GermanTeam 2001

Ronnie Brunn[3], Uwe Düffert[1], Matthias Jüngel[1], Tim Laue[2], Martin Lötzsch[1], Sebastian Petters[3], Max Risler[3], Thomas Röfer[2], Kai Spiess[2], and Andreas Sztybryc[2]

[1] Institut für Informatik, LFG Künstliche Intelligenz, Humboldt-Universität zu Berlin, Rudower Chaussee 25, 12489 Berlin, Germany
[2] Bremer Institut für Sichere Systeme, TZI, FB 3 Mathematik/Informatik, Universität Bremen, Postfach 330440, 28334 Bremen, Germany
[3] Fachgebiet Simulation und Systemoptimierung, FB 20 Informatik, Technische Universität Darmstadt, Alexanderstr. 10, 64283 Darmstadt, Germany

1 Introduction

The GermanTeam is the successor of the Humboldt Heroes who already participated in the Sony Legged Robot League competitions in 1999 and 2000. Because of the strong interest of other German universities, in March 2001, the GermanTeam was founded. It consists of students and researchers of five universities: Humboldt-Universität zu Berlin, Universität Bremen, Technische Universität Darmstadt, Universität Dortmund, and Freie Universität Berlin. However, for the system presented in this document, the Humbold Heroes only had reinforcements from Bremen and Darmstadt. The two other universities will actively participate with the beginning of the winter semester.

2 Software Architecture

The basic concept of the GermanTeam is a shared memory architecture. In the shared memory, several data structures are stored for exchanging information between the different processes on the robots. The control software consists of several objects, which are running in parallel. The most important ones are:

HSensor reads data from the robot's sensors.
HMotion sends move commands to the robot.
HIntegra builds up the world model. Apart from image processing, the recognition of the ball, other robots, landmarks, and the self-localization are performed in this process.
HControl is responsible for the robot's behavior.

Due to the encapsulation of all robot specific functions in *HSensor* and *HMotion*, other processes may be also be run on a PC under the debugging tool *DogControl*. DogControl also offers many functions for visualization and communication, e.g., for showing the results of the image segmentation algorithm, or for controlling the robot from a PC.

A. Birk, S. Coradeschi, and S. Tadokoro (Eds.): RoboCup 2001, LNAI 2377, pp. 705–708, 2002.

3 Motion

In contrast to other RoboCup leagues, motion is still a research topic in the Sony Legged Robot League, because it means *walking*. The GermanTeam implemented a motion net that manages the transitions between different motions, and that ensures the *completeness* of these transitions. The motion net allows to combine all available motions, and it eases the development of new ones. As most simple motions (like kicking or standing up) can be defined by sequences of joint data vectors, a *motion description language* was developed, in which all motions are specified. For more complex motions, the description language provides commands to execute native C code functions. That way, all gaits are integrated.

4 Image Processing

The vision system processes YUV-images with a resolution of 176×144 pixels acquired by the robot's camera. The processing starts with a color segmentation. The color of every pixel is assigned to one of eight color classes, e.g. the orange of a ball, the yellow of a goal and some landmarks, or the green of the field and some landmarks. This assignment is performed by using a lookup table, the so-called color table. The color table assigns each color from a $64 \times 64 \times 64$ YUV color space to a color class by using the YUV-intensities as indices into the lookup table.

While segmenting an image, it is also run-length encoded to simplify the construction of the so-called *blobs*, and to compress the data. A blob is a data structure representing a region of connected runs of the same color class. For each region, eight characteristic points are determined that represent the maximum expansion of the blob in vertical, horizontal, and diagonal directions. The employed algorithm to calculate the blobs is based on a state machine that directly operates on the runs. The method follows the boundary of each eight-connected region, and was originally developed by Quek [3].

5 Object Recognition

The recognition of objects searches for areas or groups of areas with special shapes, colors, and positions. Before the object recognition takes place, the influences resulting from the three degrees of freedom of the robot's head are removed. Then, Allen's [1] interval calculus is employed to describe spatial relations between blobs. A goal is a yellow or sky blue rectangle lying above a green area, i.e. the carpet. A flag consists of a pink rectangle and a second green, yellow or sky blue rectangle above or below that has nearly the same size. The ball is an orange octagon with at least three of its corners lying on a circle. Its position is determined from intersecting the middle-perpendiculars, which allows to calculate the center of the ball even if most of it lies outside the image. The distances to objects are calculated from their sizes in the image. The directions

