# NaOISIS*: A 3-D Behavioural Simulator for the NAO Humanoid Robot

Aris Valtazanos and Subramanian Ramamoorthy

School of Informatics
University of Edinburgh
Edinburgh EH8 9AB, United Kingdom
a.valtazanos@sms.ed.ac.uk, s.ramamoorthy@ed.ac.uk

**Abstract.** We present NaOISIS, a three-dimensional behavioural simulator for the NAO humanoid robot, aimed at designing and testing physically plausible strategic behaviours for multi-agent soccer teams. NaOISIS brings together features from both physical three-dimensional simulators that model robot dynamics and interactions, and two-dimensional environments that are used to design sophisticated team coordination strategies, which are however difficult to implement in practice. To this end, the focus of our design has been on the accurate modeling of the simulated agents' perceptual limitations and their compatibility with the corresponding capabilities of the real NAO robot. The simulator features presented in this paper suggest that NaOISIS can be used as a rapid prototyping tool for implementing behavioural algorithms for the NAO, and testing them in the context of matches between simulated agents.

**Keywords:** Robot Simulators, Robotic Soccer, Strategic Behaviours

## 1 Introduction

Software simulators are essential testing and debugging tools for large-scale robotics applications, such as those encountered in the RoboCup domain. The establishment of the NAO humanoid robot as the hardware platform for the Standard Platform League (SPL) has been followed by the development of several such environments, each focusing on different aspects of the complex robotic soccer problem. Most simulators can be classified in one of two extreme categories; on the one hand are the implementations that attempt to generate a full, three-dimensional, physical representation of the robot and its dynamics, aimed at improving low-level sensorimotor skills. On the other hand lie the two-dimensional, kinematics-only simulators, which are normally used in the development of sophisticated strategic algorithms for multi-agent systems.

While the merit of each of these two extremes is indisputable, there has been little work in developing intermediate-type, three-dimensional behavioural simulators, which would bring together features from both categories. In broad

* NaOISIS is derived from "NAO" and the ancient Greek word "Noisis" (***Νόησις***), which means cognition.

terms, a 3-D behavioural implementation would abstract away the complex, low-level dynamics found in a physical simulator, while still restricting robots to a realistic, partially observable, perspective view of their environment that impacts their decision-making capabilities - the latter feature cannot be modeled accurately in two dimensions.

In this paper, we present NaOISIS, a three-dimensional behavioural simulator for the NAO humanoid robot developed in the MATLAB environment. As outlined above, the simulator builds on the notion of decoupling *motion dynamics* from the implementation of behavioural algorithms, while maintaining basic physical features and properties (such as collisions between the robots, the ball and the goalposts, and kicks of varying speed and directions). Furthermore, our implementation focuses on the accurate modeling of NAO's *sensing* capabilities, by providing simulated cameras and sonar sensors that closely match the real robot's hardware specifications. Thus, it is possible to recreate a realistic *perception-action* cycle, where the simulated agents must make decisions facing the same perceptual constraints as a physical NAO robot.

In the context of developing physically realisable strategic behaviours, one may argue that the abstraction of low-level motion dynamics is too strong a concession to make. However, we claim that this may not necessarily be the case for two reasons.

On the one hand, any experienced physical simulator user can testify that it is extremely difficult to achieve a reliable one-to-one correspondence between simulated motions and motions in the real world - any complex, non-trivial motion must undergo significant transformations before being ported from a simulator to the robot, and vice versa. This complication is magnified when considering closed-loop motions (e.g. walking), where the interplay between the robot's sensors and the environment becomes even harder to simulate accurately.

On the other hand, this abstraction is supported by recent advances in the SPL community, which have seen the development of robust walking engines for the NAO robot; because of the standard platform nature of the competition, it is possible to adopt these engines directly. For example, the latest version of the Aldebaran NAO SDK [7] features a closed-loop implementation that accepts commands of the form $(dx, dy, d\theta)$ and plans a path to move the robot to this desired location. Through such interfaces, the decoupling of motion dynamics from decision making and path planning arises naturally, so it is possible to focus on strategic interactions instead.

In light of the above considerations, our proposed NaOISIS simulator serves as a useful middle ground for developing new behavioural algorithms for the NAO humanoid robot. The selection of MATLAB as a programming environment facilitates experimentation with several sophisticated machine learning, optimal control, reinforcement learning and path planning techniques, implementations of which are not available or are difficult to implement in other programming languages. This choice does come at the expense of computational speed (compared to other candidates such as C or C++), although this is not too restrictive. In the following sections of this paper, we first briefly review other simulators available for the NAO, and we then present the salient features of NaOISIS.

