Salzburg Interest Group on Integrated Systems

siGis-22-2001 August 31, 2001

EMMA – Architecture and Subsystems of a Mobile Autonomous Robot with a Preference for Soccer

2001 WSES International Conference on
Robotics, Distance Learning and Intelligent Communication Systems (RODLICS 2001)
September, 1-6, 2001, Malta

Helmut A. Mayer helmut@cosy.sbg.ac.at

Josef Schmidbauer jschmid@cosy.sbg.ac.at

Gerald Stieglbauer gstiegl@cosy.sbg.ac.at

Department of Computer Science
University of Salzburg

Correspondence to:

Helmut Mayer
Universität Salzburg
Institut für Computerwissenschaften
Jakob–Haringer–Straße 2
A–5020 Salzburg
AUSTRIA

Telephone: +43-662-8044-6315
FAX: +43-662-8044-611

EMMA – Architecture and Subsystems of a Mobile Autonomous Robot with a Preference for Soccer

HELMUT A. MAYER and JOSEF SCHMIDBAUER and GERALD STIEGLBAUER
Department of Computer Science
University of Salzburg
Jakob–Haringer–Strasse 2, A–5020 Salzburg
AUSTRIA
{helmut}, {jschmid}, {gstiegl}@cosy.sbg.ac.at

Abstract: - In this paper we present the current stage of development of EMMA, a small autonomous robot inspired by the Khepera robot widely distributed in the scientific community. EMMA features a modern microcontroller as the central processing unit (CPU) being very efficient in processing sensor and motor signals and a black–white camera as the main sensory system. The test application of EMMA is autonomous robot soccer, where two robots play against each other in a specified environment without any connection to an external control computer. The control program in this application is based on a fuzzy rule base generating the motor signals according to the information provided by an on–line object recognition analyzing the camera image. We also briefly present a number of software tools being currently developed for EMMA, and conclude with plans for future enhancements to the embedded mobile agent.

Key–Words: - Mobile Autonomous Robots, Robot Architecture, Evolutionary Robotics, Robot Soccer, Fuzzy Controller

1 Introduction

The exponential increase in computational power of microprocessors and storage capacity during the last 20 years paved the way for the construction of small robots able to interact with their environment by means of specific control programs. Naturally, these small robots resemble some of the vital properties of living organisms (Lund et al., 1997). As these properties are shaped by evolutionary processes in nature, which already have been transposed to computer science in the field of *Evolutionary Computation* (EC) a new field – *Evolutionary Robotics* (ER) (Nolfi and Floreano, 2000) – emerged. In ER evolution can be used to optimize hard– and software, most noteably the control program (the "brain") of the robot.

With a background in the evolution of *Artificial Neural Networks* (ANNs) (Huber et al., 1995; Mayer et al., 1999) we wanted to join many researchers investigating the use of evolved neurocontrollers for their robots. Due to limited financial ressources we could not afford to use the (among research institutions) widely distributed Khepera robots (K–Team SA, 1995), so we started to construct our own robot named EMMA. Throughout the development of this robot we focused on robot soccer to test and refine all subsystems. In the following we report on the hardware architecture, the software for robot soccer, a variety of software tools currently being implemented, and conclude with an outlook on future research topics.

2 EMMA Hardware Architecture

The *EMbedded Mobile Agent* (EMMA) is built based on a modular architecture (Figure 1) in order to ease future additions or changes of modules. At present, EMMA (a cylindrical entity with a diameter of 6 cm and a height of 12 cm) consists of four modules:

- Processor Module
- Memory Module
- Camera Module
- Power Module

The heart of the processor module is the Infineon C167 microcontroller – a 16–Bit processor with a CPU speed up to 33 MHz. In our configuration the C167 has a performance of approx. 10 MIPS. Additional hardware (timers, pulse width modulators (PWM), peripheral event controller (PEC)) within the microcontroller provide on–chip circuitries being particular useful for

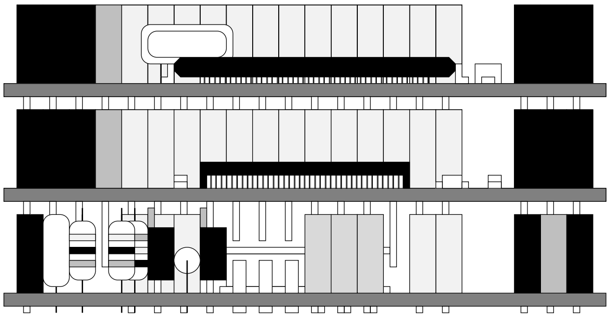

Figure 1: EMMA hardware modules from top to bottom: processor, memory, power (camera module not shown).

robots (e.g., the PWM channels are used for motor control). This high level of integration allows to keep the robot small.

