**Summarizing paper**

**Title:** AutoRef: Towards Real-Robot Soccer Complete Automated Refereeing

**The focus of the paper:** Automated referee, AutoRef, for the SSL

**Capability:** Capable of working with either real robots using the standard SSL vision system, or with simulated robots using the simulation system.

**The core of AutoRef:** A finite state machine that tracks the state of the game and determines whether the match should be stopped or restarted at each moment.

**Main Objective:** 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. But they did not implement the detection of all the rules of the game, as some are particularly challenging to identify from a perception point of view.

**Advantages over human referees:** It can be difficult for a human referee to keep track of the fast-moving robots and the ball, whereas this is not a problem for an automated referee. Similarly, an automated referee could enforce complicated rules such as the **offside rule.**

**Architecture:**

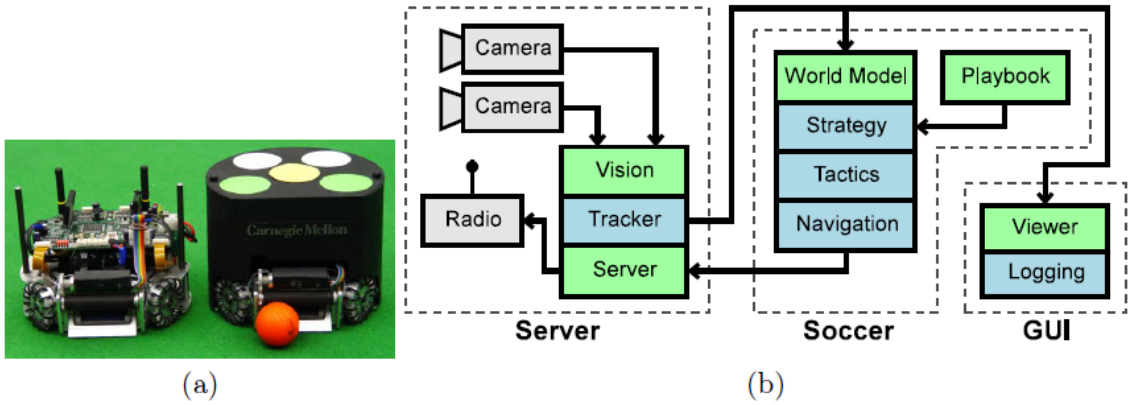


Fig. 1: (a) A CMDragons robot shown with and without a protective cover; (b) The general architecture of the CMDragons offboard control.

**Server system**: Performs vision and manages communication

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**: A high-fidelity 3D soccer simulator

The simulator is a drop-in replacement for the soccer server


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

**Types of considered rules:**

Time-related rules

Ball-related rules

Robot-related rules

**State Machine**

Once per camera frame, AutoRef receives as input the values in the table below:

| 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 |

They devised a finite state machine with transitions governed by the input values, which allows 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:** Occurs while the normal game is happening. AutoRef observes the state of the game until a condition is met which means the game 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.

Other commonly occurred events:

**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. AutoRef compares the positions of the robots to the saved positions; **once all robots have moved**, it signals a **kick-off** and enters **Wait-Stop**.

**Delay-Wait:** This state only occurs **for one frame at a time**; it occurs when AutoRef needs to set the game state to a value that 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.

**Graphical Explanation of the State Machine**

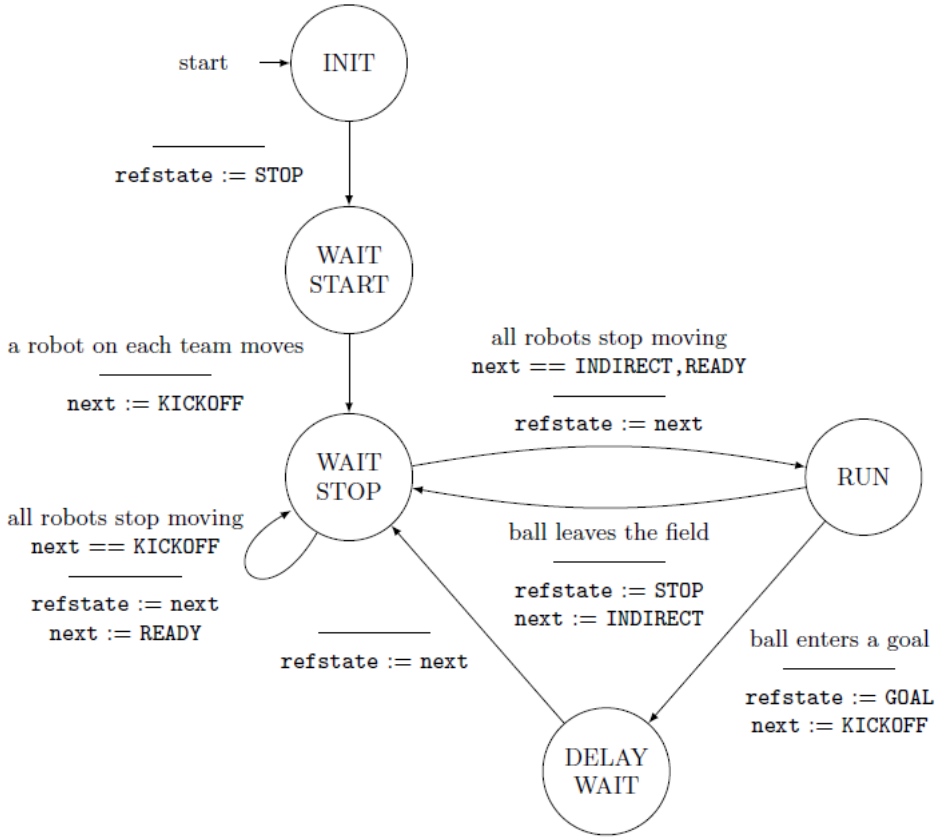


Fig. 3: The finite state machine describing AutoRef.