Carpe Noctem Cassel Team Description 2016

Dennis Bachmann, Eduard Belsch, Thore Braun, Nugroho Fredivianus, Kurt Geihs, Michael Gottesleben, Stefan Jakob, Thomas Kleppe, Nils Kubitza, Kai Liebscher, Lisa Martmann, Thao Van Nguyen, Stefan Niemczyk, Stephan Opfer, Tobias Schellien, Phileas Vöcking, Lukas Will, Andreas Witsch

Distributed Systems Research Group, University of Kassel, D-34121 Kassel, Germany

Abstract. In this paper we describe the most important improvements of our middle-size league robots in the last year. Concerning the software components, we ported our framework to C++ and implemented a new localization algorithm, which provides global localization with a low computational overhead. Furthermore, we developed our new hardware devices, which improve the maintainability and robustness of our robots. In particular, we made a new concept for the kicking and ball-handling mechanism, the omni-vision, and the low-level actuation control device.

Keywords: Carpe Noctem Cassel, RoboCup MSL, Team Description Paper, Mechanical Concepts, Localization, Path Planning

1 Introduction

The RoboCup Team Carpe Noctem Cassel (CNC) has been founded in 2005 as part of the Distributed Systems Group of the University of Kassel. Since that time, we participated in eleven national and international tournaments. The purpose of the team is manifold. One purpose is to give undergraduate students an early opportunity to research related and practical topics from the area of AI and robotics. Carpe Noctem Cassel is a student-based team, with currently one PostDoc, three PhD candidates, and 12 undergraduate students. Another purpose of the team is to offer a sophisticated testbed for the Distributed Systems Research Group in general. Our research mainly focuses on mechanisms to model and establish cooperative behaviour. In this context, we developed the multi-agent coordination language ALICA (A Language for Cooperative Interactive Agents) [5,7]. ALICA models multi-agent behaviour from a global perspective, which provides an intuitive understanding to the developer. Its fully distributed implementation strongly supports dynamic team composition and unreliable communication, as it is tolerant towards packet loss and delay. Therefore, ALICA agents track the actions of the others using their local observations and updates these based on the information received over the network. As described in Section2, in the last two years, we ported ALICA from C# to C++ to address a broader user community and improve the efficiency.

To facilitate the implementation of ALICA behaviours, we provide a path planner, which relies on a Voronoi Diagram [3]. Each cell of the diagram represents a possible obstacle on the field. Since the Voronoi edges describe the

most distant line between a pair of obstacles, we can apply an A^* search on the Voronoi edges to avoid other robots. In the context of the new C++ version of ALICA, we also investigated a new implementation of this path planner, which is now based on the CGAL library [8], see Section 3 for details. In Section 4, we describe a new localization algorithm, which combines two well-known approaches, to smoothly switch between a low runtime and the ability to provide a global localization.

The development of an efficient cooperative behaviour in robotic soccer requires a robust and maintainable platform. Hence, we develop new sensor and actuation devices. In particular, we reworked the camera system, which enables an unobstructed view (see Section 6). We also designed a new kicking device, which allows to adjust our kicking angle within a continuous range, as presented in Section 5. Finally, we developed a new control board for the low-level actuation devices. Here, we could reduce the efforts for maintenance and monitoring, by integrating a ready-to-use ARM processor board, which runs a Linux system, as described in Section 7.

2 Porting Software to C++

At the International MSL RoboCup Workshop in 2013, a survey about our high-level multi-agent coordination framework ALICA revealed, that most of the participants would only consider to use our software, if it would be coded in C++. At that time, most of our software, including ALICA, were written in C#. Due to the survey, the prevalence of C++ in the robotic community, and poor tool support for C# under Linux, we ported our C# software components to C++.