Finally, we illustrate how NaOISIS can be used as both a debugging tool and a simulator for full soccer matches, by presenting representative screen shots from sample runs.

## 2 Related Work

SimSpark [3] is a general-purpose, open-source simulator, designed to accommodate a variety of robotic systems. A specific implementation tailored to the NAO robots was adopted as the official simulator for the RoboCup 3-D Simulation League [11] in 2009. SimSpark features a physical model of the NAO humanoid dynamics, while also modeling the limited field of view of the robot and its perspective view of the environment. The simulator architecture is generic enough so that different behavioural frameworks can plug-in and interact in the context of a simulated match. However, despite being adequate for a simulated match, the underlying dynamics do not yet faithfully resemble the dynamics of a physical robot, as can be seen in videos from recent competitions [8][9]. Other examples of general-purpose simulators used in the RoboCup domain include USARSim [6], which has been mostly used in the context of the Rescue League [12].

A similar NAO implementation exists for the popular proprietary Webots environment [2], providing similar features as SimSpark. The Webots simulator provides an interface for developing controllers in various programming languages (C, C++, Java, MATLAB). A controller plugging directly to the Aldebaran NAO SDK (NaoQi) has also been developed, allowing the sharing of function calls. However, similar problems with the correspondence between real and simulated dynamics exist in this simulator.

B-Human [5], the 2009 and 2010 winners of the RoboCup Standard Platform League, have also released a custom-made simulation environment. This simulator forms part of their software development kit (available for download online) and is therefore mostly suited to their own custom cognitive architecture.

Perhaps the most widely used simulator in the RoboCup domain is the official simulator of the 2-D Simulation League [4]. Although not designed specifically for the NAO robot, it can be used to simulate more complex, multi-agent strategic algorithms, a subset of which could be potentially implemented on real humanoids. However, the lack of a realistic sensor model makes it difficult to achieve a close correspondence between real and simulated strategies.

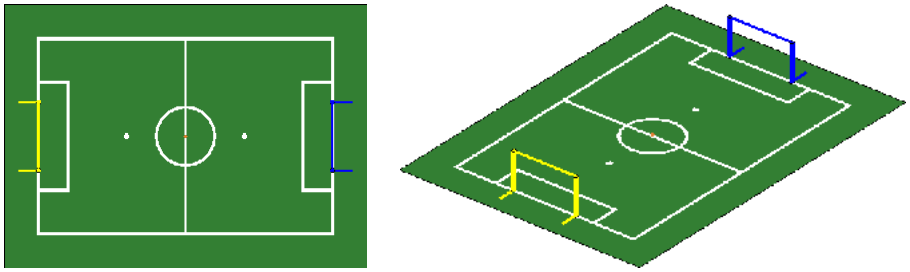
## 3 The NaOISIS Simulator

In this section we present NaOISIS, a 3-D kinematic simulator aimed at creating strategic behaviours for robot soccer teams. NaOISIS is written in MATLAB, and is therefore suitable for use with a number of machine learning, path planning, reinforcement learning and optimal control toolboxes. The simulator can also serve as a debugging tool for *state estimation* algorithms, by visualising and

comparing the robots' egocentric beliefs to ground truth data. The NaOISIS source code[1] is available for download from [1].

### 3.1 Environment and Sensors

**Field and Robots** The main window of the NaOISIS simulator visualises the soccer field where the robots interact (Figure 1). The dimensions of the field are the same as those used of the RoboCup pitches (6x4m), with the goal posts, central circle, penalty boxes, and penalty kick cross spots similarly scaled (see [10] for a detailed specification). The ball is an orange sphere of a 3cm radius. As in RoboCup, the two goal mouths are colour-coded yellow and blue to assist in vision-based localisation.



(a) Top-down view. (b) Angular view.

