Soccer Team based on Agent-Oriented Programming

J. Ll. de la Rosa(1), A.Oller(1), J. Vehí(1), J.Puyol(2)
(1) Intelligent Systems and Control Engineering Group
Institute of Informatics and Applications (IIiA)
University of Girona, Catalonia
(2) Artificial Intelligence Research Institute (IIIA)
CSIC, Catalonia

***Keywords*.-** *Agents, CACSD, Soccer*

Abstract- **In this paper the analysis, design and implementation of a soccer team of micro-robots is explained. Besides the technical difficulties to develop these micro-robots, this paper also shows how to develope a multi-agent co-operative system by means of Matlab/Simulink† a widely known Computer Aided Control System Design framework. Agent-Oriented Paradigms formalise interactions between multiple agents in terms of changing their *mental states* by communication between agents. Their practical implementations are usually conceived by means of Object-Oriented Paradigms. Nevertheless, the implementation of Agent-Oriented Paradigms in Matlab/Simulink is not straightforward. Thus, the obtained real implementation is an integrated system that includes several programming paradigms so as hardware platforms. Finally, the proposal of the integrated framework for the micro-robots soccer team is shown.**

1. INTRODUCTION

Multi-agent based mobile robotics require new examples from application and call for new control schemes. By this way, Micro-Robot World Cup Soccer Tournament (MIROSOT) [1] meeting is a forum where rules and constraints about the way that multiple micro-robots could work together in a game is a chance to apply techniques focused on agents. This game is the micro-robotic soccer where 3 micro-robots have to play against another team of the same characteristics and try to score as much as possible.

1.1 MIROSOT

According to MIROSOT specifications, soccer micro-robots are limited to the size 7.5x7.5x7.5cm., and must run autonomously on a 130x90cm. sized ground. MIROSOT rules have strong analogy with real soccer rules although adapted to current robotics technology. Teams can use a centralised vision system to provide robots with *situations* in terms of their position and orientation so as with the ball position. The vision-based facility was widely used by participants to the MIROSOT tournament. The vision-based measures are used by a computer host, here called HOST, to calculate individual robotic movements that are send (broadcasted by an FM emitter) to the robots to be executed. This is depicted in "*Fig. 1*". Thus, since robots have autonomous behaviour but the whole team must actuate as a group, they must co-operate. This means that robots must communicate each other, and the proposal of this work is to deal with a more general problem, that is to do reasoning on communication as agents do. Therefore, Agent-Oriented Paradigm (AOP) frameworks [2] are used in this work.

† MATLAB and SIMULINK are Trade Marks of MathWorks.
Windows 95 is a Trade Mark of Microsoft Corporation.

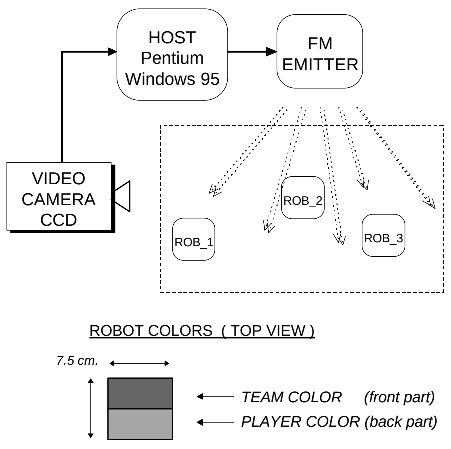


Fig. 1 The Overall System

1.2 AGENT-ORIENTED PARADIGMS AND CONTROL

Nowadays, a commonly proposed solution to the behaviour control in a community of robots is the application of agents theory [3]. The control of the global behaviour of a Soccer Team problem gives the opportunity to apply agents theory due to the distributed architecture of the mobile robots and the problem itself that implies coordination, competition, and co-operation by means of communication among each soccer player and perception. Therefore, the communication of information and knowledge is the most important action that might be taken by each agent, in this case, each soccer player.

On the other hand, control engineering projects are developed as usual in the following steps : analysis, design, and implementation. There are modern computer-based tools that handle the control engineering practice, named Computer Aided Control Systems Design (CACSD) environments. The robotics applications could be more easily developed by using such environments because of the availability of linear and non-linear systems analysis and design tools that could be useful to control robotics movements.

