AutoRef: Towards Real-Robot Soccer Complete Automated Refereeing

Danny Zhu, Joydeep Biswas, and Manuela Veloso

School of Computer Science, Carnegie Mellon University,
Pittsburgh, Pennsylvania 15213, USA
{dannyz,joydeepb,veloso}@cs.cmu.edu

Abstract. Preparing for robot soccer competitions by empirically evaluating different possible game strategies has been rather limited in leagues using real robots. Such limitation comes from factors related to the difficulty of extensively experimenting with games with real robots, such as their inevitable wear and tear and their usual limited number. RoboCup real robot teams have therefore developed simulation environments to enable experimentation. However, in order to run complete games in such simulation environments, an automated referee is needed. In this paper, we present AutoRef, as a contribution towards a complete automated referee for the RoboCup Small-Size League (SSL). We have developed and used AutoRef in an SSL simulation to run full games to evaluate different strategies, as we illustrate and show results. AutoRef is designed as a finite-state machine that transitions between the states of the game being either on or required to stop. AutoRef purposefully only uses the same visual and game information provided in SSL games with physical robots, which it uses to compute the features needed by the rules and to make decisions to transition between its states. Due to this real input to AutoRef, we have partially applied it to games of the physical robots. As AutoRef does not include all the rules of the real SSL games, we currently view it as an aid to human referees of SSL games, and discuss the challenges in automating several specific SSL game rules. AutoRef could be extended to other RoboCup real soccer leagues if a combined view of the game field, ball, and players is available.

1 Introduction

To date, the creation and evaluation of strategies in RoboCup SSL has tended to be rather *ad hoc*: teams work on their code throughout the year; when the tournament comes, they play a few games against other teams, and try to draw conclusions based on that. An issue with this is that teams are very limited by the small amount of time during which they can actually see the performance of their strategies. This is, in large part, due to the need for refereeing when games are played. A human must be present to set the game state and place the ball when it leaves the field, or else robots are unable to play games autonomously.

Apart from this need, the difficulty of maintaining a large number of real robots in the face of the normal wear and tear of the game is another significant

limiting factor to adequate testing. Many teams have developed simulation environments to overcome this limitation, but such environments still require the full-time attention of a human to be fully useful.

In this paper, we focus on an automated referee, AutoRef, for the SSL. It is capable of working with either real robots using the standard SSL vision system [1], or with simulated robots using our team's simulation system. Using AutoRef, we have run many full games in simulation to demonstrate the evaluation of different strategies.

The core of AutoRef is a finite state machine that tracks the state of the game and determines whether play should be stopped or restarted at each moment. For the most part, it takes the same input whether it is running with real or simulated robots (the exception being the height of the ball, which is currently difficult to reliably obtain outside of simulation). Based on this input, AutoRef evaluates the rules of the game to determine how to advance at each timestep. Because of the great similarity between the real and simulated inputs to AutoRef, we are able to partially apply it to real games.

There are still two core challenges for an automated referee for a real robot league, preventing a full application. First, the ball needs to be positioned by the referee at specific positions after each stoppage of the game. Second, all of the events of the game, particularly those centered on ball motion, need to be checked for compliance with the rules based on real perception. For now, we do not address the first challenge, and rely on a human to position the ball on the field. (However, we do envision eventually having a mobile robot capable of handling the ball, as any other real robot soccer player, to act as the ball positioner.) In this work, we address the second challenge, but we do not yet implement the detection of all the rules of the game, as some are particularly challenging to identify from a perception point of view.

With either simulated or real robots, automated referees have several advantages over human ones. It can be difficult for a human referee to keep track of the fast-moving robots and ball, whereas this is not a problem for an automated referee. Similarly, an automated referee could enforce complicated rules such as the offside rule, which has not been adopted in the SSL precisely because of the burden it would place on the human referee.

2 Related work

Automated online processing of RoboCup games has been addressed in several contexts. Several teams have discussed automated refereeing for simulations: Rocco [2], an early simulation commentator, uses a declarative event database to transform the continuous inputs from the game into the discrete events on which to comment. Röfer et al. [3] briefly discuss a simulator-based referee for the Standard Platform League (SPL), which behaves similarly to AutoRef in simulation, but the rule checking and state detection should be simpler due to the slower pace of the game.