**Fig. 1.** Soccer field.

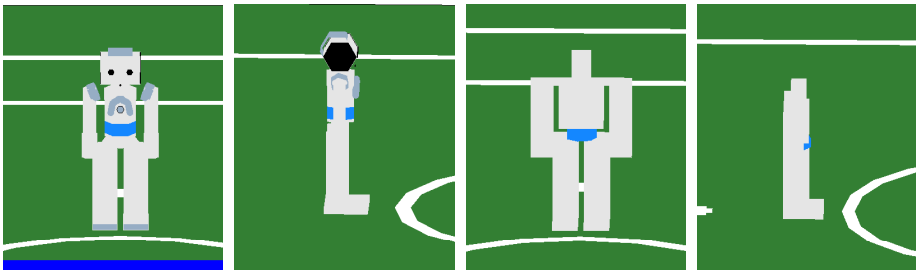
The current version of NaOISIS supports two different robot versions. Figures 2(a)-2(b) shows the design which is visually closer to the real NAOs. Unfortunately, this version takes longer to render[2], so a simpler robot design, approximating the NAO through square patches, is often preferred (Figures 2(c)-2(d)). However, it is worth noting that both versions are scaled to match the dimensions of the real humanoid (see [7] for specification), so the difference between them is mainly aesthetic.

As in the RoboCup competition, robots wear either a blue or a pink waistband, depending on the team they are playing for. The waistbands are also included in both robot designs, and serve as an additional cue for visual robot detection and distance estimation.

**Cameras** Given that our simulator focuses on developing plausible strategic behaviours for humanoids, it is essential to endow the simulated robots with a

[1] NaOISIS is released under a GPL (`http://www.gnu.org/licenses/gpl.html`) license.

[2] We use the built-in OpenGL libraries provided in MATLAB for scene rendering, and the camera toolbox to set the perspective view for the robots.



(a) Simulated NAO robot. (b) Side view of Figure 2(a). (c) Simplified design using square patches. (d) Side view of Figure 2(c).

**Fig. 2.** Soccer field.

**Table 1.** Simulated camera specification.

(a) Camera locations.

| Camera | x(cm) | y (cm) | z(cm) |
|---|---|---|---|
| Bottom | 4.88 | 0.0 | 47.09 |
| Top | 5.39 | 0.0 | 53.72 |

(b) Head angle limits.

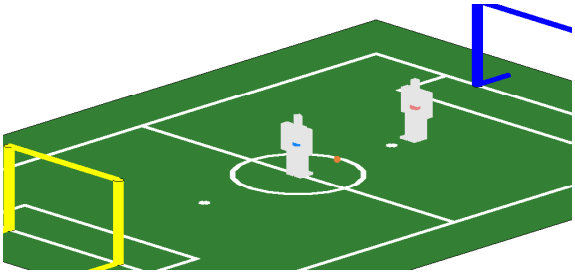
| Rotation type | Minimum | Maximum |
|---|---|---|
| Yaw | $-35°$ | $+30°$ |
| Pitch | $-120°$ | $+120°$ |

realistic sense of their environment. The simulated vision system of NaOISIS was modeled on the cameras of the real NAO robot. Each simulated agent has access to two cameras (Table 1(a)[3]) - a top one, allowing a more complete coverage of the soccer environment, and a bottom one with an additional rotational offset of 40° about the y-axis to allow tracking of nearer objects (e.g. the ball when kicking). Additional constraints have been placed on the head yaw and pitch angles of the robots (Table 1(b)). The cameras have a diagonal of 58°, and return 320x240 images in RGB format.

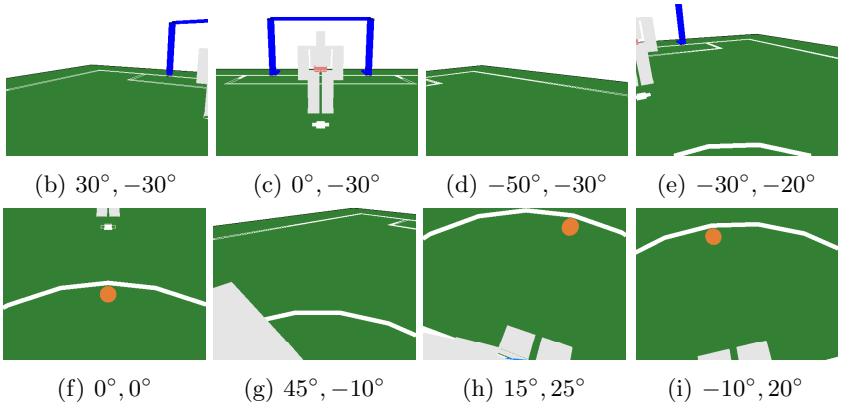
The simulated vision system is illustrated in Figure 3. Figure 3(a) shows an example scene with two robots and a ball. Figures 3(b)-3(i) plot the perspective view as seen by the bottom camera of the robot located on the halfway line, for various head pitch and yaw combinations.

