Robocup MSL workshop 2017

Ton Peijnenburg[1], Jaap Vos[2], Wouter Kuijpers[3], Ricardo Dias[4]

[1]VDL Enabling Technologies Group, Eindhoven, [2]ASML, Veldhoven, [3]Eindhoven University of Technology, [4]University of Aveiro

RoboCup 2018 MONTRÉAL · CANADA

INTRODUCTION

Tech United Eindhoven, ASML Falcons and VDL Robot Sports hosted the 7th International RoboCup Middle Size League Workshop. The workshop was held at the 16th, 17th and 18th of November, in Eindhoven, the Netherlands.

OBJECTIVES AND PROGRAM

Thursday, 16th November

A mini-symposium was held in at the Eindhoven University of Technology, where the participating teams showed their latest improvements and a guest speaker talk by Nuno Lau of CAMBADA explaining the details of the simulation leagues.

Friday, 17th November

The second day took place at the ASML Falcons and VDL Robot Sports MSL practice field. This day was dedicated to a joint brainstorm on a common MSL simulator. The results of this session can be found in the Wiki of the project repository.

Saturday, 18th November

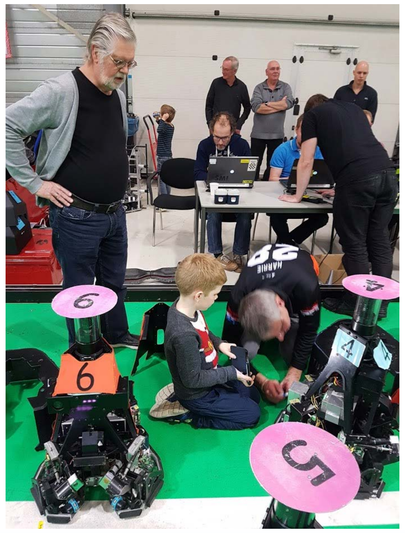
The third day featured the wrap-up session. Teams were assigned to the MSL simulator tasks and a discussion about the league roadmap followed. The participants also came up with a list of proposals for rule changes for 2018, which is to be delivered to the Technical Committee in charge. After lunch, a mini-tournament of friendly matches began.

Sunday, 19th November

The morning was dedicated to more matches of the mini-tournament.

MSL simulator

The MSL Simulator is a joint effort of multiple active teams in the MSL in building a simulator tailored for the RoboCup MSL league. The first efforts in running simulated versions of MSL matches between different teams were made in the MSL Workshop 2016 (Kassel, Germany) and a thorough discussion on the requirements and technologies to use was continued in the MSL Workshop 2017 (Eindhoven, Netherlands).



FINDINGS

Simulator Server

- An annual MSL Simulation League is already being organized in China, which uses the open-source NuBot Team simulator (simmatch, a.k.a. GroSim = Gazebo + ROS).
- A member of the NuBot team was present during the workshop and on the presented this simulator during the workshop presentation session.
- The general feeling of the other teams was that the Gazebo was the preferred simulation engine to use, but would like to drop the ROS dependency. So, the GroSim has been selected as a baseline starting point for the MSL Simulator. The idea is to strip ROS out of their simulator.
- Additionally, the ball possession and scrum broker algorithms must be enhanced with respect to the current GroSim approach.
- Because this is a bigger project and we will deviate quite a bit from the GroSim, instead of forking we decided to copy the simulator code and include it in a subfolder of this project.

Sync/Realtime Simulation

- A real-time simulation is the simplest one to develop - server sends the ground truth to the client and the client replies with their outputs as fast as they can.
- A sync'd simulation essentially pauses simulation waiting for the client to respond
- Sync'd simulations would enable 2 nice features:
- Remote simulations (minimizes connection delay impact, otherwise the remote team is always in disadvantage)
- Speed-up simulation (a simulated match could potentially take less time than real matches)
- Most teams are using system time internally in their agents. This makes it impossible to take advantage of speed-up simulation.
- We settled that for now, we are not yet interested in remote and speed-up simulations, so we defined to have a real-time simulator working first.

Interface And Communications

- Server-Client paradigm - The simulator act as a server and communicates with several clients
- Types of clients: Team Client - 2 max, Referee Client - 1 max, Visualization Client - infinite

FINDINGS (continued)

- Will use ProtoBuf for messages between server and client, since Gazebo already uses ProtoBuf internally
- A TCP/IP link has been chosen to scale in the future to remote and sync'ed simulations
- Single TCP port for all communications
- Make use of docker images to share agents between teams - no need for distributing source code, nor binaries + configuration files, etc. An opponent team only needs to provide a docker container that is able to communicate with the server and receive commands from the refbox.

