

The magmaOffenburg 2016 RoboCup 3D Simulation Team

Klaus Dorer, Jens Fischer, Stefan Glaser, Duy Nguyen, Michael Obrecht,
David Weiler¹

Hochschule Offenburg, Elektrotechnik-Informationstechnik, Germany

Abstract. After having described many different aspects of our team software in previous years, in this paper we take the freedom to describe the magmaChallenge framework provided by the magmaOffenburg team. The framework is used as a benchmark tool to run different challenges like the running challenge in 2014 or the kick accuracy challenge in 2015. This description should serve as a documentation to simplify the maintenance by the community and to add new benchmarks in the future.

1 Introduction

One important aspect of RoboCup is its strong community. This community is intended to be an open community. It lives from efforts teams spent outside their own code into the infrastructure used. One such effort is the creation of a benchmark framework in which it is relatively simple to define new benchmarks using the 3D soccer simulation. The magmaChallenge software is such a framework openly available at our github page [1]. In this document we describe the overall architecture of magmaChallenge (Section 2), the benchmark class design 3, the frontend from a user perspective 4 and the benchmarks included so far 5.

2 Architecture

The overall architecture of magmaChallenge is shown in Figure 1. In order to run benchmarks, one first requires the soccer simulation server software rcssserver3d [4] available at [2]. RoboViz is required to visualize the action of the robots and is available at [1]. The agent component is the robot control software that is benchmarked and will typically be provided by the user of magmaChallenge. Example agent software is provided at [3].

2.1 Proxy

The proxy is a slightly modified version of the magmaProxy available at [1]. The original proxy is provided as a jar file in folder lib. Modifications are implemented in package magma.tools.benchmark.model.proxy and includes three changes:

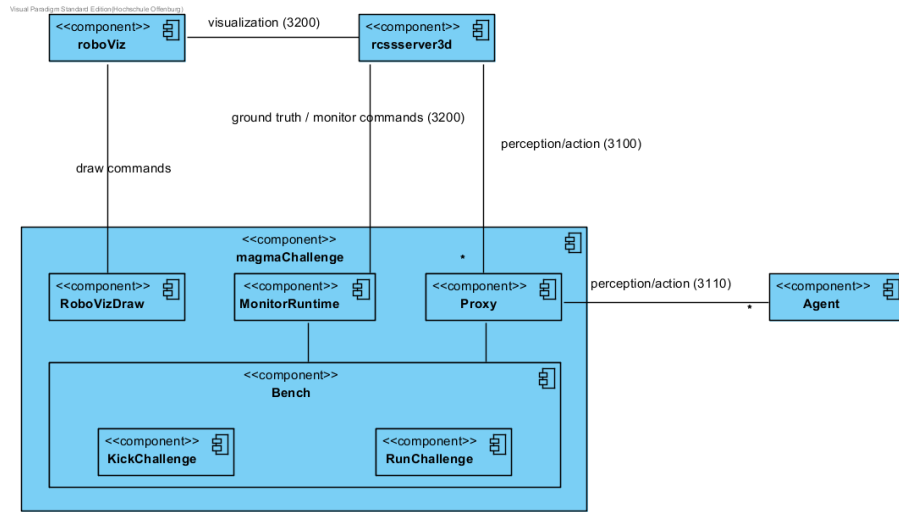


Fig. 1. Architecture of magmaChallenge benchmark tool.

- It searches the perception of agents for force resistance perceptions in order to provide information whether the legs of an agent touch the ground.
- It blocks action messages containing a beam command, since beaming is not allowed during benchmarks.
- It adds beam commands to the action messages in case the magmaChallenge wants the agent to be beamed. This can be used in cases, where the drop height of an agent needs to be specified, which is not part of the monitor protocol.

2.2 MonitorRuntime

The monitor runtime is a component that connects to the monitor port of the soccer simulator in order to get ground truth of the simulation situation. It also allows to modify the simulation with the commands specified in the monitor protocol. The commands include beaming the ball with three position and three speed coordinates, beaming a player with three position coordinates and the amount of z-rotation (the z coordinate seems to be ignored by the server), commands for changing the playmode, especially for kickoff and some other commands specified in the monitor protocol (see `IServerCommander` interface). The monitor runtime has not yet been released as source on github and is provided as `monitorRuntime.jar`.

