DAInamite
Team Description for RoboCup 2013

Axel Heßler, Yuan Xu, Erdene-Ochir Tuguldur and Martin Berger
{axel.hessler, yuan.xu, tuguldur.erdene-ochir, martin.berger}@dai-labor.de
http://www.dainamite.de

DAI-Labor, Technische Universität Berlin, Germany

1 Introduction

The team DAInamite from DAI-Labor of Technical University Berlin wants to take part in the *Standard Platform League* (SPL) in the RoboCup 2013 event in Eindhoven, Netherlands with its five NAOs V4 H21 (RoboCup edition). We are new to this league, but not so new to the RoboCup competitions as we started with the 2D soccer simulation league in 2006 (see e.g. [1–3]).

We have collected first experiences regarding the SPL during the RoboCup German Open 2012. We implemented a simple agent with a *sense-think-act* execution cycle in the Java programming language, and used as many of Aldebaran's modules as possible. We collected additional practical experience during a 14-days event called Ideenpark[1] in August 2012. There we played demo games together with members of the NAO Devils SPL team from TU Dortmund, and learned a lot from them about humanoid robot soccer.

After these two events we started from scratch again following a simple methodology: we use modules, which are available and working from Aldebaran. Therefore, we can focus on problems that are more benficial or interesting. We rely on Aldebaran's NAOqi architecture as a communication infrastructure for modules. We prototype our ideas and concepts in the Python programming language. Only time critical parts such as motion and vision are implemented in C/C++.

2 Team Members

The team consists of Bachelor and Master students from the Technical University Berlin (TU-Berlin). They are backed by student assistants and researchers working at the Distributed Artificial Intelligence (DAI) Laboratory, which is directed by Prof. Dr. Sahin Albayrak. The current team members are:

– Reserchers working at the DAI-Lab: Axel Heßler (team leader), Erdene-Ochir Tuguldur, Martin Berger, Yuan Xu and Felix Bonowski.

[1] http://www.ideenpark.de/ideenpark/funbox/roboterfussball

– Student assistants working at the DAI-Lab: Lars Borchert, Stephen Prochnow, Philipp Brock and Leily Behnam Sani.
– Undergraduate students at TU-Berlin: Ruth Motza, Nadine Rohde, Thomas Schlegel, Arne Salzwedel and Lukas Weise.

Prof. Dr. Dr. h.c. Sahin Albayrak is the head of the chair Agent Technologies in Business Applications and Telecommunication (AOT) at the technical University Berlin. He is also the founder and head of the DAI-Lab. The DAI-Lab performs applied research and development of new systems and services and apply and test the solutions in real environments to make them tangible for users. Current application fields are, for example, electromobility, smart grid, home, health, ambient assisted living, security, and service robotics.

Concrete research fields of our team members are agent-oriented software engineering, agent testbeds, cooperation and coordination, machine learning, planning and scheduling, computer vision, and human-robot interaction. The main application field of our NAOs is teaching agent-oriented and robotic principles.

3 Architecture

We are using Aldebaran's NAOqi architecture as the basis. As mentioned earlier, the main policy was to use what was already available and working. Missing functionality is implemented by adding our own NAOqi modules. These new modules are written in either Python or C++, depending on the requirements. In the following those modules are described.

3.1 Motion

We have developed our own motion module in C++ for better performance in soccer games. Especially, we implemented a fast (20 cm/s) omni-directional walk based on *Linear Inverted Pendulum* [4].

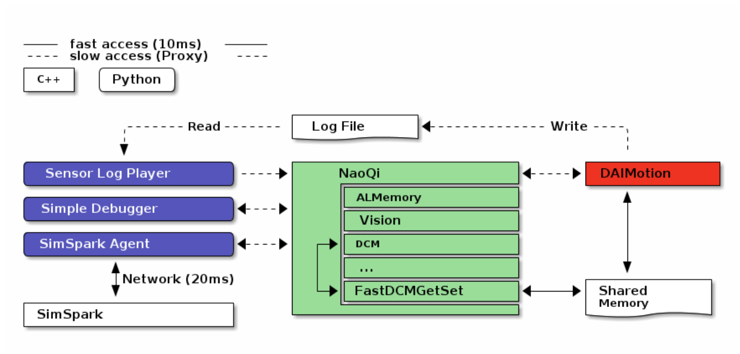

Fig. 1. Architecture of motion module, see text for details.

We also keep our motion module compatible with other modules from Aldebaran. APIs of ALMotion are implemented, and functionality such as *Self-collision avoidance*, a simple *Fall Manager*, and *Smart Stiffness* were developed as well. Furthermore, the module also provides camera's homogenous transformation matrix defived from the robot's configuration and odometry data for other modules.

In order to test and debug our motion module easily, we divided it into several different sub-modules, see Figure 1. The *DAIMotion* can run as a local or remote module. It accesses the DCM through shared memory when it is remote. The module can also run with recorded logfiles and the SimSpark simulator.

3.2 Vision

Our team has reimplemented a calibration free vision algorithm proposed in [5]. Like our motion module, the vision module is also implemented in C/C++ and replaces the Aldebaran video device module. To accelerate the image capturing, we are using *Video4Linux* directly to capture images from both cameras in parallel. By default, the vision module processes only images from the bottom camera. But if requested, the images from the top camera are processed in parallel to locate the goal posts faster. However, the vision cannot guarantee to process both frames in time when two cameras are used.

