AllemaniACs 2003 Team Description

Frank Dylla, Alexander Ferrein, and Gerhard Lakemeyer

Knowledge Based Systems Group
RWTH Aachen,
Aachen, Germany
{dylla, ferrein, gerhard}@cs.rwth-aachen.de

Abstract. This paper describes the scientific goals of the AllemaniACs 2003 Mid-Size soccer team. We present our robot platform and sketch out our software system. We show some features of the high-level robot programming language GOLOG which we use for modeling the behavior of our robot.

1 Introduction

RoboCup's Mid-Size league has high requirements to system designers. Researchers focus on the fields of sensors and actuators, computer vision, communication, and system integration, just to mention a few, under the strict real-time constraints given. Another issue is the decision making process of a robot and how a team of robots can be coordinated, how cooperative team-play can be achieved. Thus, the robot should not only think about its own actions but also take teammates into account when making plans for future actions. Therefore, the decision module of the robot must be able to reason about actions and their effects and project or simulate them into the future. For our decision making module we use a variant of GOLOG, a logic-based programming language which combines explicit agent programming as in imperative languages with the possibility to reason about actions and their effects.

2 Robot Hardware

Regarding the experiences of other teams that "off-the-shelf" robots must be completely reengineered for RoboCup's Mid-Size to be competitive, we decided to develop the platform on our own [1]. The intention was to develop robots competitive in RoboCup which can also be used in the office domains of service robotics applications. The platform has a size of 39 cm $\times$ 39 cm $\times$ 40 cm (Fig. 1). For power supply we have two 12 V lead-gel accumulators with 15 Ah each onboard. The battery power lasts for approximately one hour at full charge. The robot has a differential drive, the motors have a total power of 2.4 kW. This power provides us with a top speed of 3 m/s and 1000°/s by a total weight of approximately 50 kg.

Fig. 1. A scene from the Championships in Padua; AllemaniACs in the defense (right-hand side)

Onboard we have two Pentium III PC's at 933 MHz running Linux, one equipped with a framegrabber for a Sony EVI-D100P camera mounted on a pan/tilt unit. Our other sensor is a 360° laser range finder with a resolution of 0.75 degree at a frequency of 20 Hz. For communication a WLAN adapter based on IEEE 802.11b is installed.

3 Robot Software

Our software system consists of a collision avoidance, a self-localization, a mapping, a vision, a global world model, and a high-level decision module. For interprocess communication we use a blackboard communicating via shared memory internally. For communication between different computers the blackboard uses UDP. The collision avoidance module uses the 360° laser range finder. We use the A^* algorithm to calculate collision free paths to a target point every 50 ms. Localization is based on the fusion of the Monte Carlo approach using the laser range finder and a vision algorithm using probability maps [2]. Except for symmetry the robots are able to localize with the laser sensor on the field within a few seconds even if crowded with other robots . Using vision information like the distance to the yellow goal, symmetry can be resolved. The world model provides global position information about the robots and the ball. Furthermore, data about the current state of the game, e.g. game is interrupted or before kick-off, and the resp. roles the robots fill, are stored here.

4 The agent specification language GOLOG

For specifying our high-level control we use a variant of the logic-based high-level agent programming language GOLOG [3]. GOLOG is a language based

on the situation calculus [4]. Over the past years many extensions like dealing with concurrency, exogenous and sensing action, a continuous changing world and probabilistic projections (simulation) [5,6,7] made GOLOG an expressive robot programming language. We integrated those features in our ICPGOLOG interpreter [8]. For the decision making, we furhter integrated a planning module into GOLOG which chooses the best action to perform by solving a Markov Decision Process (MDP) (we refer to [9] for reading on MDP and to [10] on integrating MDPs into GOLOG).

In the following we give an overview of some of the features of ICPGOLOG not going into details. For an extended example of how a multi-agent plan for a double pass can be modeled and executed in the SIMULATION league in ICPGOLOG we refer to [8]. The ICPGOLOG language features are:

- Sequence: a_1, a_2
- Nondeterministic Choice: $a_1; a_2$
- Solve an MDP: $solve(p, h)$, where p is a GOLOG program and h is the horizon up to which the MDP is solved
- Test: $?(c)$
- Event-Interrupt: $waitFor(c)$
- If-then-else: $if(c, a_1, a_2)$
- While-loops: $while(c, a_1)$
- Condition-bounded execution: $withCtrl(c, a_1)$
- Concurrent actions: $pconc(a_1, a_2)$
- Probabilistic actions: $prob(val_{prob}, a_1, a_2)$
- Probabilistic (offline) projection: $pproj(c, a_1)$
- Procedures: $proc(name(parameters), body)$

For modeling our high-level agent we have to specify the primitive actions the robot can perform together with their preconditions and effects. These are actions like $goto(x, y, \theta)$, $intercept\_ball$, or $dribble\_to(x, y)$. Besides primitive action there are exogenous and sensing actions. With exogenous actions external events, e.g. a game restart, can be modeled. With the *pconc*-statement programs can be executed concurrently, the *pproj*-statement starts a simulation of a ICPGOLOG program. The robot can simulate several different alternatives which actions to perform and the choose the most promising. Another important feature is the possibility to model uncertainty by the *prob*-statement. It is possible to model for example the success of a *goto* action or even build a model for the behavior of the opponent goal-keeper. The *solve* statement initiates the solving of the MDP formulated by the program p up to a fixed horozon h. It returns a policy, i.e. an optimal course of action given p the robot can perform regarding the current game situation.

The offense's behavior of the AllemaniACs robots were totally determined by the MPD-like planning of our GOLOG dialect. The robot made plans of length four including teammate's and opponent's behavior. In total a number of $\approx$ 900 plans were made per agent with an average computation time of 0.3 seconds over the 8 games we played at the RoboCup 2003.

References

1. Wunderlich, J.: Technical description of the AllemaniACs soccer robots. Technical report, LTI / KBSG, Aachen University, Germany (2002) in German.
2. Jones, M.J., Rehg, J.M.: Statistical color models with application to skin detection. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition. Volume 1. (1999) 274–280
3. Levesque, H.J., Reiter, R., Lesperance, Y., Lin, F., Scherl, R.B.: GOLOG: A logic programming language for dynamic domains. Journal of Logic Programming **31** (1997) 59–83
4. Reiter, R.: Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press (2001)
5. Giacomo, G.D., Lésperance, Y., Levesque, H.J.: ConGolog, A concurrent programming language based on situation calculus. Artificial Intelligence **121** (2000) 109–169
6. Grosskreutz, H., Lakemeyer, G.: cc-Golog: Towards more realistic logic-based robot controllers. In: Proceedings of the 7th Conference on Artificial Intelligence (AAAI-00) and of the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-00), Menlo Park, CA, AAAI Press (2000) 476–482
7. Grosskreutz, H.: Probabilistic projection and belief update in the pGolog framework. In: Second International Cognitive Robotics Workshop. (2000)
8. Dylla, F., Ferrein, A., Lakemeyer, G.: Specifying multirobot coordination in ICP-Golog – from simulation towards real robots. In: Proc. of the Workshop on Issues in Designing Physical Agents for Dynamic Real-Time Environments: World modeling, planning, learning, and communicating (IJCAI 03). (2003)
9. Puterman, M.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley (1994)
10. Boutilier, C., Reiter, R., Soutchanski, M., Thrun, S.: Decision-theoretic, high-level agent programming in the situation calculus. In: AAAI'2000. (2000)