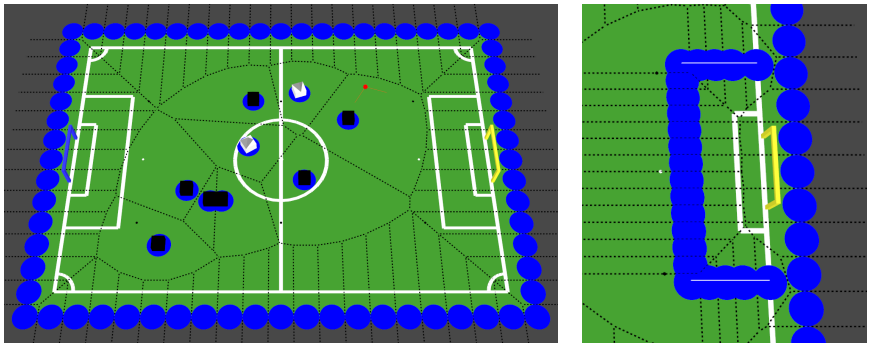
Furthermore, our complete framework is now publicly available on GitHub under the open-source MIT License. This even includes all our strategies, sensor data processing algorithms, controllers, and supplementary tools for the RoboCup MSL domain. Although our software utilises ROS [4] as inter-process communication middleware, we took care that ALICA is designed as independent as possible. The only requirements/dependencies of ALICA are a Linux based operating system, the C++11 Standard Library, and our own configuration file library. ALICA provides easily implementable interfaces for arbitrary communication middlewares and custom clocks. The clock interface, for example, allows for replaying log files with adjustable speed and for synchronous running with slower simulators.

Another ported software component, we want to mention here, is our constraint satisfaction solver [6,9]. It allows to solve arbitrary positioning problems, e.g., one-on-one defence or double pass positioning with little programming effort.

3 Path-Planning with CGAL

Our current path planner utilises a road-map approach that creates a Voronoi Diagram of the robots environment (see Figure 1(a)). Porting our software from

C# to C++ enabled an easy integration of the Computational Geometric Algorithms Library (CGAL) into our ported path planner. CGAL offered us several improvements over our own C# implementation. The run-time performance was improved in general, as CGAL allows to build a Voronoi Diagram incrementally, and to remove obstacles from the Voronoi Diagram. Therefore, we are able to influence our path planning by changing the Voronoi Diagrams depending on the situation. The MSL RoboCup rules, for example, forbid to have two defenders in the penalty area. If one of our defenders is inside the penalty area, we change the Voronoi Diagrams of all other field players in a way that avoids target points inside, as well as, paths through the penalty area (see Figure 1(a)).



(a) In-Game Voronoi Diagram (b) Blocked Penalty Area

Fig. 1. Voronoi-Based Path Planning

The Voronoi Diagram itself has several properties, which are interesting for the RoboCup MSL domain. If the Voronoi cells are formed by one robot each, then, assuming all robots have the same velocities and accelerations, a Voronoi cell is the area the corresponding robot can reach before all other robots. Therefore, the Voronoi Diagram is an ideal geometric data structure for determining pass points and optimizing the positioning for zone defence strategies.

4 Carpe Noctem Dynamic Localization

The ability to determine the position of robots is one of the key requirements in order to enable a strategic and comprehensive game play. Over the last years, we evaluated two different localization strategies during our participation in multiple RoboCup tournaments. In the first years, a particle filter was used. Here, 1200 position hypotheses (particles) are sampled and evaluated based on detected lines. These were optimized by an approach similar to a genetic algorithm. Therefore, the best matching particles are used to sampled a new generation. After few generations, the best matching particle was used as robot position. The particle filter allowed us to localize a robot without requiring a known starting

position. However, the high amount of computation time (approximately 10 ms on a Intel Core2Duo with 1600 Mhz) was a major drawback.

The RoboCup Team Tribots developed an localization approach based on error-minimization [2]. This approach tracks the robot position and refines it based on detected lines. Therefore, the error function, which computes the quality of the matching between detected lines and tracked position is derivated for the three position parameter (x, y, and α). Following this gradient to a local optimum leads to an minimization of the matching error. However, the approach requires an initial position hypothesis. Hence, we lost the ability to localize a robot at an arbitrary field position. In conclusion, a delocalized robot needs to be taken out of the game.