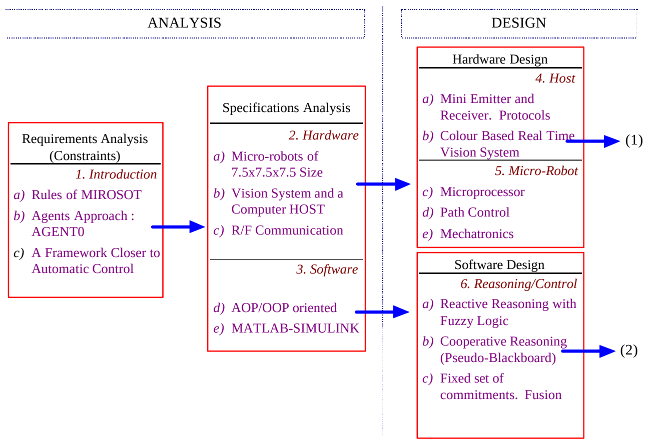
These two approaches are clearly different : the AOP provides us the conceptual model to describe, in a declarative style, the behavior of the team by means of high level constructs and the current CACSD are not good at representing declarative knowledge but procedural. When dealing with complex systems further declaration is fundamental. Therefore, at the moment to solve complex systems control then the distribution of knowledge is essential in addition to automatic control, and the case of the soccer micro-robotics game is specially adequate.

Therefore in this paper the conjunction of AOP and CACSD systems is proposed, becaused the AOP introduces declarative knowledge in terms of communication operators for dealing with decisions, capabilities and commitments of agents that are necessary in this project. But these concepts are, in practice within any control project, difficult to be explicitly used both for the analysis and design phases because of the lack of control-oriented tools that were integrated in agents design, and also because of the lack of Type Data Abstraction (TDA) of CACSD systems. In this work Matlab/Simulink is used because it contains matrices TDA and also is a control oriented programming language. It is specifically designed to help in the three phases of control engineering projects, both as a continuous-time simulation framework for testing the integrated simulated solution and also as a control system for micro-robots.

1.3 *MULTI-AGENTS*

The next basic elements of the multi-agent frameworks can be distinguished : a common environment, agents, interaction between agents, and interactions between agents and the environment. In this work, the environment is the soccer field that is shared by another soccer team that continuously evolves in it, so that it must be considered as a dynamic environment : positions and velocities are continuously changing.

The team is composed by three similar agents, whose behaviour can vary widely according to the instantaneous state of their internal parameters, and depending on the value of two particular fuzzy variables : distance to the ball (DPB) and distance from the goal (DPG). It seems natural that only the close neighbourhood of every agent is of immediate relevance to that agent, so that the size of this neighbourhood is used to set a limit to the DPB variable range. By this way, decisions will be taken with degrees of certainty which can be zero out of the neighbourhood limits.



ANALYSIS
DESIGN
Requirements Analysis (Constraints)
1. Introduction
a) Rules of MIROSOT
b) Agents Approach : AGENT0
c) A Framework Closer to Automatic Control
Specifications Analysis
2. Hardware
a) Micro-robots of 7.5x7.5x7.5 Size
b) Vision System and a Computer HOST
c) R/F Communication
3. Software
d) AOP/OOP oriented
e) MATLAB-SIMULINK
Hardware Design
4. Host
a) Mini Emitter and Receiver. Protocols
b) Colour Based Real Time Vision System
5. Micro-Robot
c) Microprocessor
d) Path Control
e) Mechatronics
(1)
Software Design
6. Reasoning/Control
a) Reactive Reasoning with Fuzzy Logic
b) Cooperative Reasoning (Pseudo-Blackboard)
c) Fixed set of commitments. Fusion
(2)

Fig. 2 Analysis and Design Phases

Every agent includes perception and communication capabilities, so as decision capabilities. Since all the micro-robots of this project have the same technical specifications, various reactive models (behaviour roles, as for instance, defense, attack, and goal keeper) are programmed on them to avoid interfering each other, and to co-operate when they could mutually benefit.