**Sonar sensors** Whereas vision is an important tool for ball detection and localisation, sonar sensing is essential for more interactive tasks such as robot avoidance. NaOISIS features a pair of simulated sonar sensors, also modeled on the corresponding devices of the NAO humanoid. Each of these sensors has a range from 0.15 to 0.75m (these values are modifiable). Moreover, there is an option for adding varying levels of Gaussian - or other - noise to the sensor readings, so as to reflect irregularities that would occur in a physical setting.

[3] The origin is at the midpoint of the two feet, x-axis points to the front, y-axis left, and z-axis up.



(a) Panoramic view of the scene.



(b) $30°, -30°$ (c) $0°, -30°$ (d) $-50°, -30°$ (e) $-30°, -20°$

(f) $0°, 0°$ (g) $45°, -10°$ (h) $15°, 25°$ (i) $-10°, 20°$

**Fig. 3.** Field of view for various head angle combinations (yaw,pitch).

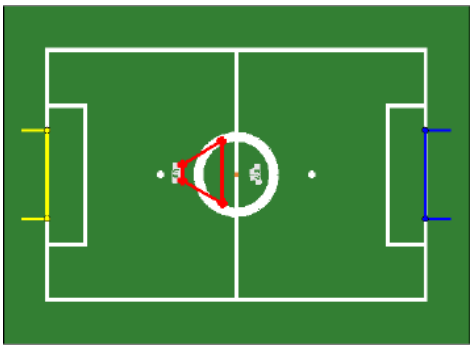
Figure 4 shows a visualisation of the joint range of a robot's sonar sensors, together with the technical specification.

### 3.2 Physical Interactions

Despite not modeling motion dynamics explicitly, NaOISIS offers a coarse simulation of physical interactions between robots and their environment. The focus is on modeling motions that form an essential part of a humanoid robot's strategy, such as translational/rotational movements and kicking actions. Collisions between robots, balls, and goalposts are also considered. The current version of the simulator does not account for more complex movements, such as getting up from a fall or dives for goalkeepers.

**Moving** Each simulated agent has access to a simple motion engine which allows it to navigate around the soccer field. The engine accepts commands of the form:

$$(dx, dy, d\theta) \tag{1}$$



(a) Trapezoidal approximation of the joint range of the left robot's two sonar sensors. Only objects within the range of the trapezium are detected.

| Sensor | Angle | Height | Angular range |
|---|---|---|---|
| Left | $+19.48°$ | 40cm | $18°$ |
| Right | $-19.48°$ | 40cm | $18°$ |

(b) Specification.

**Fig. 4.** Sonar sensor range visualisation and specification.

which correspond to the desired linear and angular displacement with respect to the robot's current coordinate frame. To ensure that the executed commands are plausible, the following are also defined before the simulator is run:

- The maximum linear and angular velocities of each robot
- The spread of the error added to each command (which may vary along the three dimensions of the motion)

Thus, if a robot requests to move by an amount that exceeds its capabilities, this command will be scaled down and/or perturbed accordingly by the simulator.

**Kicking** Kicking actions are more difficult to model than translational and rotational movements because they involve interactions between the robots and the ball. Thus, a coarse approximation to the dynamics of these interactions is required to achieve physically plausible kicks. NaOISIS currently supports four different types of (non-directional) kicks:

- Left/right-footed straight kicks
- Left/right-footed side kicks, where the robot first extends the kicking foot to the front and then performs a side motion.

At the start of the simulation, the *maximum* traveling distance for each kicking type is defined. Then, when a robot wishes to kick the ball, it utilises a command with the following information:

| Kick type | Min x (cm) | Max x (cm) | Min y (cm) | Max y (cm) |
|---|---|---|---|---|
| Left straight | 10 | 20 | 0 | 10 |
| Right straight | 10 | 20 | -10 | 0 |
| Left side | 10 | 22 | -15 | 0 |
| Right side | 10 | 22 | 0 | 15 |

**Table 2.** Admissible rectangles for each kicking type, defined in terms of the robot's coordinate frame (x-axis points to the front, y-axis to the left). For each kicking command, the ball will move only if it lies within the corresponding rectangle. The tips of the left and right feet are at positions (10,5) and (10, -5) respectively.