The memory module contains two 8 Bit SRAM memory ICs with 512 kWord each. The resulting 1 MByte of memory is accessed from the processor module using the system bus. The memory bus is organized as demultiplexed 16–Bit bus which requires a lot of I/O ports and wires but saves complex additional logic. It also allows an easy extension to a multi–processor system.

The power module hosts two motors with amplifiers controlling motor speed. Four NiCd accumulators provide a voltage of 4.8V for all modules resulting in a working time of about 30 minutes (with a fully active EMMA).

The camera module contains a Logitech BW QuickCam which is connected with the processor module via the system bus. The camera delivers an 80×60 image with 64 grey levels. The image is transferred into memory each time EMMA has completed processing of the last image. A LED on the camera module can be configured by software to signal internal states of EMMA.

3 Autonomous Robot Soccer

The main application we have been using for the development of EMMA's hard– and software is robot soccer which has attracted enormous interest in recent years due to the Robocup initiative (Robocup, 2000). Two robots are placed in a black rectangular box, where on each front end rectangular pieces of grey paper constitute the goal. The ball is a yellow tennis ball, and the robots wear a cylindrical "uniform" with black and white stripes from top to bottom (Figure 2) [1]. Each of the robots tries to push the ball into the opponent's goal. This setup is close to Robocup's *Small–Sized Soccer Robot League* (F–180), where, however, external vision systems are permitted.

Figure 2: EMMA (with uniform besides), ball and goal in the soccer box.

3.1 Object Recognition

In the following we describe the details of EMMA's vision system (Figure 3). for the soccer task. As the image processing and object recognition is performed on–line, the algorithms have to be as fast (and simple) as possible even at the cost of occasional misclassifications of objects. However, as EMMA is constantly moving, hence receiving different images, only consistent misclassifications in a series of images will affect overall behavior.

The most serious problem for the object recognition are changing light conditions during a soccer game. We have performed a number of experiments to adapt the camera parameters to the light on–line, but there exist game situations, where it is impossible to adjust the camera (having a vision–only robot) correctly. E.g., if EMMA faces the black wall of the soccer box, we cannot decide, if the camera image is too dark, or EMMA simply does not have any objects within its field of sight. Consequently, we perform camera calibration before the game starts. Additionally, the calibration procedure serves a second task, namely to inform EMMA of the goal it is supposed to attack. At game start EMMA has to be positioned in the soccer box such that it sees this goal.

[1] Based on the rules at the First Autonomous Robot Football Tournament, Brighton, 1997.

Calibration starts with eliminating all WHITE (the brightest grey level) pixels in the image (in the case there are no WHITE pixels, it first increases camera sensitivity until there are) by adjusting camera parameters. A second important calibration step is the identification of the *Object Band* which is the interval in the image's histogram defining the range of the grey levels of potential objects. Again, the (grey) goal is used to determine the object band. As the other objects ball and robot are brighter than the goal, we have to determine the brightest grey level being sufficient to detect the goal. In essence, we fix the upper bound of the object band to WHITE - 1 and incrementally decrease the lower bound until the goal can be detected by the object recognition (see below). With this lower bound the object band is determined and used throughout the soccer game, i.e., pixels with grey levels outside the object band are never processed (band pass filter).