In terms of observing real games, RFC Stuttgart [4] presented an automated cameramanthat could take a video of a complete game of the Middle Size League (MSL). The cameraman uses shared information from a team to keep an on-board pan-tilt unit pointed toward the ball or other object of interest at all times. Merely following the ball involves limited interpretation of the game, while commentating involves some understanding of the elements of a game. Commentating is related to refereeing in terms of the real-time observation and analysis of games, though the decisions that need to be made based on the observations are different. Veloso et al. have previously created CMCast [5], a pair of humanoid robots (SONY QRIOs) that commentated a game of the RoboCup four-legged league with the SONY AIBO robots. The two CMCast robots are able to autonomously move along the field, localize themselves, and visually track objects and detect predefined events, which they then announced with a rich set of spoken utterancess. Tanaka-Ishii et al. presented Mike, a commentator for the simulation league [6], which they later joined with the torso of a humanoid robot to commentate on SSL games [7]. The commentators and referees face similar event detection challenges, but differ in the referee's physical embodiment and the commentator's desired interaction with spectators.

Two other projects to which this work is particularly comparable are the open-source `ssl-autonomous-refbox` [1], which provides rule checking based on perception during an SSL game, and an automated referee presented by Tech United Eindhoven for the MSL [8]. The work discussed here goes further than the former by broadcasting appropriate commands to the competing teams in response to its observations, and further than the latter, by incorporating more rules and, at least in simulation, being able to run a game from start to finish.

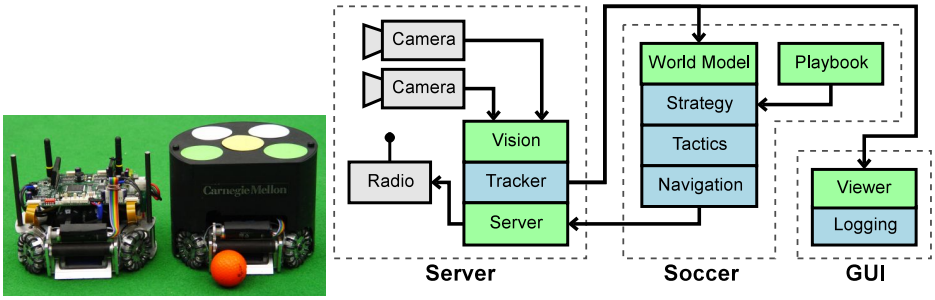
3 SSL: a CMDragons perspective

The RoboCup SSL promotes research in multi-agent coordination in real-world adversarial domains. In this league, teams of six robots play soccer with a golf ball in a field of size $6.05\,\mathrm{m} \times 4.05\,\mathrm{m}$. The robots are constrained to each be contained entirely within a cylinder of diameter 180 mm and height 150 mm. The robots are controlled via radio commands sent by its team's offboard computer. Overhead cameras observe the field, one for each half, and the vision data is processed on a neutral computer by the SSL Vision System (SSL-Vision) [1] based on the early successful CMVision [9]. SSL-Vision broadcasts the detected locations and orientations of all the robots and the ball on the field to the team computers via a wired Ethernet network. With the global vision and centralized planning, the teams focus on the hardware development and coordination behaviors and high-level teamwork strategy. The SSL games are fast-paced, with the robots travelling up to $3\,\mathrm{m/s}$ and kicking the ball at speeds of up to $8\,\mathrm{m/s}$ (enforced by the rules). To illustrate the elements of an SSL team, we present our CMDragons [10], also built upon research used to create the previous CMUnited

[1] `https://code.google.com/p/ssl-autonomous-refbox`

(1997-1999) and CMDragons teams (2001-2003,2006-2010,2013) and CMRoboDragons joint team (2004-2005).

Robot hardware Our team consists of six homogeneous robots (designed and built by M. Licitra in 2006, and pursued and maintained by J. Biswas since 2010). Figure 1(a) shows an example robot with its omnidirectional drive with four custom-built wheels. The main kicker is a solenoid attached directly to a kicking plate; the chip-kicking device is implemented by a flat solenoid located under the main kicker, which strikes an angled wedge to kick the ball at an angle. Ball catching and handling are performed by a motorized rubber-coated dribbling bar [11].



(a) (b)

Fig. 1: (a) A CMDragons robot shown with and without a protective cover; (b) The general architecture of the CMDragons offboard control.