2.3 RoboVizDraw

RoboVizDraw is a small utility that provides access to roboViz' capability to overlay drawings into the soccer visualization. Its commands include drawing

points, lines, circles, spheres, polygons, annotations and selecting agents. It is provided as utility inside the `monitorRuntime.jar`.

3 Design

Figure 2 shows the design of the magmaChallenge. The overall architecture is a model-view-controller architecture. In this chapter we only detail the model part and only the important classes within it. The shaded area contains the classes of the magmaChallenge framework.

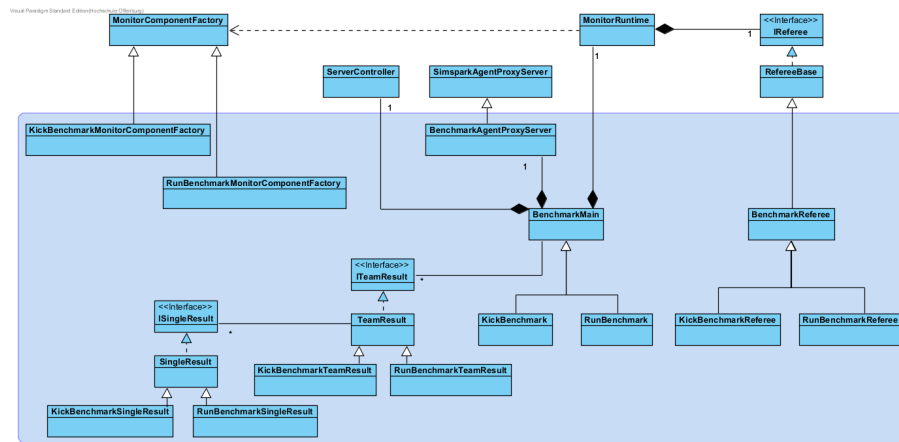


Fig. 2. Design of the model component.

The framework cares for launching the server and the agent or agents that should participate. It will detect, if an agent is connecting directly to the server not using the proxy, which is not allowed. It will also run a specified number of oversampling runs, if required by the benchmark.

3.1 Referee

The referee classes specify the rules of a benchmark. A referee is triggered by the monitor runtime each time it receives an update from the server. The `onDuringLaunching()` and `onStartBenchmark()` methods should be overwritten to specify the setup of a benchmark. The `onDuringBenchmark()` method defines the rules of a benchmark while it is running and when it is finished. If it returns true, the benchmark is over. The `onStopBenchmark()` method is then called to calculate the utility of the benchmark run.

3.2 Results

The result classes contain benchmark specific results. `SingleResult` represents the result of a single benchmark run. `TeamResult` is the result of one team averaged over a specified number of single results. The benchmark contains a list of team results, one for each competing team in a challenge.

3.3 Component Factory

The monitor runtime allows to inject an abstract factory to use for creating its subcomponents. This is used in the kick benchmark and run benchmark to create the proper referee component.

3.4 Other

The `ServerController` is a utility that allows to start, stop or kill a soccer simulation server instance. Class `SinglePlayerLauncher` allows to start an instance of an agent by calling a Unix shell script. Class `TeamConfiguration` holds all team specific benchmark configuration information. Finally, class `RunInformation` contains all information required for a single benchmark run.

4 Front-end

Figure 3 shows the front-end of `magmaChallenge`, here for the kick challenge for a single team. The drop-down box allows to select the benchmark that should be run. Right of it, the values of some parameters can be specified. The table's rows will show all teams participating in a challenge. The columns specify team-specific and benchmark-specific configuration as well as diagnostic information opened when clicking the button in the status column.

The button bar has following buttons:

- **Open Start Script...** Opens a file open dialog to select a start script of a single team for which the benchmark should be run.
- **Open Competition...** Opens a file open dialog to select a configuration file that specifies the parameters for each team.
- **Test** Runs a test for each team in the table. The test will start a single run of the the benchmark for each team and stop it after a short time. The test is to make sure that the software of all participating teams works as expected.
- **Competition** Will run a complete competition, i.e. the selected benchmark for each team in the table.
- **Stop** Will stop the current competition/benchmark after the next oversampling run is finished.
- **Stop Server** Will stop or kill the currently running `rcssserver3d`.
- **About** Will show author and version information.

