pRoPhEt MAS: Reactive Planning Engine for Multiagent Systems

Daniel Saur and Kurt Geihs

Abstract Autonomous mobile robots can substantially increase their effectiveness in dynamic environments using planning. This paper proposes a design for a soft real-time planning system for autonomous robots and offers a generic and modular approach to control a team of robots. Our system pRoPhEt MAS is based on ALICA (A Language for Interactive Cooperative Agents) and offers the coordination of team behaviors at runtime. In the evaluation scenario the system pRoPhEt MAS uses a state of the art planner "Fast Downward Planning System." The evaluation focuses on planning during execution time. The team executes the best solutions found, selected by the heuristic, under certain time constraints. The results show that the execution with soft real-time planning is as good as sequential planning and execution. Hence, it offers the ability to react quickly in dynamic domains.

Keywords Multiagent systems · Dynamic domains · Planning

1 Introduction

Dynamic and nondeterministic domains such as RoboCup MSL[1] impose challenges for the realization of fast reactions to world changes with dynamically changing behaviors. Furthermore, dynamic domains need an intuitive goal description for modeler. Oftentimes, the plan of a team of agents, describing the activities for every agent, is manually coded and covers all possible foreseen situations. With an

[1] http://www.robocup.org/robocup-soccer/middle-size.

D. Saur (✉) · K. Geihs
Distributed Systems Group, University of Kassel, 34121 Kassel, Germany
e-mail: saur@vs.uni-kassel.de

K. Geihs
e-mail: geihs@vs.uni-kassel.de

E. Menegatti et al. (eds.), *Intelligent Autonomous Systems 13*,
Advances in Intelligent Systems and Computing 302,
DOI 10.1007/978-3-319-08338-4_12

increasing number of agents or complex problems, the overhead for maintenance, modeling, and testing increases enormously. This approach can react only on the modeled multiagent plan, and is therefore problematic because it becomes very likely that a possible situation is missing among the manually coded ones. Solving planning problems takes a certain amount of time. Therefore, fast reaction is not possible if the environment changes rapidly with state of the art planning.

Hence, the challenge of multiagent systems in a dynamic environments is to optimize the planning process. The IPC (International Planning Competition) is held every year to tackle complex planning problems; however, the search is limited to 30 min. This length is not suitable for dynamic changing environments. Another approach is to join planning and executing into an anytime planner.

In this paper, we introduce a novel approach for soft real-time planning, focusing on teams of agents acting dependent on the planning process. pRoPhEt MAS observes the searching process and selects the best solution determined by the planner heuristics in due consideration of the soft real-time limit. We use state-of-the-art planning systems and plans that depend on the actual world situation. If the team is not able to reach the goal, pRoPhEt MAS plans again.

The approach uses the agent-oriented language ALICA [10] to describe, coordinate, and execute behaviors. We expand ALICA by an element to define planning problems. These problems are described with PDDL [3] (Planning Domain Description Language). It offers an interface for planners on the basis of the PDDL description. PDDL enables the interchanging of planning engines easily. Furthermore, we integrate state-of-the-art planning systems to cope with planning problems. Finally, we present pRoPhEt MAS, which uses this planning system and reacts on the search progress and the actual world situation accordingly using a soft real-time limit.

In Sect. 2 we start to discuss related works. Next, we introduce the basics of ALICA in Sect. 3, which offers the basis for describing team activities (plans). ALICA allows the modeling of behaviors for a team from a global perspective instead of interfacing single agent programs. We will present an extension to the language of ALICA for supporting planning problems in order to describe a goal description, which the team of agents must reach, independent of any particular world state. Finally, we present in Sect. 3.3 a planning framework, which can assist in offline plan creation, when the developer would like to specify certain situations. Moreover, solving online planning problems in soft real-time using a "best effort planner" will be described in Sect. 4, followed by the conclusion in Sect. 5.

2 Related Work

The international planning competition (IPC) takes place every year, where newly developed planners evaluate hard planning problems. However, PDDL is not suitable for periodic world state changes.

The basic idea for the development of PDDL [3] was to define a common interface to describe this problem class. PDDL defines a language to describe the existing world, actions to execute by agents and the goal state.

Brenner and Nebel introduced the language MAPL [1]. The article describes a continual planning algorithm realized with MAPL. For the proof-of-concept, Brenner and Nebel evaluate MAPL in the grid world domain. A small team consisting of four robots must find their position on the grid, which is done in simulation.