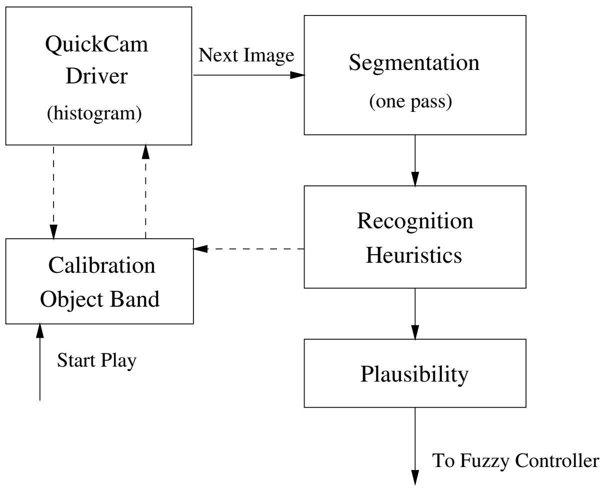

Figure 3: Component view of EMMA's vision system.

With the fixed object band we start scanning each image beginning with the lower, left corner. Each line is scanned to the right until part of a candidate object is found (Figure 4). It is important to note that a single object is not a uniform area in the image, but consists of pixels of many different grey levels due to light reflections, shadows, and camera noise. The decision, if an area of the image is to be considered as a candidate object is based on a noise–tolerant segmentation, where the following parameters play an important role:

- PIXEL_JUMP defines the number of pixels outside the object band which may occur between object band pixels. This parameter simply accounts for noise induced by light and camera. In theory a pixel row with alternating object and non–object pixels would be part of a candidate object (if PIXEL_JUMP > 0). EMMA's current setting (empirically found) is 1.
- MIN_X_PIXELS defines the minimum number of pixels in a line (row) including noisy pixels deemed to be part of a candidate object. Its current setting (empirically found) is 3.
- MIN_Y_PIXELS is the analog in y–direction (columns). Its current setting (empirically found) is 3.

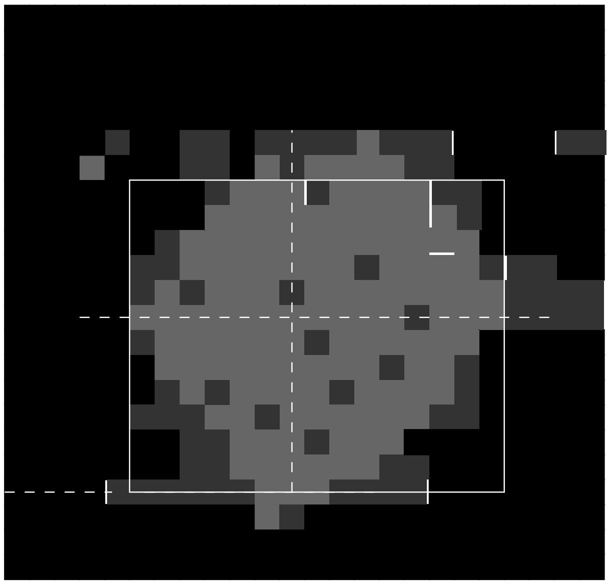

Figure 4: Principle of segmentation into candidate objects (only light grey is in object band, scan lines are dashed, candidate object segment is solid).

If an x–line of a candidate object is detected, the surrounding area is examined. As image processing should be as fast as possible, only a single y–line upwards from the middle of the found x–line is scanned for MIN_Y_PIXELS. If the scan is successful, the area is further scanned from the middle of the y–line in both x–directions, left and right. Overall this results in only four calls to the scan procedure for identifying a candidate object. The height of the object is simply the length of the y–line, while the width is determined by the longer x–line (bottom or middle). The area of the candidate object is then made virtually invisible for the continuing image scan which further reduces processing time. Scanning of the image continues at the lower, right corner of the previously identified candidate object, until the whole image has been scanned.

Segmentation now possibly leaves us with a rather large number of candidate objects as for instance the ball may be divided into a number of different candidate objects due to noise effects and the primitive but fast image scan. Thus, all candidate objects are categorized in a two–stage process.

First, objects are separated based on their position in the image. As the camera is looking down on the ground with an inclination of approx. 30° the ball can never be in the upper part of the image. If an object is within this area, it is either the robot (opponent) or the goal. This is further resolved by the simple fact that the width of the goal is larger than its height, and vice versa (this holds for many situations, but not for all).