Our new localization approach called CANDL (CArpe Noctem Dynamic Localization) combines both approaches to enable a fast localization and a relocalization of our robots. Therefore, we sample a dynamic number of n particles, similar to a particle filter. Afterwards, the best matching particles are selected and are refined by m optimization steps. CANDL switches smoothly between the particle filter and the error-minimization approach by adjusting n and m. By choosing high values for n (e.g. 1200) and small values for m (e.g. 0) the localization behaves like a particle filter. Vice versa, small values for n (e.g. 1) and high values for m (e.g. 20) results in a error-minimization like behaviour. As a result, we can localize our robots at any position on the field. Furthermore, after an initial localization a dynamic switch to the error-minimization enables a fast and precise position determination. A final algorithm for a dynamic adaptation of both parameters is currently work in progress. We plan to make CANDL publicly available on GitHub under the open-source MIT License.

5 New Concept for Kicker and Ball-Handling

During the last year, we developed a new mechanical concept for our kicker and ball handling devices (see Figure 2). The CAD files and mechanical drawings are part of the qualification material. We achieved several improvements, compared to the earlier versions. The height over ground of the kicking lever can be changed continuously. This allows for kicking trajectories that are more flexible than by adjusting the kick power. The current position of the kicking lever is monitored by an integrated sensor. The motor which is driving the kicking lever has a break mounted that fixates the lever during a kick. The overall configuration is shown in Figure 3(a).

The active ball handling concept consists of two arms that are very adaptable to different balls and driving manoeuvres of the robot. Each arm includes a damper for receiving/stopping fast passes and is hold in position with several springs. The spring configuration allows to choose different amounts of contact pressure between the ball and the wheel. The actual position of each arm is monitored by a linear sensor, as shown in Figure 3(b).

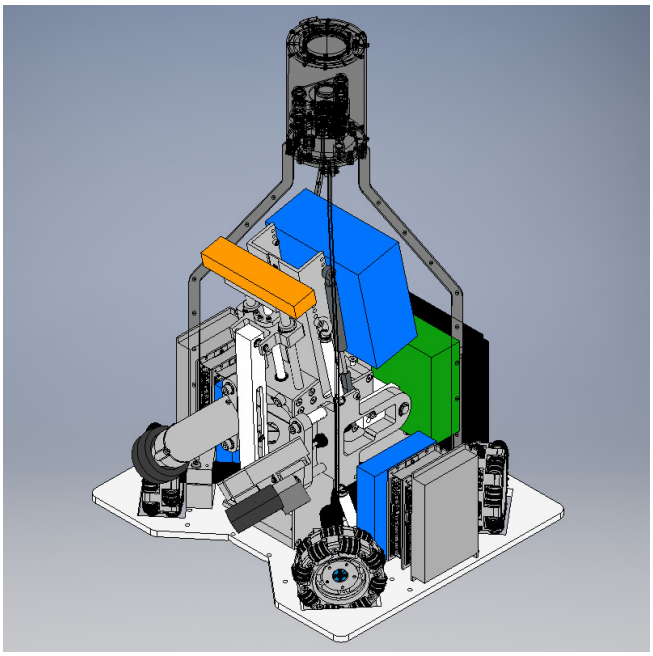
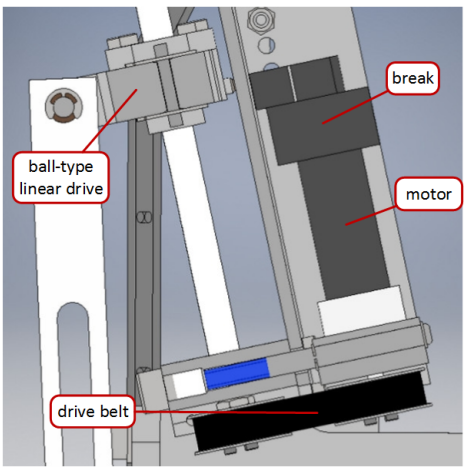
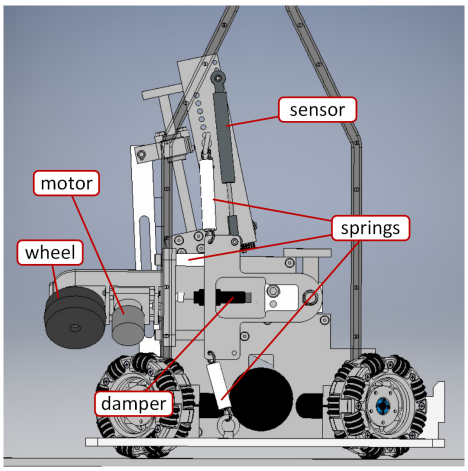