Ulusar [11] shows an approach for real-time planning evaluated in RoboCup[2] simulation while using PDDL. This approach is a high-performance problem solver, but the main difference with our approach is that it is not aiming at real-time planning. The author uses a training phase for caching to select a suitable plan in certain situations. Furthermore, the cached plans are for a fixed team size.

McCoy et al. [6] show an approach in a real-time scenario in the gaming branch. This approach is developed specifically for a certain game, and they use a multiagent approach that is not cooperative. Every agent acts best for himself.

Kumar et al. [5] realize a multiagent planning using probabilistic inference. This is achieved by constructing a graphical model using likelihood maximization, which is equivalent to plan optimization. The execution time of this approach is high. Furthermore, the approach does not consider the communication performance.

RAPID [2] is an anytime planner for imprecisely-sensed domains. It performs well on a large problem simulation with a lot of state variables. However, Brunskill et al. do not evaluate RAPID in robotic or multiagent domains.

Russel et al. [7] introduce an anytime algorithm, whose quality of results improves as computation time increases. It provides useful performance components for time-critical planning and control of robotic systems. The approach does not handle sensor noise and is evaluated in simulation.

3 Design

In this section, we will sketch first the basics of ALICA, which offers the foundations for the description of multiagent plans and the planning framework. Afterward, the document describes the expansion of ALICA for defining planning problems. Finally, we will describe the planning framework.

3.1 Overview of ALICA

ALICA provides modeling facilities for cooperative behaviors within clear operational semantics. Furthermore, the language can be used in highly dynamic domains

[2] www.robocup.org.

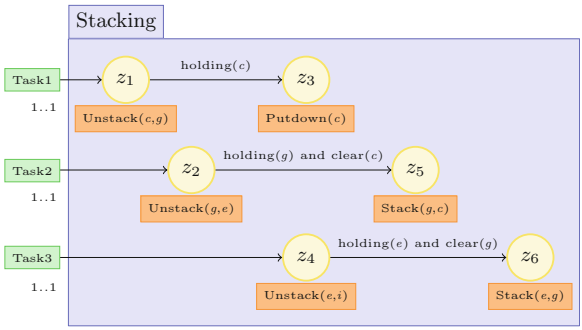


Fig. 1 Example ALICA plan

[8]; therefore, this paper does not focus on the coordination of the plan, but rather on its generation and execution.

The core elements of the language are:

- Basics:
  - $\mathcal{L}$: With language $\mathcal{L}(Pred, Func)$ meant to describe the agents' belief with a set of predicates $Pred$ and a set of function symbols $Func$
  - $\mathcal{R}$: Set of roles, which the agent can take
  - $\mathcal{B}$: Set of atomic behaviors, which can interact with the environment
  - $\mathcal{P}$: Set of cooperative behavior description
  - $\mathcal{P}_\vee$: Set of alternative plans
  - $\mathcal{T}$: Every task describes a function in a plan
  - $\mathcal{Z}$: States exists in plans and represents a step in this plan
- Plan elements:
  - Behaviors—atomic single-agent action programs: $\mathcal{Z} \mapsto 2^{\mathcal{B}}$
  - Plans—abstract multiagent activity descriptions defined by: $\mathcal{P} \mapsto 2^{\mathcal{Z}}$
  - Plantypes—sets of alternative plans defined by PlanTypes: $\mathcal{Z} \mapsto 2^{\mathcal{P}_\vee}$
  - Tasks—denote specific activities within plans defined by Tasks: $\mathcal{P} \mapsto 2^{\mathcal{T}}$
  - Roles—descriptions of capabilities
- Conditional elements:
  - Pre—denotes a pre-condition of a behavior or plan defined by Pre: $\mathcal{P} \cup \mathcal{B} \mapsto \mathcal{L}$
  - Run—denotes a runtime condition of a behavior or plan defined by Run: $\mathcal{P} \cup \mathcal{B} \mapsto \mathcal{L}$
  - Post—denotes a post-condition of a behavior or plan defined by Post: $\mathcal{Z} \mapsto \mathcal{L}$

Figure 1 illustrates an example ALICA plan using the core elements of the language. This figure shows an example of the blocks world domain.[3] We defined roles $\mathcal{R}$ which are suitable for the task $\mathcal{T}$ (Task 1–3) dependent on the robot capabilities. Every agent in the team can be assigned to one of the tasks with respect to the

[3] http://ipc.icaps-conference.org.

minimum and maximum cardinalities 1..1 and the precondition. The "Stacking" plan $\mathcal{P}_s$ contains a state machine for every agent in the team with several states $\mathcal{Z}$ (yellow circle). Every state contains a plan (orange boxes), which contains a state machine of basic behaviors $\mathcal{B}$. These plans represent the basic skills of the agents. The agents can switch states with conditional transitions. The plan shows how to build a stack of c, g and e.