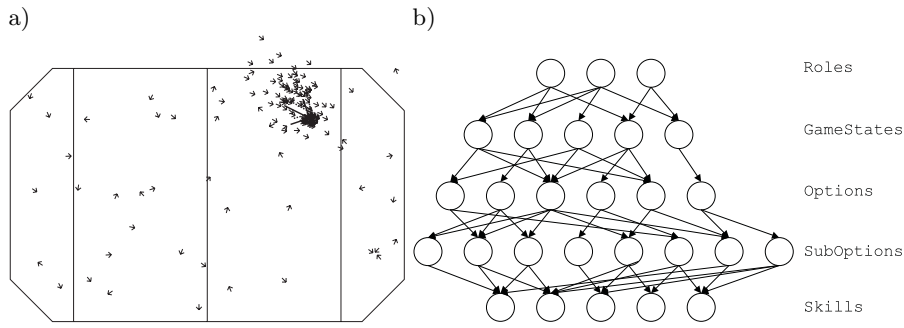

Fig. 1. a) Distribution of samples during Monte-Carlo localization. The large dotted arrow marks the real position of the robot, the solid arrow marks the estimated location. b) A five layer state machine for behavior. The circles represent behavior states.

to objects are determined from their positions in the image, and from the pose of the head of the robot, i.e. the field of view of the camera.

6 Self-Localization

The GermanTeam used a Markov-localization technique employing the so-called Monte-Carlo approach [2]. It is a probabilistic method, in which the current location of the robot is modeled as the density of a set of particles (cf. Fig. 1a). Each particle can be seen as the hypothesis of the robot being located at its position. Therefore, such particles mainly consist of a robot pose, i.e. a vector representing the robot's Cartesian coordinates and its rotation.

The localization approach works as follows: first, all particles are moved according to the *motion model* of the previous action of the robot. Then, the probabilities for all particles are determined on the basis of the *observation model* for the current sensor readings, i.e. bearings on landmarks calculated from the actual camera image. Based on these probabilities, the so-called *resampling* is performed, i.e. moving some of the particles to the location of the sample with the highest probability. Afterwards, the average of the probability distribution is determined, representing the best estimation of the current robot pose. Finally, the process repeats from the beginning.

7 Behavior

The GermanTeam used a *multi layer state machines* architecture to control the behavior of the robot. In this approach, each behavior is a state that is performed until the active state itself decides that another state should be executed. For instance, a state "kicking the ball to the goal" is executed until the ball is not seen anymore (the next state would be "search ball"), or the ball is to far away

(the subsequent state would be "go to ball"). The underlying assumption is that the different behaviors themselves know best whether they should continue their execution, or if not, which behavior is a useful successor.

In a soccer game there are very different types of decisions that have to be made: On the one hand, there are very global and long-term behavior states such as "playing the ball", "waiting for a pass", "returning to the own goal", "waiting for the kick off", etc. that do not change very often. One the other hand, there are short-term reactive behaviors such as "turning left around the ball", "positioning in the center of the own goal", and "passing the ball sideward" that can frequently alternate.

To be able to make both long-term and short-term decisions, the different behavior states were distributed over five layers (cf. Fig 1b):

Roles define the general behavior of a robot during the whole game, e.g. "player1" or "goalie", but also "defensiveGoalie" or "offensiveGoalie".

Game-States are general states of the robot such as "waiting for a kickoff", "waiting for a pass", "searching the ball", "playing the ball", etc.

Options are long-term intentions, plans such as "passing the ball", "kicking the ball to the goal", "return to goal", etc.

Sub-Options are short-term intentions, plans such as "kicking the ball forward to the goal", "position in the left center of the own goal", etc.

Skills are basal capabilities, implemented in decision trees. They select gaits or kicks, and they are purely reactive, without internal states. They are parameterized from the sub-options layer.

In each layer, only one state is executed at the same time. The active state will determine the following state for the next lower layer. Only the skills initiate real motions of the robot.

8 Conclusion

Although the GermanTeam did not reach the quarter final, the fourth place in the challenge showed that many parts of the system are quite promising. For RoboCup 2002, the GermanTeam will replace the shared memory architecture by a more flexible communication scheme, and will revise all modules. Some parts of the functionality, i.e. the behavior architecture, will be formalized.

References

1. J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26:832–843, 1983.
2. D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo localization: Efficient position estimation for mobile robots. In *Proc. of the National Conference on Artificial Intelligence*, 1999.
3. F. K. H. Quek. An algorithm for the rapid computation of boundaries of run length encoded regions. *Pattern Recognition Journal*, 33:1637–1649, 2000.