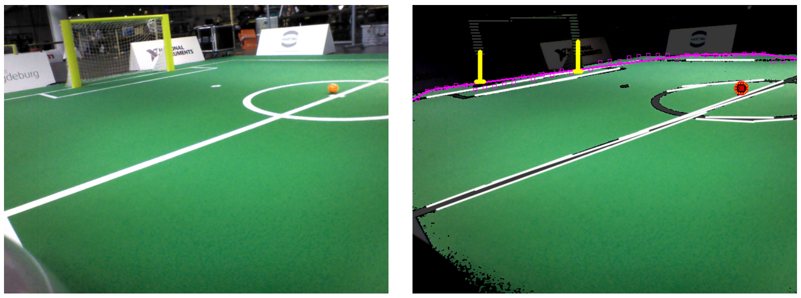

Fig. 2. Original image (left) and a visualization of the processing results(right), the field and detected entities are highlighted. The goal posts are marked in yellow, the field border in magenta, the lines in white and the ball in red color.

Currently, the vision module is able to detect the goal posts, the field border, lines and the ball. The results are written into the agents central memory managed by Aldebaran's module *ALMemory*. Other modules, such as localization, can read the results from this memory. An exemplary result of the vision processing is depicted in figure 2. Detected objects are highlighted in different colors.

Furthermore, the vision module provides methods for debugging and configuring, such as setting camera parameters and enabling/disabling cameras.

3.3 Localization

Localization is prototyped in Python at the moment. Similar to other teams [6, 7], we use a Kalman Filter for this purpose. The measurement of robot's pose is calculated by recognized goal posts, and the prediction of robot's pose is provided by odometry from motion module. The robot's pose has to be tracked continuously, to distinguish the opponent's goal from our own.

3.4 Ball tracking

Like our other high level components, the ball tracking is implemented in Python. Our vision module reliably detects the ball's position if a ball is present in the image and is not heavily occluded. If no ball is present, the vision occasionally returns false positives. To cope with this situation, we have implemented a simple multiple model Kalman filter as in [6]. The global ball position is transformed from the pixel coordinates to the global position on the soccer field using the camera matrix. The ball tracking uses the global ball position as measurement, and returns a list of ball hypotheses containing the predicted ball position, velocity and the kalman filter error.

3.5 Behavior

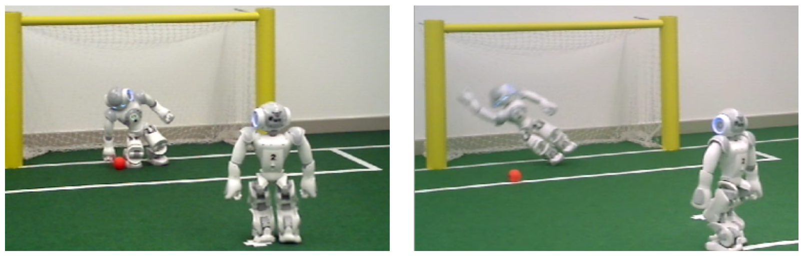

Fig. 3. Snapshot of the goalie performing a save while staying up straight (left) and by falling to the side (right).

A prototyped soccer behavior is implemented in Python. In the beginning of the game all players assume they are on their half of the field to localize themselves, and they are able go to the kick-off positions autonomously.

Currently, there are three roles defined in our NAO team. The robots communicate with each other and share their perceptions: own position, goal positions and ball position. We plan to use the ball perception of the goalie to break the symmetry of two yellow goals.

– *Striker:* The field player who is closest to the ball becomes the striker. It approaches the ball and positions itself behind it into the direction of what it considers to be the opponent's goal. The robot then executes a kick-motion and repeats.

- *Supporter:* The field players who are not currently assigned the striker role become supporters. Supporters do not actively pursue the ball. In the future they should position themselves strategically.
- *Goalie:* The goalie tries to stay roughly in the middle between its own two goal posts. During the game, the goalie computes the direction of the ball movement by using the information provided by the ball tracking module. The robot tries to parry if the direction of the ball points towards its own goal and the ball's velocity exceeds a threshold. It will then try to block the ball by either staying up straight and lowering one arm to the ground or executing a save falling to the side as depicted in Figure 3 respectively.

4 Conclusion

We described our approach to soccer playing NAOs. We showed our first solutions to major problems related to humanoid robot soccer.

We can imagine large improvements when we solve special problems in depth. One of the next steps will be to give the supporter role more meaningful behavior.

Acknowledgments

Large influences came from four German teams, namely: Nao Team Humboldt, Nao Devils TU Dortmund, B-Human Bremen University and Nao-Team HTWK Leipzig. We mainly learned about concepts and algorithms from them and want to thank them at this point. But we did not use any code of them so far.

References

1. Endert, H., Wetzker, R., Karbe, T., Heßler, A., Brossmann, F.: The dainamite agent framework. Technical report, Dai-labor TU Berlin (2006)
2. Endert, H., Karbe, T., Krahmann, J., Trollmann, F., Kuhnen, N.: The dainamite 2008 team description. RoboCup 2008 (2008)
3. Hessler, A., Berger, M., Endert, H.: Dainamite 2011 team description paper. Robocup 2011 (2011)
4. Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., Hirukawa, H.: Biped walking pattern generation by using preview control of zero-moment point. In: ICRA. (2003) 1620–1626
5. Reinhardt, T.: Kalibrierungsfreie Bildverarbeitungsalgorithmen zur echtzeitfähigen Objekterkennung im Roboterfußball. Master's thesis, Hochschule für Technik, Wirtschaft und Kultur Leipzig (2011)
6. Quinlan, M.J., Middleton, R.H.: Multiple model kalman filters: a localization technique for robocup soccer. In Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S., eds.: RoboCup 2009. Springer-Verlag, Berlin, Heidelberg (2010) 276–287
7. Jochmann, G., Kerner, S., Tasse, S., Urbann, O.: Efficient Multi-Hypotheses Unscented Kalman Filtering for Robust Localization. In: RoboCup 2011: Robot Soccer World Cup XV, Springer (2012)