3.2 Expansions of ALICA

ALICA supports the description of multiagent plans, which a team of collaborative robots can execute. The main problem is that the plan has to model every possible world state or otherwise the team cannot act in unanticipated situations. Therefore, ALICA needs some expansions for integrating a planner.

$$\begin{aligned} \text{ProblemElement}&: \mathcal{O} \mapsto 2^{\mathcal{P} \cup \mathcal{B}} \\ \text{UpdateTime}&: \mathcal{O} \mapsto \mathbb{R} \\ \text{AlternativePlan}&: \mathcal{O} \mapsto \mathcal{P} \\ \text{WaitPlan}&: \mathcal{O} \mapsto \mathcal{P} \\ \text{World}&: \mathcal{O} \mapsto \text{Pre} \\ \text{Goal}&: \mathcal{O} \mapsto \text{Post} \end{aligned}$$

A ProblemElement defines a set of Plans $\mathcal{P}$, or Behaviors $\mathcal{B}$, which represent single actions. The *UpdateTime* represents the soft real-time boundary for the planner. pRoPhEt MAS, at a frequency of the *UpdateTime*, will check if there exists some possible actions and may update the actual plan. Until the first plan is available, the team executes the *WaitPlan*. This can be used for positioning or setup for the planning problem. If the planner fails, the team will execute the *AlternativePlan* as a fallback plan. Finally, the planning problem contains a world and goal description. ALICA uses conditions that were manually coded beforehand. The world will update automatic in runtime. However, world can be defined to solve planning problems in offline mode.

3.3 Planning Framework

We begin by explaining how to use planning problems $\mathcal{O}$. Figure 2 shows an example for a blocks world problem. At the beginning, it is possible to start and stop the team's behavior. This plan illustrates that the team of agents have to search trough all the stacks. We explicitly modeled a plan to solve this problem. The "StackProblem" is defined as the planning problem, which may contain basic skills like "Stack",

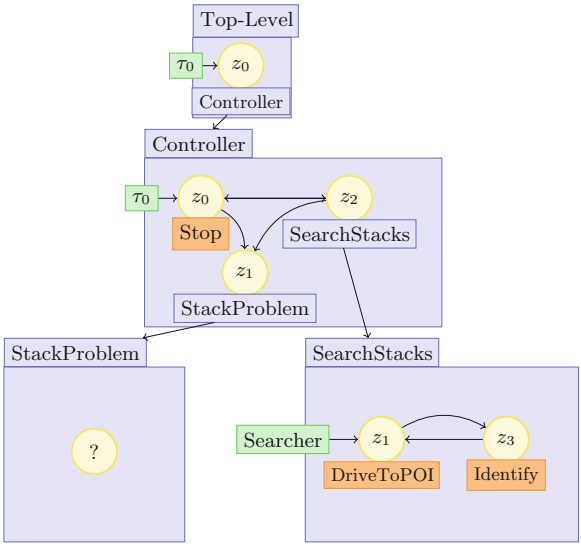


Fig. 2 Example plan for an online stacking problem

"Unstack", "Pickup" and "Putdown"; therefore, the engine tries to find a plan at runtime. pRoPhEt MAS (Reactive Planning Engine for Multiagent Systems) updates the plans dependent on the "UpdateTime." The integrated planning engine searches a valid sequence of actions of the "ProblemElement" which satisfies the goal. Every root plan must span an acyclic "plan tree" like in Fig. 2. In every dynamic situation, we can define a planning problem instead of a static plan.

