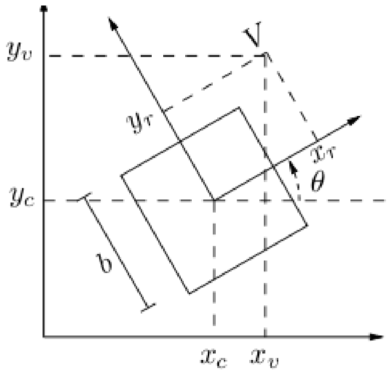


Fig. 4. Relevance of a point to a square. Figure taken from [16].

When both robot A and B have a vertex colliding into the other robot it indicates parallel or perpendicular collision. Otherwise, an oblique collision is taking place.

After a robot-robot collision state is predicted (the simulator predicts that collision will happen in the next simulation time) or confirmed (the collision began in a previous simulation time but the robots are still in contact) we calculate the after-collision velocity C as the average vectorial velocity of the two robots A and B (3).

$$C = \frac{V_A + V_B}{2} \tag{3}$$

The total slip, S, in parallel collision case for both robots, A and B, are defined as the absolute difference of the colliding robot speed ($V_X = V_A$ or V_B) and the after-collision velocity C with respect to the robot speed, that is.

$$S_X = \frac{||V_X - C||}{||V_X||} \tag{4}$$

Parallel Collision. The total slip S_X of each robot $X = A \; or \; B$ is distributed between left and right wheels, considering the mechanical torque (T), that tends to rotate or flip objects. The torque in this case is defined by the vectorial product

$$T_X = r_X \times (V_A - V_B) \tag{5}$$

where r_X is a vector pointing from the centroid of robot $X = A$ or B to the collision point, calculated as described above.

Let us denote by Y and $\bar{Y}$ two different states of the robot wheels which depends on the relative position and motion direction of the colliding robots, in such a way that when one wheel of the robot is in state Y the other necessarily is in state $\bar{Y}$. The wheel that is in state Y is always that being in the right side of the robot with respect to the direction of motion. Thus, when the robot moves forward (positive velocity) the right wheel is in state Y and the left wheel in $\bar{Y}$ state. When the robot moves backward (negative velocity) the left wheel is in state Y and the right wheel is in state $\bar{Y}$. The slip calculation described below depends only on the state of the wheel.

The slip of the wheel in state Y of robot $X = A$ or B is defined by

$$S_{Y,X} = \frac{1 + ||T_X||}{2\,||T_{max}||} \tag{6}$$

where T_X is given by equation 5 and T_{max} is the maximum expected torque given by

$$T_{max} = r_{max} \times (V_A + V_B) \tag{7}$$

where r_{max} is the distance of the centroid of the robot to a corner. The slip of the other wheel of the robot X (being in state $\bar{Y}$) is then given by

$$S_{\bar{Y},X} = 1 - S_{Y,X} \tag{8}$$

The modified speed of the wheel being in a state $W = Y$ or $\bar{Y}$ of the robots $X = A$ and B is in this case calculated as

$$v^*_{W,X} = (1 - S_{W,X})\,C \tag{9}$$

and the modified robot velocity is then given by

$$V^*_X = 0.5\,(v^*_{Y,X} + v^*_{\bar{Y},X}) \tag{10}$$

Perpendicular Collision. It was observed that in most cases when a perpendicular collision occurs the colliding robot (A) and the collided robot (B) depict different behaviors. Robot A rotates over its own axis following the movement of robot B which rotates

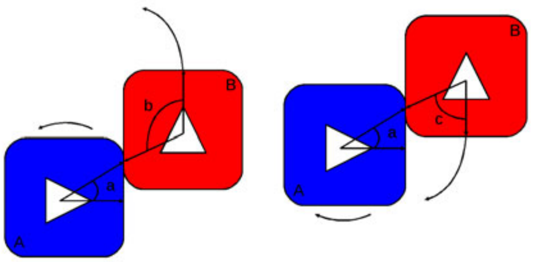


Fig. 5. Perpendicular Collision. 'a' is the collision angle of robot A, and 'b' and 'c' are the collision angles of robot B for positive and negative velocity, respectively.

around robot A, as shown in Figure 5. The orientation of robot B defines to which side robot A will turn. This rotating coupled motion usually continues until robot A retreats.