As there are only three different objects in our soccer box, we already have categorized them completely. However, there remain a number of problems where the most prominent one is the multiplicity of objects. This is resolved in a second object recognition stage with another simple heuristic. If there are more objects of the same type, the one covering the largest area is selected, while the others are rejected as potential objects. Two special cases are considered additionally, as they constitute important situations of the soccer game, namely, overlap of goal and robot, and goal and ball which in most cases would be merged to a single object, if no special processing took place. Both cases are handled by some more scans of the area covered by the candidate objects. E. g., if EMMA sees the goal, but no ball, we try to locate the ball inside the found goal object.

After this two–stage object recognition at most three (different) objects have been identified. Now, the distance and the angle of the objects relative to EMMA are computed, again using fast approximations. The distance to an object is simply the number of pixels from the bottom of the image to the bottom of the object box. The relative angle of an object is the number of pixels between the middle of the image and the middle of the object box.

Before the information on the detected objects is passed to the fuzzy controller, some simple plausibility checks are performed which may be interpreted as cognitive processes inside EMMA. An important task is the discrimination of the two goals. EMMA always only sees one, so it has to keep track which one it is facing. Basically, if the goal disappears at one side of the image, and reappears on the other side, the heuristic switches the goal status. However, a single misclassification of the goal will likely result in an incorrect assumption about the goal status. Definitely, a magnetic orientation sensor would reliably solve this problem.

3.2 Fuzzy Controller

As outlined above the fuzzy controller receives as its input six parameters from the object recognition output, namely, distance and angle of ball, goal, and robot. As EMMA only is able to see a single goal in an image, the parameters of the other goal are estimated. Actually, the control program is a meta–fuzzy controller (Skarmeta et al., 1999; Lavrov, 1994) organized in an hierarchical manner similar to Brook's (1986) layered architecture (Brooks, 1986) (Figure 5).

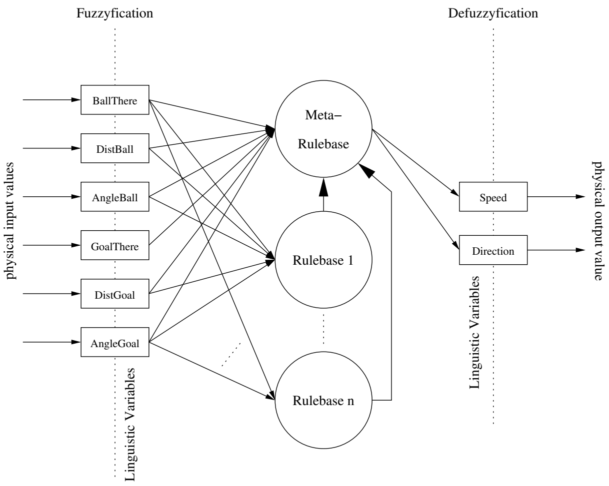

Figure 5: Basic architecture of EMMA's meta–fuzzy rule base.

First, fuzzy controller input is processed by a meta rule base which chooses among a certain number of different basic behaviors of EMMA. Second, the selected behavior rule base generates the actual motor signals. All rule bases have been constructed based on empirical tests on an EMMA soccer simulator or on the real robot, but we have also started to experiment with the artificial evolution of fuzzy rules. Currently, the meta rule base selects between 12 different behaviors, and the behavior rule base contains roughly 200 elementary rules. Some examples for specific behaviors are *Follow Ball*, *Follow Robot* (in case the ball cannot be seen), or *Shoot Goal*. Parts of the rule base for the latter behavior are shown in Table 1. Note that the rules are associated with a weight factor assigning different levels of significance to each rule. The membership functions are mostly of triangular shape, but the fuzzy software environement also processes arbitrary polygons (being used in some cases).

Rule Base Shoot Goal
IF yAngleBallG1 = left AND yAngleBall = left_middle
THEN yDeltaV = left_middle WITH 0.6
IF yAngleBallG1 = null AND yAngleBall = left_middle
THEN yDeltaV = left_middle WITH 0.8
IF yAngleBallG1 = right AND yAngleBall = right_middle
THEN yDeltaV = right_midd le WITH 0.6
IF yAngleBall = left_middle
THEN yV = front_small WITH 0.4
IF yAngleBall = right_small
THEN yV = front_middle WITH 0.5
IF yDistBall = small AND yDistGoal1 = small
THEN yV = front_small WITH 1.0

Table 1: An example of a behavior rule base (only a fraction of the 22 rules is shown).