- The type of the kick
- The desired speed of the kick, as a number between 0.0 and 1.0, where 1.0 corresponds to maximum allowed speed for the desired type of kick.

The precise trajectory of the ball after a kicking command depends on its *position* and *angle* relative to the kicking foot. A ball will be affected by a kick only if it lies within a rectangle defined with respect to the robot's coordinate frame (Table 2). If the ball lies in the appropriate rectangle, the direction of its trajectory will be the angle formed by the line connecting the ball and the tip of the robot's foot (see caption of Table 2), and the line which is normal to the foot on the robot's coordinate frame. Some special kicking cases are also accounted for; for example, if the robot attempts a side kick from the "wrong" side, it will cause a straight kick of reduced speed. Figure 5 shows several examples of kicking commands and the trajectories they incur depending on the ball position.

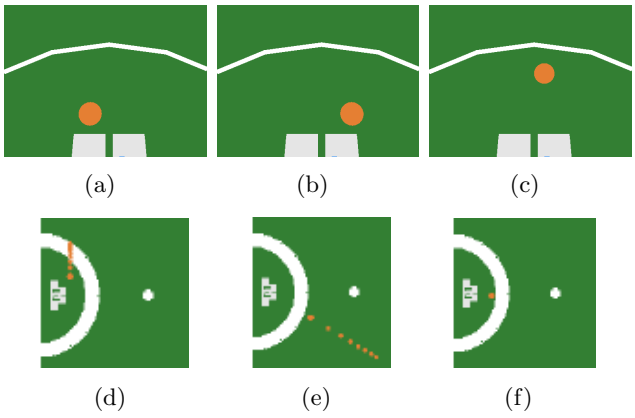
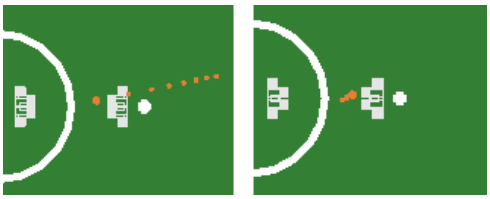



**Fig. 5.** Robot's perspective (top) and resulting ball trajectories (bottom) for various ball positions and kicking types. (a)-(d): right side kick. (b)-(e): right straight kick with the ball positioned at an angle (but within the required rectangle). (c)-(e): right straight kick - the ball is outside the kicking range so the resulting trajectory is null.

**Collision between robots and objects** Collisions between robots and the various objects in the soccer field are also coarsely modeled in the NaOISIS simulator. The following types of collision are taken into account:

- Collision between two moving robots
- Collision between a moving and a static robot
- Collision between a moving robot and a goalpost
- Collision between a moving robot and a moving ball (as the result of a kick)
- (Accidental) collision between a moving robot and a static ball

For each robot, two different sets of rectangular bounds are considered: one for its base and one for its torso. For the first three types of collision, the simulator checks for intersections between torso bounds and/or goalpost cylinders; for the latter two, the torso bounds are replaced with the robot's base bounds. Whenever a ball is involved in a collision, the simulator also computes its adjusted trajectory (Figure 6).



(a) (b)

**Fig. 6.** Ball trajectory before (a) and after (b) collision with robot.

### 3.3 Information Exchange

Agents are also provided with a simple message-passing capability, which allows them to communicate messages to their teammates. As NaOISIS is a discrete-step simulator, message exchange occurs once every decision cycle. Each robot updates a *shared data structure* with a list of relevant data, such as its self-localisation belief and the observed position of the ball and/or the other robots. The robots then communicate this structure to their teammates, who may use it to adjust their decisions or to form cooperative strategies. As with physical interactions, there is an option to add a non-zero probability of messages not being delivered successfully.

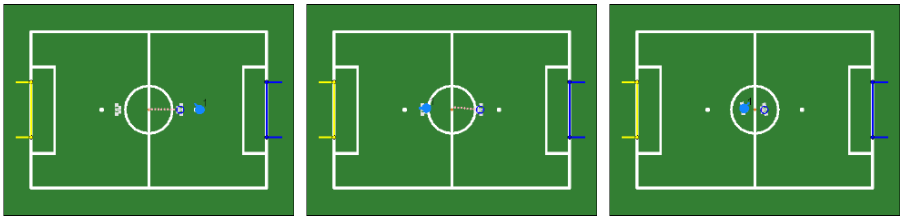
### 3.4 Experiments and Testing

The simulator components described in the previous sections can be used to design and test physically plausible behaviours and algorithms. Depending on the user's requirements, NaOISIS may be used to either debug such behaviours and test their properties under varying conditions, or to simulate full soccer matches between teams of autonomous agents.