An important capacity of an agent in multi-agent environments is the ability to decide its own actions based on its own goals. However, when considering the relationships between individual goals and community goals two issues arise : how the satisfaction of individual goals affects the satisfaction of community goals, and how the satisfaction of community goals leads to the satisfaction of the individual goals. In this work, the satisfaction of community goals is guaranteed because final decisions are criticised by a fusion procedure, which can be understood as a private action of a new agent so-called *coach-agent* . The role of this agent is to review the set of individual decisions and make some changes only when needed (see section 2.2.3).

When operating in multi-agent environments, intelligent agents must generally co-ordinate their actions ; thus they must communicate the proper knowledge and information. To develop techniques for deciding what to communicate is problematic [4] because it requires every agent to have a model of a message recipient and to infer the impact of a message on the recipient based on that model. In this work, communication primitives consist of a subset of those used in AGENT0 [2], so that timely decision making is intended.

2. AGENT-ORIENTED PROGRAMMING

AOP can be intuitively viewed as an specialisation of the Object-Oriented Programming (OOP). Whereas, on the one hand, the OOP proposes to conceive software applications made of information modules or structures (so-called *objects*) that are able to exchange information between each other and that have individual ways of handling incoming messages by means of the so-called *methods*. On the other hand, AOP specialises the OOP framework by extending the state of the *objects* that now will be considered as *agents* with mental states (which consist of components such as beliefs, capabilities, and decisions). Moreover, the AOP contains further advanced and specialised methods of message passing between agents, as for example : to inform, request, offer, accept, reject, compete, and assist each other.

The declarative knowledge (rules of reasoning) of agents has to be implemented using Matlab/Simulink which does not include object-oriented programming capabilities. Therefore, differences between both paradigms (agents and simulation/control) in the analysis phase are solved by splitting objects and developing reasoning in a sequential order. Thus, reasoning is developed in three steps:

1. Each agent decides its own *reactive* action depending on its position on the ground and the relative situation of the ball. Then, they *inform* to any agent this decision.
2. Each agent decides its *cognitive* action. Agents get new information and take new decisions (co-operative –cognitive– ones) that have higher degree of certainty than the reactive ones [5]. Then, they *inform* to *coach-agent*.
3. Individual decisions of micro-robots are criticised by *coach-agent* and converted into actions by selecting from proposals of the soccer agents.

A simulation programme has been designed to test different reactive models, then any real micro-robots team runs with different co-ordination patterns by changing its own reactive models.

2.1 THE AGENT0

The emergence of a number of prototypical *agent languages* [6][7][8] is one sign that agent technology is more widely used, and that many more agent-based applications are likely to be developed in the near future. One such language is Shoham's AGENT0 system, which is the inspiration of this work. In this language, an agent is specified in terms of sets of:

1. Agent's *capabilities* : actions the agent is able to perform.
2. Initial *beliefs* and *commitments* : logical statements about the world, that the agent believes to be true or false.
3. *Commitment Rules* consist of antecedent conditions that are matched against incoming messages and the agent's internal states.

The key component, which determines how the agent acts, is the commitment rule set. Each commitment rule contains a *message condition*, a *mental condition*, and an *action*. To determine whether any rule fires, the message condition is matched against the messages the agent has received; the mental condition is matched against the beliefs of the agent. If the rule fires, then the agent becomes committed to the action. Actions may be *private*, corresponding to an internally executed subroutine, or *communicative*, i.e., sending messages.

Soccer players must be able to receive information about the environment (ball and players of the other team) and from the other players of the same team. The decision of every soccer player uses all this information as the basis of the reflexive reasoning and particular actions must be taken. Six actions, four private and two communicative, are defined for soccer players (see “Table 1”).

〈 Private actions: shoot the ball (SHOOT), get the ball (GET), move forward (FORW), and go backwards (BACK).
〈 Communicative actions: send a decision to a specified soccer player (INFORM), and request an action to another soccer player (REQUEST).