Robot behavior architecture overview Figure 1(b) shows the behavior architecture for our offboard control. The major architectural components are a server system that performs vision and manages communication with the robots, and two client programs, namely (i) a soccer program, which implements the individual and team behavior strategy and robot navigation and control, and (ii) a graphical interface program for monitoring, replaying, and testing the team. The server acts as the central translation point between our team clients and the external systems, namely SSL-Vision and the human-operated referee box.

Simulator We use a high-fidelity 3D soccer simulator [12] based on the NVIDIA PhysX engine [2]. Our simulator incorporates the exact geometric shapes of the robots for accurate collision modelling. Kicking and dribbling are simulated by directly imparting impulse linear or angular momentum to the ball. All The values of the impulse momenta are calibrated to match the speed of the kicked ball in the real world.

The simulator is a drop-in replacement for the soccer server. It accepts standard robot control messages from clients and sends vision messages back to

[2] `https://developer.nvidia.com/physx`

them; however, instead of generating the vision messages based on actual cameras, it reports the positions of robots in the physical simulation [13], accounting for a simulated communications delay approximately matching that encountered when using real robots. The positions of the robots are drawn with the extra predicted information. The the recent speed or acceleration of the ball or any robot, and a text log from the soccer The viewer displays a very extensive set of information, which can be easily selected with multiple levels of detail.

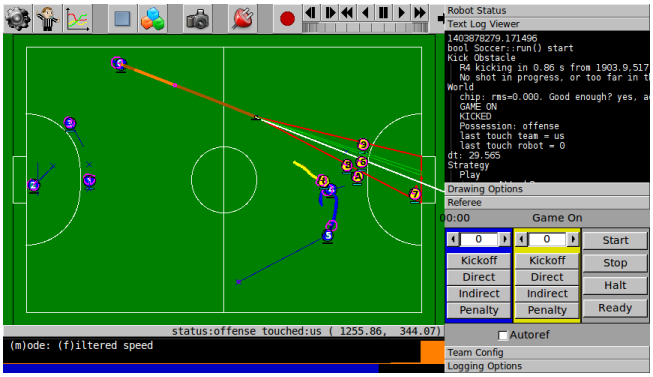


Fig. 2: A snapshot of our sophisticated graphical visualization program used for simulation, monitoring the real robots, and replaying logged game data.

· Over the several years that CMDragons has been competing, we have developed quite a complex set of algorithms for playing soccer and evaluating performance, and as a result it is important to be able to reduce the amount of effort and attention required to run test games with our robots; an automated referee, as we contributed in this work, provides a valuable step in this direction.

4 AutoRef

We describe the implementation details of AutoRef. This includes the means by which it communicates with the rest of the CMDragons codebase, the types of rules present in the game, and the internal details of how it enforces those rules.

4.1 AutoRef interface with the game

AutoRef executes as a standalone program; it operates based solely on vision information received from our main server, and sends commands to the server in the same format as commands from the normal human-operated referee box.

In particular, the design of our system means that, at least on the protocol level, AutoRef can be used nearly transparently with either the physical robots or the simulation, and indeed we have used it in this fashion. Currently, it can act as a sort of assistant to a human referee: it automatically signals a stoppage

when the ball has gone off the field, indicating which team was the last to touch the ball. There are practical difficulties with using physical robots; for example, since SSL-Vision provides only locations of the ball and robots, not raw video feeds, it is impossible for teams to tell when the human referee has finished placing the ball and stepped off the field, so AutoRef waits for the human's signal to restart play. In simulation, it can simply signal to the simulator that the ball should be moved to a particular location; the simulator can then drag the ball smoothly to the desired location.

4.2 Types of game rules

The rules of the SSL fall into a few main categories, some of which are easier than others to handle with AutoRef.

Time-related rules The conceptually simplest rules are those relating to game time. Each half of the game lasts ten minutes; AutoRef keeps track of the elapsed time in the game and ends the current half when appropriate. Time starts counting when the ready command is sent to teams at the beginning of a half, and each half is ten minutes long. At the end of the second half, AutoRef can either signal for a new game to begin or simply leave the robots halted. Timeouts are not currently handled.

Ball-related rules The ball-related rules are primarily those that call for a game stoppage when the ball goes out of bounds.