**Developing and debugging algorithms** NaOISIS provides experimental support for a wide range of algorithmic domains, such as:

- Autonomous decision making under uncertainty
- Path planning and dynamic obstacle avoidance
- Reinforcement learning
- Belief estimation, information filtering and sensor fusion
- Vision-based localisation
- Multi-agent coordination

In most of these domains, it is important to be able to compare the agent's egocentric beliefs and estimates to ground-truth data. NaOISIS provides a set of debugging tools that can be used to visualise this comparison. Figure 7 provides a simple illustration of this functionality in the context of observation-based estimation of the opponent's location.



(a) Initial belief - no information available.

(b) Revised belief based on visual estimate.

(c) Further revision using sonar estimate.

**Fig. 7.** Egocentric beliefs of the right-hand side robot on the location of its opponent. The belief at each stage is indicated by the blue circle.

**Simulating Full Matches** The soccer match mode of NaOISIS provides additional functionalities that help simulate full soccer matches between teams of autonomous agents. Most of these are concerned with the position of the ball at the various stages of the game, so as to determine if a goal has been scored or if the ball has crossed the field bounds. In each of these cases, the ball and the robots are repositioned and the score is updated accordingly.

The collision modeling functionality discussed in Section 3.2 is used to determine whether a robot should be penalised for a certain period of time. Whenever a robot is deemed "responsible" for a collision by the simulator, it is placed on the sideline for an interval of time before it is allowed to re-enter the game. Responsibility is judged based on the relative speeds of the colliding robots, and the location of the ball. Furthermore, if a robot does not move for a long period of time while being close to the ball (and thus preventing other robots from reaching it), it is also penalised. Since robot falls are not modeled in the current version of the simulator, penalty shootouts would be highly disfavourable to

goalkeepers, and as such they have also been omitted. This will be implemented as an extension in future versions of NaOISIS.

The debug and match modes of NaOISIS are not mutually exclusive; it is possible to plot robot beliefs and other relevant information in the match version as well. Moreover, since the match is carried out at discrete perception-decision-action states, it is possible to record snapshots of the soccer field at various stages. These snapshots can then be synthesised as a movie or used otherwise for evaluation. The frames from the robots' camera feeds can also be recorded and used to debug information processing and state estimation algorithms. Figure 8 shows screenshots from a soccer match betwen two robots.

## 4 Conclusion

We have presented NaOISIS, a three-dimensional behavioural simulator for the NAO humanoid robot. NaOISIS differs from traditional physics-based simulators in abstracting away most of the low-level dynamics of the humanoid, while still modeling salient physical interactions such as kicking actions and inter-robot collisions. Thus, the simulator supports the design of more realistic strategies than it is possible with simpler, two-dimensional environments. The current version of NaOISIS can be used to both debug behavioural algorithms in a rapid prototyping environment, and test them in soccer matches between simulated agents; moreover, it can be integrated with several machine learning, path planning and reinforcement learning that are available in MATLAB. Future versions of NaOISIS will concentrate on the coarse modeling of further physical interactions, such as robot falls and dives, while also providing a more standardised environment for implementing decision making algorithms.

## References

1. NaOISIS Source Code, http://homepages.inf.ed.ac.uk/s0566900/naoisis.html
2. NaoQi for Webots, http://www.cyberbotics.com/nao/
3. SimSpark simulator, http://simspark.sourceforge.net/
4. The RoboCup 2D Soccer Simulator, http://sourceforge.net/projects/sserver/
5. B-Human Standard Platform League Team, http://www.b-human.de/en/
6. USARSim, http://sourceforge.net/projects/usarsim/
7. NAO v1.10 User's Guide, http://academics.aldebaran-robotics.com/
8. RoboCup 2010 3D Simulation League - Semifinal second extra half, http://www.youtube.com/watch?v=MNCObp1TvNE
9. RoboCup 2010 3D Simulation League - Final first half, http://www.youtube.com/watch?v=EmMt_mlpvWE
10. RoboCup Standard Platform League Official Rule Book, http://www.tzi.de/spl/pub/Website/Downloads/Rules2010.pdf
11. The RoboCup Soccer Simulation League, http://wiki.robocup.org/wiki/Soccer_Simulation_League
12. The RoboCup Rescue League, http://www.robocuprescue.org/