Fig. 2. Overview of the new Kicker and Ball-Handling Concept



(a) Kicker Concept

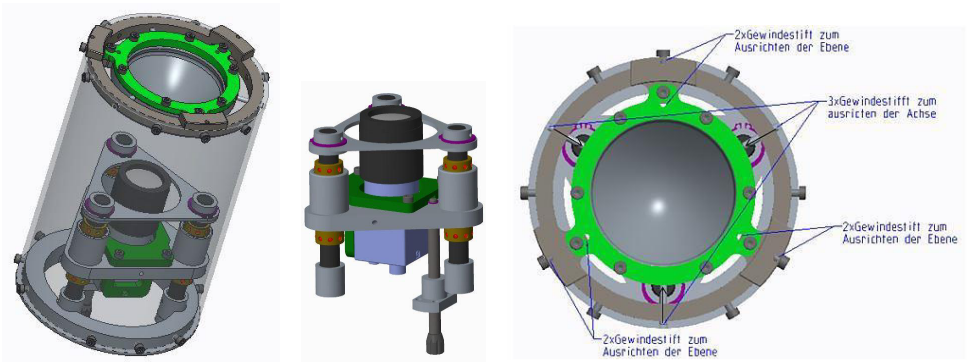


(b) Ball Handling Concept

Fig. 3. Details of the Kicker and Ball-Handling Concept

6 Omni-Vision Concept

Calibrating the vision system is a time taking job for most of the middle size league teams. Each environment provides different lightning conditions and the alignment of mirror and camera should be as precise as possible for crucial matches. Therefore, we developed a new concept for our omni-vision system, as pictured in Figure 4(a). The system combines several improvements compared to our old system. At first, the acrylic glass cylinder allows for a free sight in all directions. Furthermore, the distance between camera and mirror can easily be adjusted without tools, as seen in Figure 4(b). Finally, the plane as well as the axes of the mirror can be adjusted to the cameras position. The corresponding setscrews are shown in Figure 4(c).



(a) New System (b) Camera Adjustment (c) Mirror Adjustment

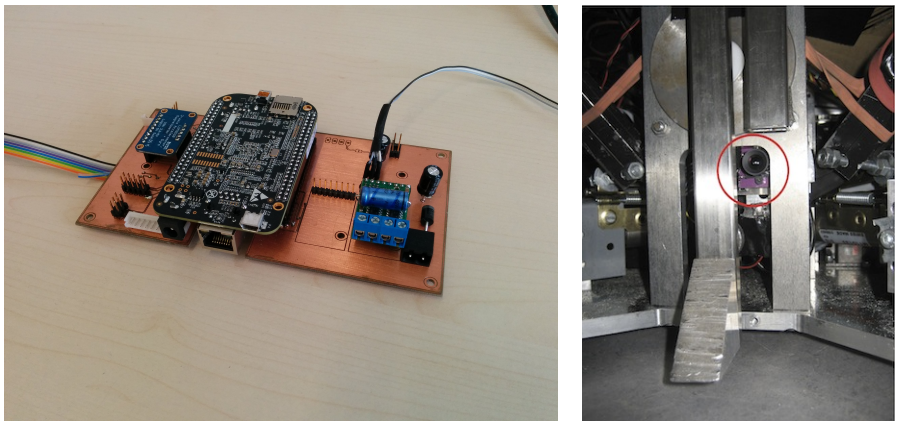
Fig. 4. The new Omni-Visio Concept

In order to avoid reflections on the acrylic glass cylinder, it is covered with an anti-glare foil. The weight of the complete system, including our camera, is 960g. The technical drawings are part of the qualification material on our website.