- When the ball enters one of the goals, a goal is awarded to the appropriate team, the ball is placed in the center of the field, and a kickoff is signaled.
- When the ball leaves the field without entering a goal, a free kick is awarded to the team which touched the ball less recently. If the ball passes over a touch boundary, a throw-in is awarded. If the ball passes over a goal boundary, a corner kick is awarded if the defending team touched the ball more recently, or a goal kick otherwise. In each of these cases, the ball is reset to a particular position near the boundary of the field: throw-ins are taken from a point 100 mm from the point where the ball exited the field, while corner kicks and goal kicks are taken 100 mm from the nearest touch boundary and 100 mm or 500 mm, respectively, from the goal boundary that was crossed.

Either in simulation or with physical robots, AutoRef can track the ball position and signal a stop when the ball leaves the field. In simulation, it can also restart gameplay at an appropriate time (once all robots have stopped moving, with a fixed timeout to avoid waiting forever).

Other ball-related rules are that an indirect free kick is awarded to the opposing team if a robot either kicks the ball faster than $8m/s$ or kicks it into the air and directly into the goal. The former has not currently been implemented in AutoRef, but it would be easy to do so, given that the server provides filtered velocity estimates of the ball at each frame. The latter is more problematic, owing to the difficulty of implementing a robust chip kick detector.

Robot-related rules Each robot must be at least 200 mm away from the opposing team's defense area, or an indirect free kick is awarded to the opposing team. AutoRef handles this rule.

A robot which touches the ball while partially or totally in its own team's defense area triggers a yellow card or penalty kick, respectively; this rule is not currently implemented but could be easily added.

A team may also be shown yellow or red cards for a variety of other offenses, such as if it "is guilty of unsporting behaviour," "is guilty of serious and violent contact," "persistently infringes the Laws of the Game," or several others [14]. These conditions are all somewhat subjective, so they are not handled.

4.3 AutoRef input and state machine

Once per camera frame, AutoRef receives as input the values in table 1.

name	description
t	the current time
n	the number of robots on the field
$r_x^{(i)}$, $r_y^{(i)}$	x- and y-coordinates of the i^{th} robot
b_x, b_y	x- and y-coordinates of the ball
c	the last team to touch the ball

Table 1: The values used as input by AutoRef. The time is given in seconds since the Unix epoch. All locations are given in millimeters, in a coordinate system with its origin at the center of the field, positive x-axis pointing toward a goal, and positive y-axis pointing 90° counterclockwise of the positive x-axis.

We devised a finite state machine with transitions governed by the input values, which allows the AutoRef to keep track of the current state of the game as described by the game rules. The machine updates once for each vision frame received from SSL-Vision. The two primary states are as follows.

- `RUN`: This is the state that occurs while normal gameplay is happening. In this state, AutoRef observes the state of the game until a condition is met that means that play should be stopped.
- `WAIT_STOP`: In this state, AutoRef waits for all robots to stop moving. This usually occurs after AutoRef has sent a "stop" or similar command to the server, to allow the robots to reach their final positions before another command is sent.

The other states, which occur less frequently, are as follows.

- `INIT`: This state occurs only once, immediately after execution begins. AutoRef simply saves the current positions of the robots and goes to `WAIT_START`.
- `WAIT_START`: This state occurs only near the beginning of execution, when waiting for clients to connect to the server. While in this state, AutoRef

periodically compares the positions of the robots to the saved positions; once all robots have moved, it signals a kickoff and enters `WAIT_STOP`.
– `DELAY_WAIT`: This state only occurs for one frame at a time; it occurs when AutoRef needs to set the the game state to a value which is only transmitted briefly. This includes the command indicating that a goal has been scored, as well as the command to indicate that a half is starting.

See Figure 3 for a graphical depiction of the state machine. Each transition is labeled with its preconditions (above the line) and the assignments that occur when it is taken (below the line). An empty condition indicates that the transition is always taken one frame after entering the source state. An assignmnent to `refstate` indicates the sending of a referee command to the server; `next` is an auxiliary variable within AutoRef. Not depicted is the handling of game time, which operates outside this diagram, since the end of a half can occur at any point relative to these states.