pRoPhEt MAS The planning framework is shown in Fig. 3. *World* and *ALICA-Engine* are the basic components of the framework. The *WorldModel* contains the information about the robots' environment in first-order logic. ALICA selects a suitable plan from the *PlanBase* which is modeled by the developer. ALICA reacts to environment changes, and actively selects new plans. If a planning problem $\mathcal{O}$ is defined, the engine will use a planning system for creating a suitable plan. (Of course, planning problems can be solved offline as well.) The plan generation is handled by a central leader to reduce the amount of communication. The leader is selected by the team implicitly, while sharing team information periodically via the *ISharing* component. Currently, the robot with the lowest ID becomes the leader. The leader starts the computation to solve the planning problem in due consideration of *WorldModel*. If the leader gets the first results by the planning system, he will share the result with his members after the plan has been validated by the *Validation* component. The team will start to execute the previously existing plan. If the team receives new plan updates, tasks may change with the former plan. However, ALICA is able to handle switching tasks by the task allocation.

pRoPhEt MAS observes the execution process. If the steps of the planning system do not fulfill the goal, the plan will fail. Every robot will move to the *WaitPlan*. Finally, the leader restarts the planning process.

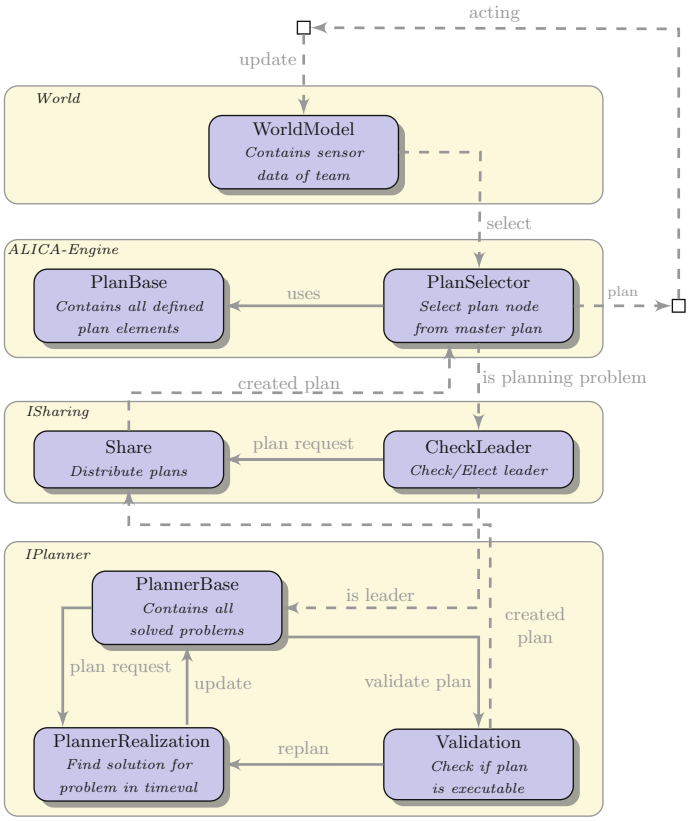


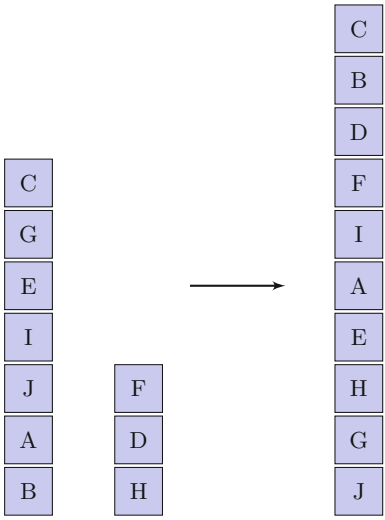
Fig. 3 Planning framework

The architecture of pRoPhEt MAS is modular. It is possible for *IPlanner* to exchange every PDDL-based planner easily. Furthermore, it is possible to define planning problems in every desired state so we can choose a specific planning system suitable for the specific planning problem.

4 Evaluation

The evaluation focuses on a scenario from the blocks world domain since the blocks world offers a test bed where the state space increases very quickly. Besides, the planning results can be understood and easily analyze. The situation of the scenario is shown in Fig. 4. The blocks on the left represent the current situation, which is defined in a planning problem as *World*. The blocks on the right represent the situation the team has to reach, which is defined in a planning problem as *Goal*. In *WaitPlan*

Fig. 4 Scenario description



every agent drives to a free position for not disturbing other tasks. *AlternativePlan* is not defined, so we assume to find a suitable plan.