4 EMMA Software Tools

As mentioned above most of the software development has been carried out by loading each modified program module onto EMMA and watch the behavior of the robot. As this is a very tedious procedure, we have started to build a tool chain for EMMA which should allow very efficient programming and testing of robot control programs before actually sending the program to the robot. The specific tools are

- DiMAS – a Java framework simulating robots, environments, and their interactions
- SOfA - a Java framework for graphical construction of control programs
- a Java byte code compiler converting the programs to C167 native assembler
- a lean operating system kernel for EMMA

The *Distributed Multi Agent System* (DiMAS) supports modelling of arbitrary robots (size, shape, sensors) and environments (currently only simple objects and light sources). Potential control programs may be assigned to the robot(s) and the final behavior of the robot can be analyzed in the simulation (Figure 6).

Salzburg Obsession for Agents (SOfA [2]) assists the programmer in constructing control programs. A control program is viewed as a logical entitity (which can be assembled from libraries, e.g., neurocontrollers) being graphically connected to arbitrary sensors and motors. This tool is laid out to design and implement control programs without writing a single line of actual Java code.

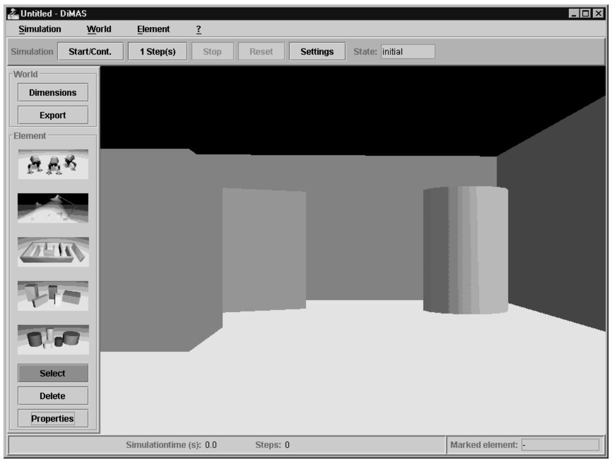

Figure 6: A DiMas screen shot with a world as seen by a robot.

Although, a *Java Virtual Machine* (JVM) is available for the C167 microcontroller, performance considerations mandate to compile the Java byte code of the programs developed with above tools to C167 native assembler. Basically, the compiler translates the single byte code instructions directly to assembler instructions so as to minimize development time (optimization is left to the assembler). For the same reason only a subset of Java native classes will be supported.

At the lowest level of the tool chain is a lean operating system kernel assisting in program transfer to the robot, coordination of multiple tasks, and communication with other devices. Hence, it will be even possible to transfer tasks from one robot to another. Theoretically, a group of robots could then work simultaneously on the same problem (distributed "thinking"), but the operating system just supports the main mechanisms of communication, while coordination of computations is left to the specific programs.

All these tools currently exist in an advanced prototype phase with a subset of the intended functionality implemented. They are steadily improved as part of project courses and diploma theses at our department.

[2]...which actually means *Couch* in German.

5 Summary and Outlook

We have presented hard– and software systems of the mobile autonomous robot EMMA having been constructed at our institution. EMMA's hardware systems are working reliably, but a major problem during development and testing has been the limited working time (30 minutes with full activity) of the robot due to battery recharges. As a consequence, a lot of testing time had to be spent on changing batteries and rebooting EMMA. A supply floor board as presented in (Martinoli et al., 1997) for the Kephera robot, where copper stripes in the floor of a robot arena constantly provide energy, would increase the working time to at least some hours.

The software converting EMMA into a soccer player allows the robot to consistently locate ball and goal, and to push the ball towards the goal. Two main problems are the confusion of goals, and deadlock situations when EMMA is pushing the ball towards the wall. As the two goals in our soccer box are identical, EMMA always has to keep track of which goal it is seeing. A single misclassification of an object as goal leads to confusion (however, the next error will likely resolve the problem). A color marking of the goals used in Robocup would be of little help, as EMMA's object recognition is currently based on object location and shape. A magnetic sensor for EMMA's orientation would completely solve the problem. The deadlock situation could be easily detected by additional bumper sensors, and is avoided completely with Robocup regulations on the playing field, where the field borders are slanted and not perpendicular to the field.