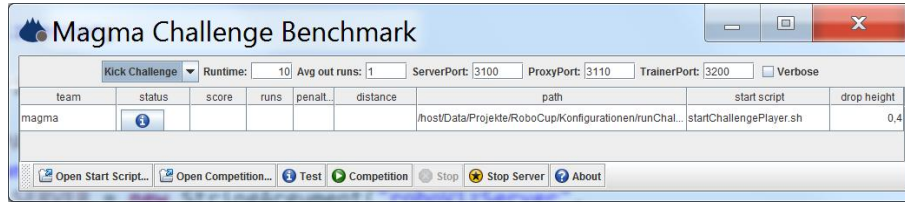


Fig. 3. magmaChallenge front-end (here for the kick challenge).

5 Benchmarks

In this section we describe the benchmarks that are available so far.

5.1 Run Challenge

Our league has committed itself to be the first league that has running robots. One measure of this has been a run challenge in RoboCup 2014.

Requirements Teams that want to participate have to provide the following:

- A start script with name 'startChallengePlayer.sh' that starts a single player of the team which runs forward as fast as it can. The script has to accept the server ip and a port to connect to. Example: `./startChallengePlayer.sh 127.0.0.1 3110`
- No beams permitted (commands including a beam will be entirely ignored by the server).
- The walk has to be 'human like' (no strange crawling or saltos or similar, judged by human)

Evaluation The score of the team is evaluated as the sum of

- the speed (in m/s)
- the relative amount of time both legs are off the ground.

The speed is measured in a 10 seconds run. The time starts when the player crosses the start line or 4 seconds after the player was beamed (by the automated referee) to its start position (-13.5, 0) which is 0.5 m behind the start line (which ever happens first). The speed is then the distance in only x-direction divided by the time to give m/s. If a player falls, this try stops and 2 meters are subtracted as a penalty from the player's current position (but at least 0 m overall distance). The time accounted in such a case will be the full 10 seconds. A player is decided to be fallen if the z part of the up vector is less than 0.6 or the z coordinate of the torso center is below 0.25.

The relative amount of time both legs are off the ground is determined by counting the cycles in which all (both) force sensors have a length of the force less than 0.01 divided by the number of available cycles (within the 10 seconds run). Both measures are averaged out in 10 trials for each participating team. The final sum is rounded to 3 digits. Equal scores will result in the same place.

5.2 Kick Accuracy Challenge

The goal of the kick challenge is to measure the precision of kicks of increasing length with respect to where the ball stops.

Requirements Teams that want to participate have to provide the following:

- A start script with name 'startChallengePlayer.sh' that starts a single player of the team walks to the ball and kicks it to the center of the field. The script has to accept the server ip and a port to connect to as well as the x and y coordinates of the player's initial position it is beamed to. Example: `./startChallengePlayer.sh 127.0.0.1 3110 -9.5 2.0`
- No beams are permitted (commands including a beam will be entirely ignored by the server).

Evaluation The score of the team is evaluated as the average of the distances to the target position. The accuracy is measured in ten kicks from random positions in the own half and a 45 degree cone seen from the destination spot. The start positions of the player are of ever increasing distance (3 - 12 meters) to the destination spot. They are the same for all teams. The destination spot is the center of the field. The time starts when the player gets closer than 0.5 m to the ball or 3 seconds after the player was beamed (by the automated referee) to its start position roughly 1 m behind the ball (which ever happens first). An attempt is over, if

- the player leaves a 2 m circle around the initial ball position,
- the ball leaves that circle and stops,
- the ball does not leave the circle 5 seconds after start.

The final sum is rounded to 3 digits. Equal scores result in the same place.

References

1. <https://github.com/magmaOffenburg>
2. http://simspark.sourceforge.net/wiki/index.php/Installation_on_Linux
3. <http://simspark.sourceforge.net/wiki/index.php/Agents>
4. O. Obst and M. Rollmann, *SPARK A Generic Simulator for Physical Multiagent Simulations* Computer Systems Science and Engineering, 20(5), September 2005