Robot Models

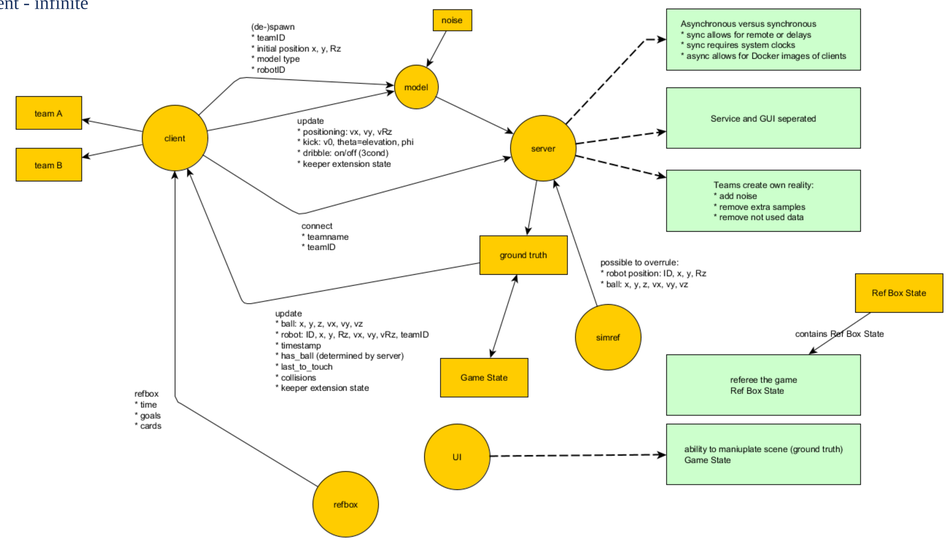
- A default level of abstraction has been defined that should work for most teams generally.
- Four models initially considered: TRIANGULAR, SQUARE, CIRCULAR, KEEPER
- These models are not purely visual. The model sends also intention of interaction with the ball (then, a broker on the server decides which team has control of the ball)
- If a team wants to go deeper in the level of simulation with their robots, they can provide their model in a form of a Gazebo plugin that communicates with their Team adapter client extra information that was not included in the default models.

Noise Simulation

- The server communicates groundtruth (noise-free) data to the clients
- It is responsibility of the team to integrate similar noise levels as their robots have in their team adapters upon receiving this groundtruth info

Sim Referee

- In a first stage, the idea is to reuse the current official Referee Box application - the RefBox2015
- The RefBox2015 application should be able to receive commands from an automatic referee application
- This automatic referee is able to communicate also with the Gazebo server: has access to ground truth information and is able to reposition robots and the ball on the field.



FOLLOW-UP TASKS

Initial Documentation - FALCONS

- Overall architecture
- Sequence diagrams

Documentation - ALL TEAMS

- Keep it in mind for all tasks below

[Referee] RefBox API verification - FALCONS

- Check if it's implemented, working locally...?
- Check 2 local Clients

[Server] Define the Server interface - VDL

- Messages (attributes per message), Serialization (ProtoBuf), Transport (TCP/IP)

[Gazebo] Adapt/Spoon GroSim - CNC, NuBot

- Strip ROS / start with gazebo and import changes on GroSim ??

[Gazebo] Create default models: - NuBot, CNC

- TRIANGULAR, SQUARE, CIRCULAR, KEEPER
- Not only visual model
- Model sends intention of interaction with the ball, a broker on the server decides

[Server/Gazebo] Create the interface with clients on the server side - CAMBADA, VDL

[Client] Create a sample client (use defined messages to communicate with the server) - CAMBADA, VDL

[Referee] Create the automatic referee (arbiter) + interaction with server - TUE, FALCONS

[Docker] Dockerize the Server (use Ubuntu 16.04) - TUE

- Use scripts to install dependencies and avoid having huge images
- Provide a tutorial for teams to create images

[Docker] Provide a docker image per team - ALL TEAMS

FUTURE WORK

[Client] Instead of using gazebo client, use a custom visualization tool

[Server] Add sync mode to enable remote and speed-up matches

ACKNOWLEDGMENT

A big thank-you to Ricardo Dias for excellent reporting on the workshop; the reporting has also been added to the MSL wiki under workshop dissemination and a GitHub archive has been created for the joint MSL simulator project.

PARTICIPANTS

ASML Falcons (ASML Veldhoven, Netherlands)

Tech United Eindhoven (Technical University of Eindhoven, Netherlands)

VDL Robot Sports (VDL Enabling Technologies Group, The Netherlands)

CAMBADA (University of Aveiro, Portugal)

Carpe Noctem Cassel (University of Kassel, Germany)

NuBot (National University of Defense Technology, China)

Hibikino-Musashi (Kyushu Institute of Technology, University of Kitakyushu, Japan)

Guest Speaker: Nuno Lau (University of Aveiro)

Some members from Fontys ICT (Information and Communication Technology Eindhoven, Netherlands)