PRIVATE	COMMUNICATIVE
SHOOT	INFORM
GET	REQUEST
FORW	
BACK	

Table 1 : Set of actions

When dealing with reactive systems, the controller is forced to send control signals to the system in every sample time and, in the soccer team this implies that the communicative actions cannot take a long time. Communication between agents could present sequences of commitments and timing problems appear when this communication takes longer than usual. Paying attention to that case, the number of commitments is constrained in every grain of time to be consistent with real time features.

Sometimes conflictive or unsolved situations appear, so actions are *criticised* [5] or reviewed by another co-operative agent, which has a global view of situations and suggests solutions to conflicts. Communicative actions involve changes on the certainty degrees of private actions so that fusion procedures are used to calculate the final actions in a more reliable and confident way.

Initial beliefs and agent capabilities can change according to the variation of the team strategy. In this work, databases are not used to store the whole history of the system, but destructive assignment (variables) are used to store only the 'recent' history.

2.2 STEP-BY-STEP REASONING

As mentioned above (see section 2), the reasoning procedure to control this soccer team has been accomplished by using three steps. Next sections describes how works these steps upon AGENT0 syntax, as well as some examples.

2.2.1 Reactive Decisions

In the first step of the reasoning procedure, every soccer player decides a private action instinctively. This decision depends on local environment configuration (BELIEFS) defined by two parameters: distance player-ball (DPB), and distance player-goal (DPG). Joint to this private action, the decision also has a degree of certainty (see *"Fig. 3"*) to increase knowledge about the others.

As an example, "*Fig. 3*a" shows a top view of the micro-robot with the ball in three possible situations. Decisions will be : SHOOT when at '1', GET when at '2', and FORW or BACK when at '3' depending on DPG value. By this way, the first reasoning procedure would be expressed like *rule 1* :

Rule 1 :

```
BEL ( AgentX, DPB, ZONE2) ⇒ INFORM ( to_any_agent, AgentX, SHOOT, 0.8)
```

Similarly, when at '3' in point 'M' (see), reasoning would be as *rule 2* :

Rule 2 :

```
BEL(AgentX,DPB,ZONE3) ∧ BEL(AgentX,DPG,FAR) ⇒
                                INFORM(to_any_agent, AgentX, FORW, certainty )
```

, where '`certainty`' is the final value obtained by fuzzy inference calculation.

After this first decision is taken, agents communicate it (INFORM) to the others through the communication channel. Thus, reactive behaviour is developed and create rough intentions.

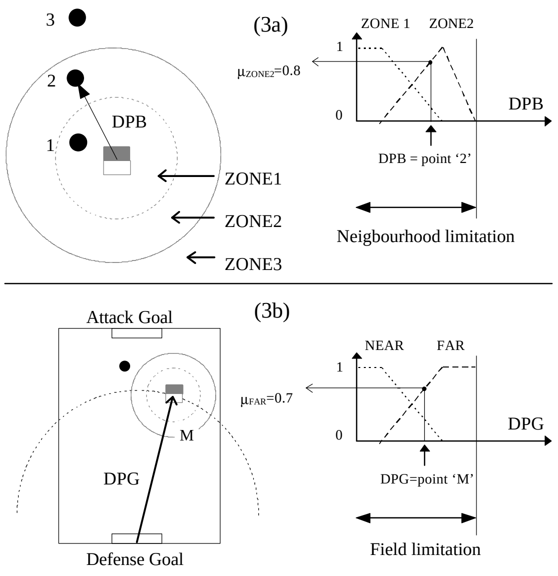


Fig. 3 : Reactive reasoning of the agents upon fuzzy sets (a)with DPB variable, (b) with both DPB and DPG variables

2.2.2 Co-operative Decisions

This step implements the cognitive reasoning. It begins with a REQUEST (for communication) action, so that every agent can know the *set (reactive_action, certainty, ID_player)* of all other playmates. Therefore, when two playmates take conflictive decisions, certainty degrees are taken into account and one of them changes the former reactive decisions. Finally, agents communicate the decision to the *coach-agent* by an INFORM action.

"Fig. 4" shows a situation where both Agent1 and Agent2 decide to GET the ball. After REQUEST themselves, Agent1 will change to another action because its DPB parameter brings less certainty than those obtained by Agent2.