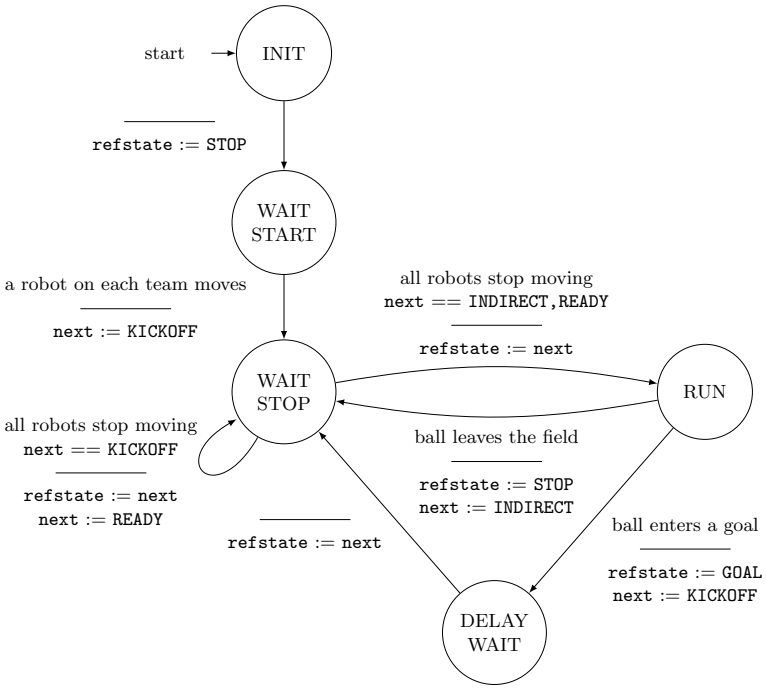


Fig. 3: The finite state machine describing AutoRef.

4.4 Calculations

AutoRef performs several kinds of numerical calculations in order to keep a game running. Primarily, it estimates when all robots on the field are ready for play

to resume, by noting when they have all come more or less to a stop, and it computes not only when but where the ball exits the field, so that AutoRef can properly replace the ball afterward.

Robot positions When entering the `WAIT_STOP` state, AutoRef saves the current positions of the robots $\left\langle r_x^{(1)}, r_y^{(1)}, \cdots, r_x^{(n)}, r_y^{(n)} \right\rangle$. While remaining in that state, it periodically compares its saved values against the most recently received values $\left\langle r_y^{(1)'}, r_y^{(1)'}, \cdots, r_y^{(n)'}, r_y^{(n)'} \right\rangle$ to check whether all the robots have nearly stopped moving; that is, whether, for a small distance ϵ and every i,

$$\sqrt{\left(r_x^{(i)} - r_y^{(i)'}\right)^2 + \left(r_y^{(i)} - r_y^{(i)'}\right)^2} < \epsilon.$$

Currently, ϵ is set to 15 mm, but that choice is somewhat arbitrary. If some i violates this condition, then AutoRef judges that the robots have not come to a stop yet and saves the new positions for later comparison. If all robots have come to a stop, AutoRef transitions out of `WAIT_STOP`.

AutoRef behaves similarly in `WAIT_START`, except that it the condition is inverted: it leaves the state when at least one robot on each team has moved a distance greater than ϵ.

Ball position When the ball goes outside the field, AutoRef needs to determine whether it has entered a goal, crossed a goal boundary without entering a goal, or crossed a touch boundary.

Define the following values:

- F_x: the shortest distance from the center of the field to the outside edge of a goal boundary
- F_y: the shortest distance from the center of the field to the outside edge of a touch boundary
- b_r: the radius of the ball
- G_w: half of the width of the opening of the goal
- G_h: the height of the opening of the goal
- b'_x and b'_y: the previous coordinates of the ball

In order to distinguish the two boundaries when the ball is near a corner, it is necessary to compare the current and last positions of the ball to determine where it first was located entirely outside the field. This is the geometrical problem of finding the intersection between one line which is parallel to an axis and another line determined by two points. The intersection of the line between the points (b_x, b_y) and (b'_x, b'_y) with the line defined by $x = l_x$ is given by (l_x, c_y), where

$$c_y = \frac{b'_y(l_x - b_x) + b_y(b'_x - l_x)}{b'_x - b_x},$$

and a similar equation applies to a line which is defined by a constant y value. (This expression can be derived by finding the equation of the line defined by the two points and substituting $x = l_x$.)