To simulate this movement within the slip model, we need to take into consideration two variables: (1) Robot's orientation, identifying which robot collided with the front or back side (A) and which collided with a lateral side (B), and (2) the angle of collision β for each robot. β is the angle spanned by the vector joining the center of mass of the robot and the collision point (middle point of the projected contact surface of the robots) and the vector of motion originated also in the center of mass, as indicated in Figure 5. Angles spanned clockwise with respect to the motion direction are assumed to be positive. Counterclockwise are considered negative angles.

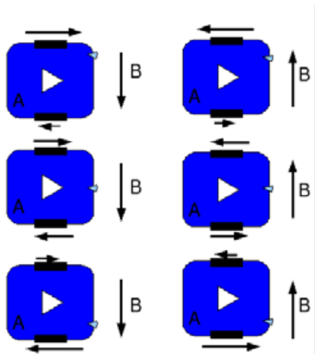
For colliding robot A the slips are determined as linear functions of the collision angle $\beta_A \in [-\pi/4, \pi/4]$. The slip of the wheel in state Y is defined by equation 11,

$$S_{Y,A} = 1 - \frac{\sigma}{2}\left(1 + \frac{\beta_A}{\pi/4}\right) \tag{11}$$

where $\sigma = sign(V_A \times V_B) = \{-1, +1\}$. Notice that $S_{Y,A} \in [0, 2]$. The slip of the opposite wheel (in state $\bar{Y}$) is calculated by equation 12

$$S_{\bar{Y},A} = S_{Y,A} + \sigma. \tag{12}$$

For illustration, in the the left part of Figure 6 we represented six cases corresponding to angles $\beta_A = \{-\pi/4, 0, \pi/4\}$ (from top to bottom) and $\sigma = \{-1, +1\}$ (left and right). The collision point is represented with a blue spot inserted in the collided robot surface (side). The arrows in the wheels represent the modified velocities after collision. Prior to collision both wheels were moving with the same velocity. In the right part of the same Figure 6 we plotted the slip of both wheels (Y, $\bar{Y}$) for positive and negative σ as a function of β, using Eq. 11 and 12.



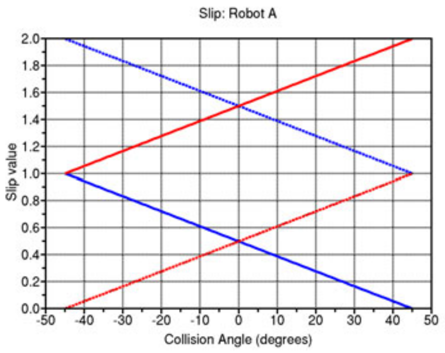


Fig. 6. LEFT: Wheel velocities of robot A in 3×2 typical perpendicular collision cases. From top to bottom the collision angles are $\beta_A = \{-\pi/4, 0, \pi/4\}$. In the left column cases $\sigma = -1$ and $\sigma = +1$ in the right column cases. RIGHT: Slip setting as a function of β_A, for positive (blue) and negative (red) σ. Continuos and dashed lines correspond to Y and $\bar{Y}$ wheels, respectively.

The slip of the wheel W in the collided side of robot B was modeled with a quadratic function of the absolute value of the collision angle $|\beta_B| \in [\pi/4, 3\pi/4]$ in the form

$$S_{W,B}(\beta_B) = c_0 + c_1 |\beta_B| + c_2 |\beta_B|^2, \tag{13}$$

while the velocity of the wheel in the other side, $\bar{W}$, was not modified in this case, that is, we set $S_{\bar{W},B} = 0$. The coefficients in Eq. 13 were calculated to satisfy $S_{W,B}(\pi/4) = 1$, $S_{W,B}(\pi/2) = 1/3$ and $S_{W,B}(3\pi/4) = 0$.

For illustration we represented in the left part of Figure 7 the velocities of the wheels of the collided robot B in six typical perpendicular collision cases. In the right part the slip of the wheel in the bumped side is plotted as a function of $|\beta_B|$ using Eq. 13.