The goal of the experiment is to compare the execution time for different team sizes of the generated multiagent plan. The plan is calculated offline, and the team starts to execute this plan afterward. We repeat the same experiment online with different *UpdateTimes*, after the first solution is available, the team starts to execute the plan. Every experiment is executed 1000 times, where the average execution times are shown in Fig. 5.

For this scenario, we use three different parameters with or without update time, team size, and update time. For the simulation, we assume that every action takes between 6 and 8 seconds. The Fast Downward Planning System takes 9.5 s on average over all trials to solve the problem from Fig. 4.

In Fig. 5 is the trial without update rate compared to the update rates from 1 to 8 s. The results show that if the update time is smaller, the team saves more of the planning time, which is included in the curve with no update rate. But it is possible that the team executes an action which is not optimal. As a result, the team has to do some more actions. This effect increases if more robots are available, which is slightly visible. The distance of the curves with update rate to the curve with no update rate increases with the number of agents. If more robots are available, the probability that the team executes a suboptimal action increases. For this reason, the plan can become invalid, and the systems will plan from the actual world situation again. If the update time is higher, the probability that the team executes suboptimal actions decrease, but the team needs more time while waiting on the first results of pRoPhEt MAS.

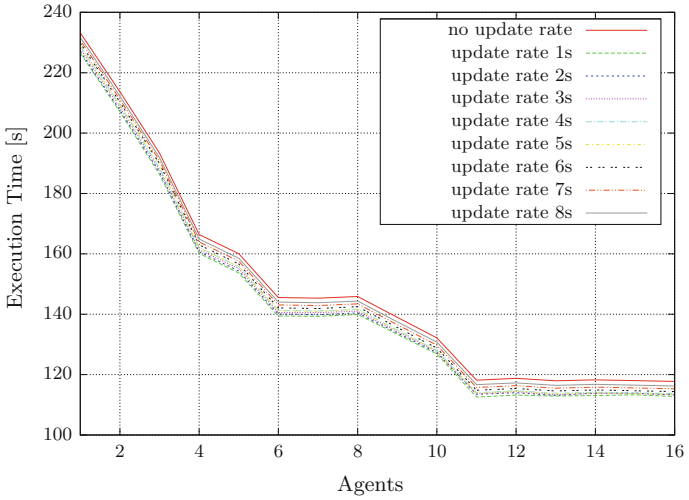


Fig. 5 Average execution time of the experiment

Furthermore, we execute the same scenario on real robot systems. The team consists of up to five robots (surveyor (SRV1)[4]). Every robot is equipped with a marker. This marker is used to localize every agent with the localization of the "Robocup small size league" [12]. In addition, every stack is detected by a marker. To reduce the overhead for grasping, we project the stacks to 2D; therefore, if robot like to manipulate stacks, it will push them across the ground. In our scenario, we use five robots in the first experiment and an *UpdateTime* of one second. Next, to realize a dynamic environment we use different agent as opponents. They are able to select a random action, where no other robot is blocking this stack at this point. The results are shown in Fig. 6. Without opponents, the total execution time decreases with the number of agents. However, the time decreases lesser compare to the simulation. The robots have to drive slower for a precise position because of sensor noise. Furthermore, the execution time is higher. Since, the obstacle avoidance takes more time. With opponents, the time increases. pRoPhEt MAS have to plan again. However, the select actions of the opponents are random. Therefore, sometimes they help to reach the goal. At the end of the scenario, the opponents often were not able to act, because the stack was blocked by an agent.

5 Conclusion

Autonomous mobile robots can improve their effectiveness in dynamic environments using dynamic planning. If the team obeys a predefined static behavior structure, the team will not able to reach the goal if the plan does not covers all possible situations.

[4] http://www.surveyor.com.

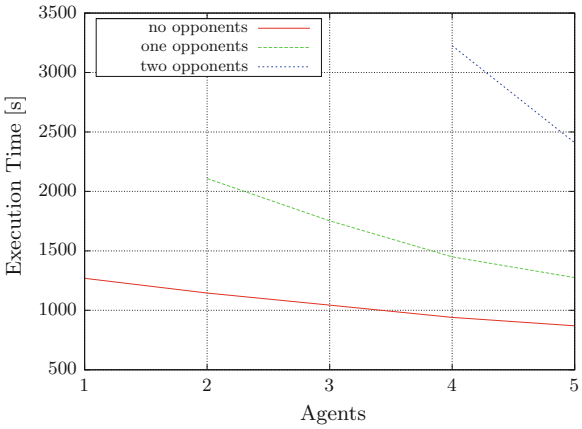


Fig. 6 Execution time of the SRV1 surveyor

Thus, the behavior of the team must be completely described by the developer. This takes a lot of time for complex scenarios, and is difficult to maintain.