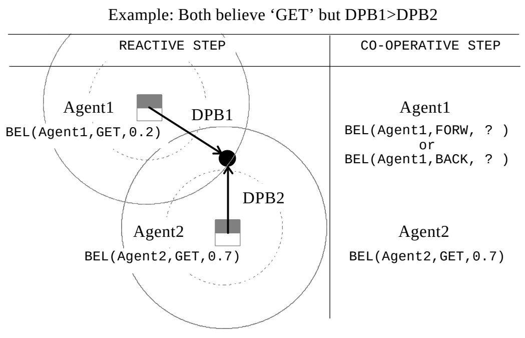


Fig. 4 Example of co-operative decision

This reasoning procedure could be expressed as :

```
INFORM(Agent2,Agent1,BEL(Agent2,SHOOT,0.7))  ∧ BEL(Agent1,SHOOT,0.2) ⇒
                                                BEL(Agent1,SHOOT,f(0.7,0.2))
```

, where $f(c1,c2)=\begin{cases} 0 & ,c1>c2 \\ c2 & ,\text{otherwise} \end{cases}$

In this situation, since f(c1,c2)=0, Agent1 will change to FORW or BACK action using rules like ‘ .

This kind of communication is an exchange of information, and the knowledge of the environment increases. In that case, formed reactive decisions become co-operative ones.

2.2.3 Fusion Procedure

Since this reactive team might be myopic in their approach, global information is useful and it is in this capacity that the *coach-agent* can interact with this multi-agent control team[9]. After co-operative set of decisions are received by the *coach-agent,* they are reviewed by a fusion procedure, which could be understood as a private action. Taking into account that this agent has a global overview of the environment, this difference in scope allows to solve remaining unsolved problems. This agent also contains some strategy parameters and can exert its influence based on a strategy of the match.

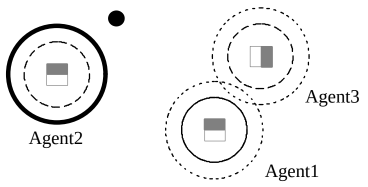


Fig. 5 Who GET the ball ?

“*Fig. 5*” shows an usual indecision- If none of the agents decides the ‘GET’ action but the strategy game needs to get it, the *coach-agent* will select the Agent2, which has the best position to get the ball. Only the *coach-agent* can evaluate this kind of situations because of its global scope, that is to say, it only knows where is the ball and where are the playmates.

By this way, the role of the *coach-agent* is to change some co-operative decisions only if needed, and then REQUEST them. By this way, the private action of Agent2 could be as follows :

```
REQUEST(Coach,Agent2, DO(GET)) ⇒  DO(Agent2,SHOOT)
```

2.3 LOCAL MOTION

Decisions are high level descriptions of the control signals to be executed by the micro-robots, so that, individual set-point positions must be calculated. In this work, the close neighbourhood of each micro-robot is described by matrices. Therefore, relative set-point positions can be described as individual matrix elements, and are calculated by means of qualitative control [10][11]. By this way, the close neighbourhood space is split and facilitates obstacle avoidance algorithms. “*Fig. 6*” shows typical simple path planning with an obstacle detected at ‘A’ point that delays the achievement of committments.

This delay highlights the importance of persistence in intention to achieve goals. However, the real implementation contains this planning in HOST due to the limited amount of memory in the micro-robots microcontroller.

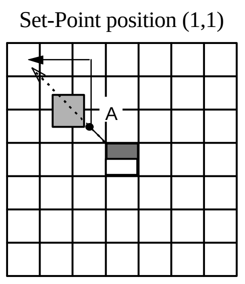


Fig. 6 : Obstacle avoidance delays the achievement of commitments

3. MULTI-AGENT PARADIGM USING Matlab/Simulink

Step-by-step reasoning brings the opportunity to use modularised languages as Matlab/Simulink is. In this case, graphical tools provided by SIMULINK are useful because every step of reasoning can be programmed as a sequence of blocks diagram.