AutoRef judges that the ball is out of the field if $b_x > b_r + F_x$ or $b_y > b_r + F_y$. In fact, this definition ignores a small region at each corner, but the difference is very small.

When the ball leaves the field, in order to determine how to respond, AutoRef computes the intersection points of the ball trajectory with the field boundaries as follows. Set $l_x = (F_x + b_r) \cdot \mathrm{sgn}(b_x)$ and $l_y = (F_y + b_r) \cdot \mathrm{sgn}(b_y)$; these represent the coordinates of the goal and touch boundaries closest to the ball, with the addition of b_r in order to capture the fact that the ball's coordinates describe the center of the ball, while the entirety of the ball must be past the lines to be counted as off the field.

Let c_x be the x-coordinate of the intersection of the trajectory with the line $y = l_y$, and let c_y be the y-coordinate of the intersection of the trajectory with the line $x = l_x$. If $|c_x| < |l_x|$, then the ball must have crossed a touch boundary, so a throw-in is awarded. See Figure 4 for a graphical depiction of the process.

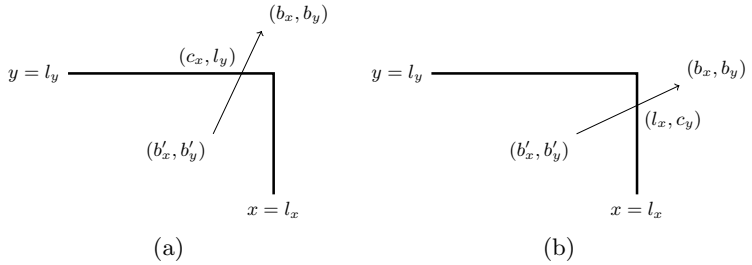


Fig. 4: Illustrations of the computation performed by AutoRef when the ball's trajectory intersects the field boundary. The thick lines depict the boundary of the field, expanded by a distance equal to the radius of the ball, and the arrow depicts the trajectory of the ball between two frames. In (a), since $|c_x| < |l_x|$, the intersection is on a touch boundary. In (b), that condition does not hold, so the intersection is on a goal boundary; the point (c_x, l_y) is farther to the right.

AutoRef judges that the ball is in a goal if the following conditions hold:

$$\begin{aligned} |b_x| &> F_x + b_r, \\ |b_y| &< G_w, \quad \text{and} \\ b_z &< G_h. \end{aligned}$$

These conditions simply specify that the ball is contained entirely inside the volume defined by the walls of the goal.

5 Demonstrations

To demonstrate the collection of statistics, we played repeated games between two instances of our own team code where one instance was handicapped in some way, either by having the acceleration and top speed of its robots reduced by some factor, or by having fewer robots on the field (Figure 5). This resulted in a gap in scores that increased with the magnitude of the handicap. These results are unsurprising, but the collection of data across hundreds of realistically simulated full games is novel, as far as the authors are aware.

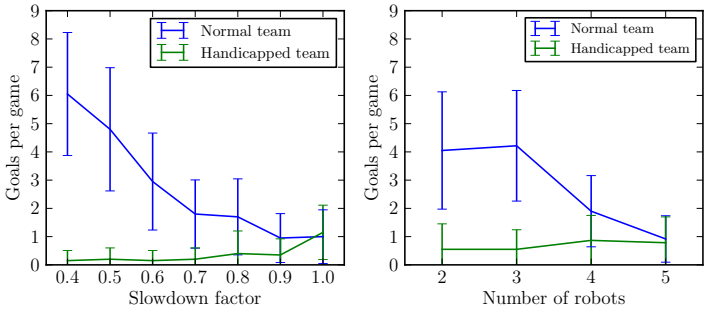


Fig. 5: The mean and standard deviation of the scores of games in which one team had its acceleration and top speed (left; 20 games per value) or number of robots (right; 60 games per value) reduced.

We also tested a limited version of AutoRef with the real robots. With a human referee present to position the ball and restart play after a stoppage, AutoRef observed the game and emitted referee commands as normal. It was able to immediately stop play when the ball left the field, as well as run a kickoff, which involves two transitions in the game state.