A main line of future research will be the investigation of dynamic neurocontrollers for EMMA's "brain" following recent reports on *Artificial Neural Networks* (ANNs) rearranging themselves during operation. Simulating the mechanisms of biological *Neuromodulators* found in abundance in *Biological Neural Networks* Ishiguro et al. (1999) and Smith and Phillipides (2000) have presented very interesting results using dynamic neurocontrollers for their robots (Ishiguro et al., 1999; Smith and Philippides, 2000). The *Synaptic Plasticity* of the dynamic neurocontrollers might play an important role in autonomous adaptation to environments the robot has never been trained on before.

6 Acknowledgments

The development of EMMA and all its subsystems has only been possible by the contribution of a number of enthusiastic people often having invested their sparetime in countless hours. Thus, we would like to thank Jochen Pfalzgraf and Roland Schwaiger for letting and making all that happen, Wolfgang Brauneis for DiMAS, Ronald Hinterbichler for the electronical hardware, Gerald Katstaller for DiMAS, Andreas Lackner for the mechanical hardware, August Mayer for visualization and testing, Thomas Mirlacher for the software core, Hannes Schmiderer for the fuzzy controller and soccer simulator, Gerald Wiesbauer for the operating system kernel, and last but not least Infineon (Vienna) for supporting hardware devices and software development tools.

References

Brooks, R. A. (1986). A Robust Layered Control System for a Mobile Robot. *IEEE Robotics and Automation*, RA–2(1):14–23.

Huber, R., Mayer, H. A., and Schwaiger, R. (1995). netGEN - A Parallel System Generating Problem-Adapted Topologies of Artificial Neural Networks by means of Genetic Algorithms. In *Beiträge zum 7. Fachgruppentreffen Maschinelles Lernen der GI-Fachgruppe 1.1.3, Forschungsbericht Nr. 580, Dortmund.*

Ishiguro, A., Tokura, S., Kondo, T., Uchikawa, Y., and Eggenberger, P. (1999). Reduction of the Gap between Simulated and Real Environments in Evolutionary Robotics: A Dynamically–Rearranging Neural Network Approach. In *IEEE Systems, Man, and Cybernetics Conference*, pages III – 239–244. IEEE.

K–Team SA (1995). Khepera User Manual. Technical report, K–Team.

Lavrov, A. A. (1994). Modular reconfigurable controllers with fuzzy meta–control. In *IEEE International Conference on Fuzzy Systems*, pages 1564–1567.

Lund, H. H., Webb, B., and Hallam, J. (1997). A Robot Attracted to the Cricket Species Gryllus bimaculatus. In Husbands, P. and Harvey, I., editors, *Proceedings of Fourth European Conference on Artifi-*

cial Life, pages 246–255, MA. MIT Press/Bradford Books.

Martinoli, A., Franzi, E., and Matthey, O. (1997). Towards a Reliable Set–Up for Bio–Inspired Collective Experiments with Real Robots. In *Proceedings of the Fifth International Symposium on Experimental Robotics*, pages 597–608. Springer.

Mayer, H. A., Fürlinger, K., and Strapetz, M. (1999). Extraction of Compact Rule Sets from Evolutionary Designed Artificial Neural Networks. In *IEEE Systems, Man, and Cybernetics Conference*, pages I – 420–424. IEEE.

Nolfi, S. and Floreano, D. (2000). *Evolutionary Robotics – The Biology, Intelligence, and Technology of Self–Organizing Machines*. MIT Press.

Robocup (2000). http://www.RoboCup.org/. WWW Repository, The Robocup Federation.

Skarmeta, A. G., Barberá, H. M., and Alonso, M. S. (1999). A Fuzzy Logic Based Language to Model Autonomous Mobile Robots. In *8th IEEE International Conference on Fuzzy Systems*.

Smith, T. M. C. and Philippides, A. (2000). Nitric Oxide Signalling in Real and Artificial Neural Networks. *British Telecom Technology Journal*, 18(4):140–149.