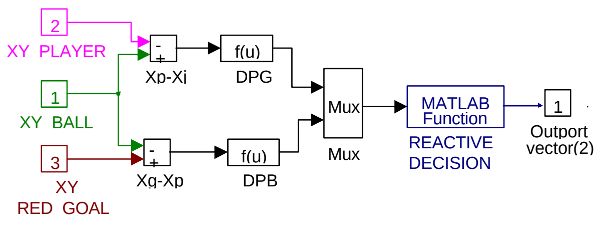


Fig. 7 First decision (PLY_X block in "Fig. 8")

As an example, *"Fig. 7"* shows the blocks diagram corresponding to the reactive decision block. Then, previously defined DPG and DPB parameters are calculated so that a MATLAB-function block executes a fuzzy reactive action (i.e., action and its degree of certainty).

With this graphical language, communication is performed using input and output arrows. Therefore, communicative actions as INFORM and REQUEST are easily implemented (see *"Fig. 8"*). An static table is placed between the first and the second steps of reasoning, and message passing is allowed (block 'TABLE').

Since this structure of communication could be understood as a basic blackboard-based scheme, the aim is to provide a high level communication channel widely used in agent paradigm.

The implementation of all reasoning procedures is fully depicted in *"Fig. 8"*, where PLY_X blocks are the reactive decision blocks of players 1, 2, and 3. The way that players send (INFORM) their decisions to the 'TABLE' block is implemented by using a MUX block, so that the decision set comes into the 'TABLE' block as a vector. In the next step, players will receive (REQUEST) that set of decisions from the 'TABLE' block as a vector, too.

Using the same vector representation, co-operative decisions come into the 'COACH-AGENT' block to be computed by a fusion procedure. This procedure consists of a sequential program that takes into account different strategy parameters : usual positions for every player, necessity to get the ball, and so on. Next, the coach-agent perfoms the COMMITMENT or the INFORM to-any-agent. Finally, a new set of decisions goes from the ‘COACH-AGENT’ block to every player and, locally, each one converts its final decision into a set-point position. This set-point is then executed by low level micro-robot motion architectures.

When the simulation facility is used, an animation block is used to simulate the micro-robot movements. In this case, a cinematic model of the micro-robot is introduced into another block and current positions can be calculated. But in real essays, current position values are provided by a vision system, and set-point positions will be sent to each micro-robot by the HOST computer (‘FM EMITTER’ block).