7 Redesign of Actuator Board

The Actuator Board is a self-made printed circuit board (PCB) integrated in each of our robots (see Figure5(a)). It has several tasks to manage. One of the most important ones, is to control the active dribbling device according to the robots movements and the feedback from the optical flow sensor (see Figure 5(b)). The optical flow sensor detects the ball's angular velocity in two dimensions. The robot movements are measured by the inertia measurement units (IMU, LSM9DS0) on the actuator board. The motors of the active dribbling device are controlled by to small motor controllers, which are also a piggyback circuit

board of the actuator board. Another job, is to send messages to our PC, when one of several buttons at the robot is pressed. The current button functionalities are reinitialising the localisation and restarting processes.



(a) Actuator Board Prototype (b) Optical Flow Sensor

Fig. 5. Actuator Board and Optical Flow Sensor

The new design of the Actuator Board utilises the Beagle Bone Black [1] micro computer board. It is similar to a Raspberry Pi, but its design is much more geared towards a robotic application scenario. Its specifications are 1 Ghz ARM Cortex-A8 processor, 512 MB RAM, 4 GB flash storage, and several buses and interfaces like Ethernet, SPI, I2C, UART, CAN, 65 GPIOs, eight PWMs, and seven analogous I/Os. This allows us to communicate between the PC and low level sensors with TCP/IP on the PC side. That way the complexity of the self-made parts of the actuator board is reduced to a simple breakout board for the Beagle Bone Black.

8 Conclusions

The Carpe Noctem Cassel Mid-Size RoboCup team of the Distributed Systems Research Group at the University of Kassel has a research focus on lean software architectures, mechanisms to model and establish cooperative behaviour, and cooperative artificial intelligence. We use the RoboCup scenario as a testbed for our research as well as for education and teaching efforts. Our robots and the control software were designed from ground up with modularity and extensibility in mind. We look forward to evaluate our ported software framework and innovative localization during the next tournaments.

References

1. BeagleBoard Foundation: Beagle Bone Black. Revision C edn. (2015), http://www.beagleboard.org/BLACK
2. Lauer, M., Lange, S., Riedmiller, M.: Calculating the perfect match: An efficient and accurate approach for robot self-localization. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005: Robot Soccer World Cup IX, Lecture Notes in Computer Science, vol. 4020, pp. 142–153. Springer Berlin Heidelberg (2006), http://dx.doi.org/10.1007/11780519_13
3. Opfer, S., Skubch, H., Geihs, K.: Cooperative Path Planning for Multi-Robot Systems in Dynamic Domains, chap. 11, p. 237–258. In-Tech (nov 2011), http://www.intechopen.com/articles/show/title/cooperative-path-planning-for-multi-robot-systems-in-dynamic-domains
4. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T.B., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software (2009)
5. Skubch, H.: Modelling and Controlling of Behaviour for Autonomous Mobile Robots. Westdeutscher Verlag GmbH (2013)
6. Skubch, H.: Solving non-linear arithmetic constraints in soft realtime environments. In: 27th Symposium On Applied Computing. ACM SAC, vol. 1, p. 67–75. ACM, ACM, Riva del Garda, Italy (2012)
7. Skubch, H., Saur, D., Geihs, K.: Resolving conflicts in highly reactive teams. In: Kommunikation in Verteilten Systemen 2011. Open Access Series in Informatics, Open Access Series in Informatics (2011)
8. The CGAL Project: CGAL User and Reference Manual. CGAL Editorial Board, 4.7 edn. (2015), http://doc.cgal.org/4.7/Manual/packages.html
9. Witsch, A., Skubch, H., Niemczyk, S., Geihs, K.: Using incomplete satisfiability modulo theories to determine robotic tasks. In: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on. IEEE, IEEE, Tokyo (11 2013)