There are rules in SSL, which are important but not yet handled by AutoRef, including the conditions for fouls, as well as restrictions on goals scored by chip kicks. Some rules are difficult to implement due to an element of subjectivity by the referee, such as the condition of "deliberately" entering the referee walking area; some because positions alone may not give enough information to determine whether they have ben triggered, such as the rule against "serious and violent conduct"; and some because we lack a reliable chip kick detector. We aim at including all the rules in AutoRef and run it on real games, but until then we envision AutoRef acting in support of human referees, and including a dedicated ball-positioning robot referee.

6 Conclusion

We presented an autonomous referee system for reasoning about rule-related events in SSL games. It is capable of operating with the CMDragons soccer simulator, in which case it can run whole games automatically, and with real

robots and vision, in which case it can support a human referee. This work may eventually be extended to a complete referee for real games; for now, we have created a framework for observing reasoning about high-level game state that may be used to fill either role. We also categorized the main rules of the game according to their impact on an automated referee, and demonstrated preliminary results from recording scores in automatically run games. We believe AutoRef can be extended to other RoboCup real soccer leagues if a combined view of the game field, ball, and players is available.

References

1. Zickler, S., Laue, T., Birbach, O., Wongphati, M., Veloso, M.: SSL-vision: The shared vision system for the RoboCup Small Size League. RoboCup 2009: Robot Soccer World Cup XIII (2010) 425–436
2. Voelz, D., André, E., Herzog, G., Rist, T.: Rocco: A robocup soccer commentator system. In: RoboCup-98: Robot Soccer World Cup II. Springer (1999) 50–60
3. Röfer, T., Laue, T., Müller, J., Graf, C., Böckmann, A., Münder, T.: B-Human Team Description for RoboCup 2012. In: RoboCup 2012: Robot Soccer World Cup XVI Preproceedings, Mexico City, Mexico, RoboCup Federation (2012)
4. Käppeler, U., Zweigle, O., Rajaie, H., Häussermann, K., Tamke, A., Koch, A., Eckstein, B., Aichele, F., DiMarco, D., Berthelot, A., et al.: 1. rfc stuttgart team description 2010. RoboCup Team Description Papers (2010)
5. Veloso, M., Armstrong-Crews, N., Chernova, S., Crawford, E., McMillen, C., Roth, M., Vail, D., Zickler, S.: A team of humanoid game commenters. International Journal of Human Robotics (2008)
6. Tanaka, K., Nakashima, H., Noda, I., Hasida, K., Frank, I., Matsubara, H.: Mike: An automatic commentary system for soccer. In: Multi Agent Systems, 1998. Proceedings. International Conference on, IEEE (1998) 285–292
7. Frank, I., Kumiko, T.I., Okuno, H.G., Nakagawa, Y., Nakadai, K., Kitano, H., Akita, J., Maeda, K.: And the fans are going wild! sig plus mike. In: RoboCup 2000: Robot Soccer World Cup IV. Springer (2001) 139–148
8. Schoenmakers, F., Koudijs, G., Martinez, C.L., Briegel, M., van Wesel, H., Groenen, J., Hendriks, O., Klooster, O., Soetens, R., van de Molengraft, M.: Tech united eindhoven team description 2013. In: Proceedings of the 17th RoboCup International Symposium (May 2013). (2013)
9. Bruce, J., Balch, T., Veloso, M.: Fast and inexpensive color image segmentation for interactive robots. In: Proceedings of IROS-2000, Japan (October 2000)
10. Biswas, J., Mendoza, J.P., Zhu, D., Etling, P.A., Klee, S., Choi, B., Licitra, M., Veloso, M.: CMDragons 2013 Team Description. In: Proceedings of the 17th RoboCup International Symposium (May 2013)
11. Bruce, J., Zickler, S., Licitra, M., Veloso, M.: CMDragons 2007 Team Description. Technical report, Tech Report CMU-CS-07-173, Carnegie Mellon University, School of Computer Science (2007)
12. Zickler, S.: Physics-Based Robot Motion Planning in Dynamic Multi-Body Environments. PhD thesis, Carnegie Mellon University, Thesis Number: CMU-CS-10-115 (May 2010)
13. Zickler, S., Vail, D., Levi, G., Wasserman, P., Bruce, J., Licitra, M., Veloso, M.: CMDragons 2008 Team Description. In: Proceedings of RoboCup 2008
14. Small Size League Technical Committee: Laws of the RoboCup Small Size League 2013. (2013)