For a more flexible approach, it becomes necessary to integrate a planning engine capable of generating a plan for the agent team based on a goal description. It is not suitable to explicitly declare the procedure to reach the goal. Rather, the team is responsible for creating the plan that contains the behaviors necessary to reach the goal state. On one hand, the plans can be generated offline as a support for plan creation. On the other hand, the planning must also be performed at runtime to react on environment changes. The planner has to offer solutions or first possible steps in soft real time.

Thus, the goal is to integrate a generic interface for the integration of planning engines, which is capable of reacting in soft real time. As a first step, we expand ALICA by adding a new element that describes planning problems. Accordingly, ALICA needs an interface for planning systems. Finally, a planning system must be designed to solve more complex problems at runtime. pRoPhEt MAS uses a modified Fast Downward Planning System, which is expanded to provide an observation of the search process. Hence, pRoPhEt MAS is able to select optimistic steps to execute. We simulated different team sizes and different update rates. The results from Fig. 5 shows that planning in runtime can be more efficient than sequential planning and execution. Finally, we evaluate the same scenario on real robot systems shown in Fig. 6. The team of robots was able to solve the problem dynamically while working against opponents.

The next steps are to concentrate on a plan repair mechanism. Currently, we plan again if the team is not longer able to reach the goal. Planning and replanning increases the communication. As a result, an improvement would be to evaluate the rate of coordination of the team as in [9]. Furthermore, we will distribute the planning process to all members of the team in our future work.

Acknowledgments The project IMPERA is funded by the German Space Agency (DLR, Grant number: 50RA1112) with federal funds of the Federal Ministry of Economics and Technology (BMWi) in accordance with the parliamentary resolution of the German Parliament.

References

1. Brenner, M., Nebel, B.: Continual planning and acting in dynamic multiagent environments. Autonomous Agents and Multi-Agent Systems, 19(3):297–331 (2009)
2. Brunskill, Emma and Russell, Stuart J.: RAPID: A Reachable Anytime Planner for Imprecisely-sensed Domains. CoRR (2012)
3. Ghallab, M., Isi, C. K., Penberthy, S. Smith, D. E., Sun, Y., and Weld, D.: PDDL - The Planning Domain Definition Language. Technical report, CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control (1998)
4. Helmert, M.:*The Fast Downward Planning System*. Journal of Artificial Intelligence Research 26 (2006)
5. Kumar, A., Zilberstein, S., and Toussaint, M.: Scalable Multiagent Planning Using Probabilistic Inference
6. McCoy, J. and Mateas, M. An integrated agent for playing real-time strategy games. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 3*, AAAI'08, pages 1313–1318. AAAI Press (2008)
7. Russell, S. J. and Zilberstein, S.: Anytime sensing, planning, and action: A practical model for robot control, pages 1402–1407. Proceedings of the International Conference on Artificial Intelligence (1993)
8. Skubch, H., Wagner, M., Reichle, R., and Geihs, K.: A modelling language for cooperative plans in highly dynamic domains. *Mechatronics*, 21:423–433 (2011)
9. Skubch, H., Wagner, M., Reichle, R., Triller, S., and Geihs, K.: Towards a comprehensive teamwork model for highly dynamic domains. In Filipe, J., Fred, A., and Sharp, B., editors, *Proceedings of the 2nd International Conference on Agents and Artificial Intelligence*, volume 2, page 121–127. INSTICC Press, INSTICC Press (2010)
10. Skubch, H.:*Modelling and Controlling of Behaviour for Autonomous Mobile Robots*. Westdeutscher Verlag GmbH (2013)
11. Ulusar, U. D.: Design and implementation of a real time planning system for autonomous robots. volume 1, page 74–79. Industrial Electronics, IEEE International (2006)
12. Zickler, S., Laue, T., Birbach, O., Wongphati, M., and Veloso, M. M. Ssl-vision: The shared vision system for the robocup small size league. In Baltes, J., Lagoudakis, M. G., Naruse, T., and Ghidary, S. S., editors, *RoboCup*, volume 5949 of *Lecture Notes in Computer Science*, pages 425–436. Springer (2009)