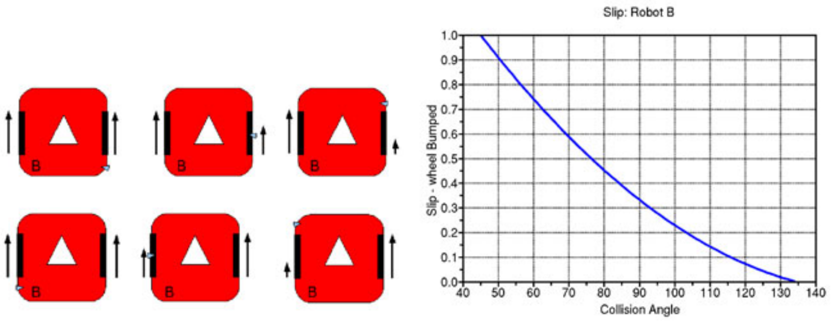


Fig. 7. LEFT: Wheel velocities of robot B when collided from the right (top) and from the left (bottom) for collision angles $\beta_B = \{\pm 3\pi/4, \pm\pi/2, \pm\pi/4\}$ varying from left to right. RIGHT Bumped wheel slip as a function of $|\beta|$.

4 Discussion of Results

MR-Simulator was tested as the software modules were released as follows: VC and RC virtual modules connection; VT and RC virtual modules message transmission; Trajectory model; Collision model; and Robots repositioning in field.

We tested using an autonomous client, the BahiaMR [17], to verify the behavior of the team using fully simulated environment comparing its behavior with that observed using real mixed reality environment (videos of competition official matches).

4.1 Connection Tests

Tests of connection and transmission of messages from virtual modules and repositioning were ajectory done successfully, checking whether connections were made correctly, or messages delivered flawlessly at the right time and if the robots were repositioned in the correct positions and orientations.

4.2 Kinetic Model Test

The kinetic model that predicts the trajectory of the robot was tested using a code written in Scilab [18]. The kinetic model test program consisted on the evaluation of the accuracy of wheel positions for different simulation time steps. As a result a simulation time step ΔT_s of 0.001 seconds was chosen. After approval of the kinetic model a C++ version was created and built within the MR-simulator.

The collision module was directly written in C++ within the MR- simulator environment. The test and parameter adjustment program consisted on: (1) Evaluation of dependence on collision intensity; (2) Calculation of the total slip factors with different options; and (3) Calculation of wheel slip factors on parallel and perpendicular collisions. As a result the formulas described above were chosen.

4.3 Test with Clients

The tests with the client were made by analyzing its behavior in a totally simulated environment. The clients were firstly programmed to perform simple trajectory movements. These movements were observed compared with those showed in the competition videos. Then, robots were programmed to perform movements that resulted in collisions. The collisions were made in various angles and with different robot speeds. The movements performed by clients in the tests were go forward, go back, go to the ball, turn around the axis (spin) and make curves to trajectory model and execute parallel, perpendicular and oblique's collision in different directions, angles and speeds to collision model.

The result was a behavior similar to those seen in the videos. In these tests, we observed great stability even when the field size and number of robots per team was changed. That makes it possible to simulate the game regardless of field size and number of robots, different from what happens today in the competition, which runs 4x4 to 6x6 robots matches over a 42" flat screen. In Figure 8 it is possible to observe simulations with 5 players per team and with 11 players per team.

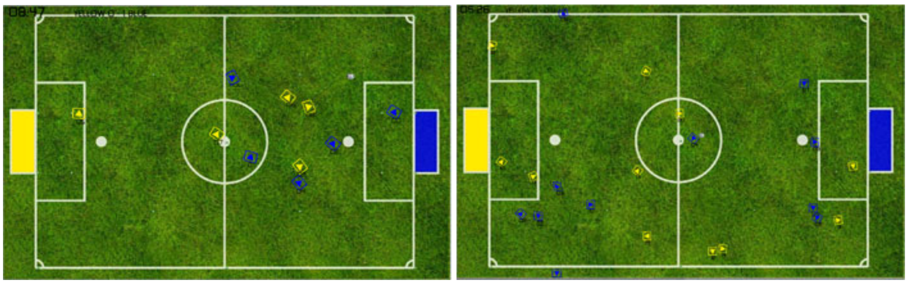


Fig. 8. Left: simulation with 5 players per team Right: simulation with 11 players per team

5 Conclusions and Future Work

This paper presented the MR-simulator, a tool to aid team development of RoboCup Mixed Reality competition and facilitate league's expansion. We described its features and mathematical models used to compute robot's trajectory and collision events. The MR-Simulator has been used since the last official competition, RoboCup 2009, held in June and July 2009 in Graz, Austria.