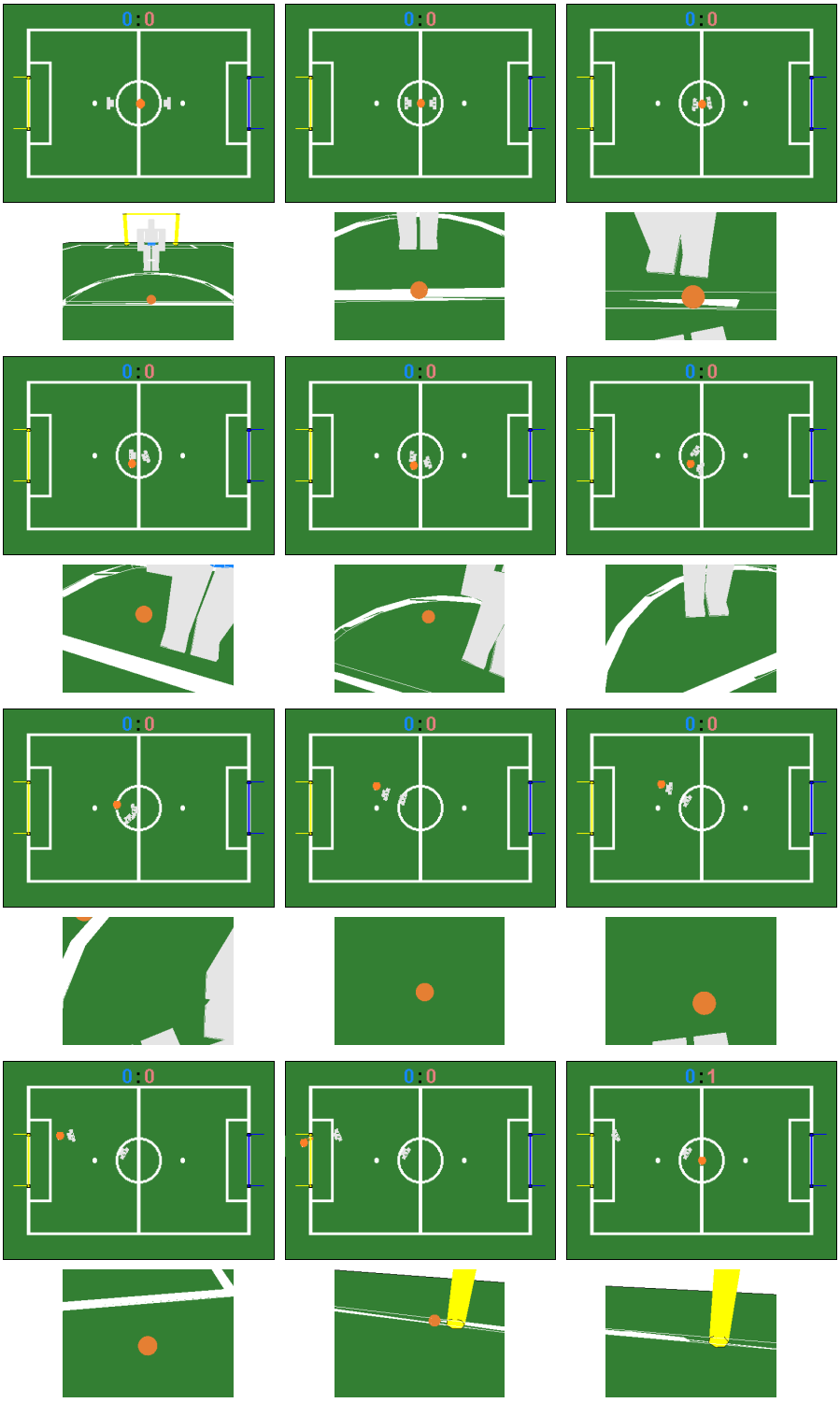



**Fig. 8.** Simulated soccer game. Top to bottom, left to right - *Odd rows:* field screenshots showing the positions of the robots and the ball at various stages. *Even rows:* the corresponding field of view of the pink robot (initially on the right hand side).