In future works we will address the implementation of a graphical interface for the simulator itself, to serve as viewer and improve the way of repositioning the robots. Moreover, we plan to develop features to control the robot manually in a debug mode; nowadays robots are controlled only by autonomous clients. Controlled noises, usual on real environments will also be added, in a configurable way, affecting messaging, positioning and robots movement (e.g. wheels locks) in order to simulate more realistic scenarios.

References

1. Gerndt, R., Schridde, C., da Silva Guerra, R.: On the aspects of simulation in the robocup mixed reality soccer systems. In: Workshop at International Conference on Simulation, Modeling and Programming for Autonomous Robots (2008)
2. Silva, G., Cerqueira, A.J., Reichow, J.F., Pimentel, F.A.M., Casaes, E.M.R.: MR-SoccerServer: Um Simulador de Futebol de Robôs usando Realidade Mista. In: Workshop de Trabalhos de Iniciação Científica e de Graduação (WTICG), pp. 54–63 (2009)
3. Guerra, R., Boedecker, J., Yamauchi, K., Maekawa, T., Asada, M., Hirosawa, T., Namekawa, M., Yoshikawa, K., Yanagimachi, S., Masubuchi, S., et al.: CITIZEN Eco-Be! and the RoboCup Physical Visualization League (2006)
4. Stapleton, C., Hughes, C., Moshell, J.: Mixed reality and the interactive imagination. In: Proceedings of the First Swedish-American Workshop on Modeling and Simulation, pp. 30–31 (2002)
5. Azuma, R.: A survey of augmented reality. Presence-Cambridge Massachusetts- 6, 355–385 (1997)
6. Milgram, P., Takemura, H., Utsumi, A., Kishino, F.: Augmented reality: A class of displays on the reality-virtuality continuum. In: Telemanipulator and Telepresence Technologies, vol. 2351. SPIE, San Jose (1994)
7. Milgram, P., Kishino, F.: A taxonomy of mixed reality visual displays. IEICE Transactions On Information And Systems E Series D 77, 1321–1321 (1994)
8. SourceForge: Mixed reality (2010), `http://mixedreality.ostfalia.de/index.php/topic,73.0.html` (retrieved January 28, 2010)
9. da Silva Guerra, R., Boedecker, J., Mayer, N., Yanagimachi, S., Hirosawa, Y., Yoshikawa, K., Namekawa, M., Asada, M.: Introducing physical visualization sub-league. In: Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F. (eds.) RoboCup 2007: Robot Soccer World Cup XI. LNCS (LNAI), vol. 5001, pp. 496–503. Springer, Heidelberg (2008)
10. da Silva Guerra, R., Boedecker, J., Mayer, N., Yanagimachi, S., Hirosawa, Y., Yoshikawa, K., Namekawa, M., Asada, M.: Citizen eco-be! league: bringing new flexibility for research and education to robocup. In: Proceedings of the Meeting of Special Interest Group on AI Challenges, vol. 23, pp. 13–18 (2006)

11. da Silva Guerra, R., Boedecker, J., Mayer, N., Yanagimachi, S., Ishiguro, H., Asada, M.: A New Minirobotics System for Teaching and Researching Agent-based Programming. In: Computers and Advanced Technology in Education, Beijing (2007)
12. Michel, O.: WebotsTM: Professional mobile robot simulation. Arxiv preprint cs/0412052 (2004)
13. Obst, O., Rollmann, M.: Spark-a generic simulator for physical multi-agent simulations. Computer Systems Science and Engineering 20(5), 347 (2005)
14. Smith, R., et al.: Open Dynamics Engine. Computer Software (2010), `http://www.ode.org` (retrieved January 28, 2010)
15. Noda, I., Noda, C., Matsubara, H., Hiraki, K., Frank, I.: Soccer server: A tool for research on multiagent systems. Applied Artificial Intelligence (1998)
16. Pedrosa, D.P.F., Yamamoto, M.M., Alsina, P.J., de Medeiros, A.A.D.: Um simulador dinamico para mini-robos moveis com modelagem de colisoes. In: VI Simposio Brasileiro de Automação Inteligente (Setembro (2003)
17. Simões, M., da S Filho, J., Cerqueira Jr. A., Reichow, J., Pimentel, F., Casaes, E.: BahiaMR 2008: Descrição do Time (2008)
18. Scilab: Scilab home page (2010), `http://www.scilab.org/` (retrieved January 28, 2010)