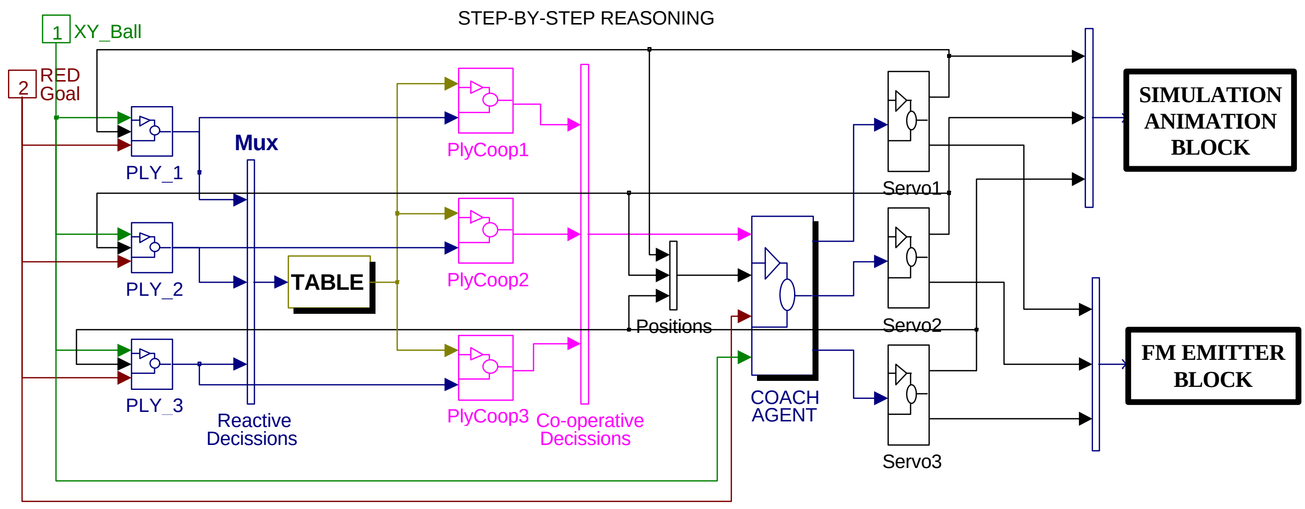


Fig. 8 Blocks diagram for Step-By-Step reasoning

“*Fig. 1*” shows a general scheme of the whole system, and the connections between its elements. So far, it is important to notice that communication between robots themselves is not available ; thus co-operative decisions are taken in the HOST computer. Therefore, since communicative actions related to reasoning process are designed by connecting blocks, an INFORM action from *coach-agent* to any-agent is implemented by using radio-communication.

4. REAL SYSTEM IMPLEMENTATION

Preceding sections described how private actions are decided using Matlab/Simulink framework. This section describe hardware, as well as special boards used for vision and communication. Paying attention to *“Fig. 8”*, the system description also includes micro-robot technical specifications split in *PLY_#* and *PlyCoop#.*

A main reason to use Matlab/Simulink software yields on the fact that one of its facilities, the Real-Time Workshop, brings the opportunity to generate C-code ready to be compiled and executed even with real time features. In this work, WATCOM C/C++ compiler [12] is used and constrained to obtain MS-DOS executable files. Also, another important feature is that specific drivers can be programmed to connect Matlab/Simulink to a hardware computer, like frame grabbing board, RS-232C serial communication port, I/O boards, and so on.

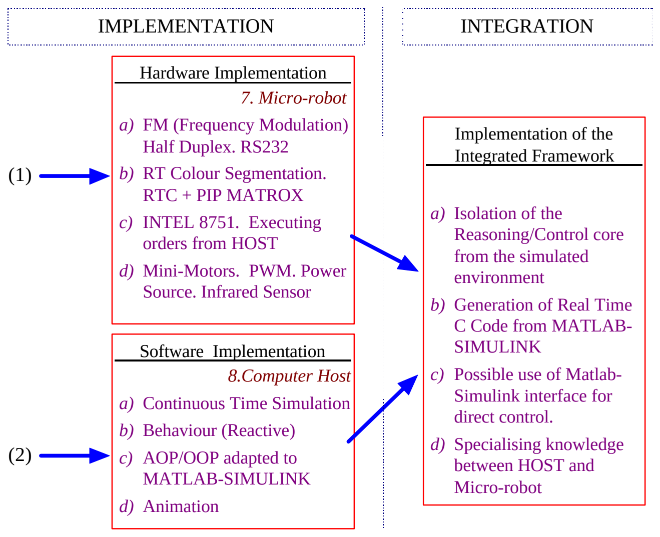

IMPLEMENTATION
INTEGRATION
Hardware Implementation
7. Micro-robot
a) FM (Frequency Modulation) Half Duplex. RS232
b) RT Colour Segmentation. RTC + PIP MATROX
c) INTEL 8751. Executing orders from HOST
d) Mini-Motors. PWM. Power Source. Infrared Sensor
(1)
Software Implementation
8.Computer Host
a) Continuous Time Simulation
b) Behaviour (Reactive)
c) AOP/OOP adapted to MATLAB-SIMULINK
d) Animation
(2)
Implementation of the Integrated Framework
a) Isolation of the Reasoning/Control core from the simulated environment
b) Generation of Real Time C Code from MATLAB-SIMULINK
c) Possible use of Matlab-Simulink interface for direct control.
d) Specialising knowledge between HOST and Micro-robot

Fig. 9 Implementation and Integration Phases

4.1 VISION SYSTEM

The reasoning procedure needs to know the set of positions, speed and orientation of the robots, so as the ball speed and ball position. Since speed can be calculated using preceding position values, current values are provided by a vision system. In this work, the vision system is composed by a CCD camera, a special-purpose real-time image co-processor (RTC) [13], and a PIP-1024B frame grabbing board [14].

The CCD video camera send images to the RTC which execute colour segmentation operations in real-time. After these operations, the frame grabbing board gets the image and digitises it to calculate gravity centres of segmented objects. As can be seen in *"Fig. 10"*, micro-robots have two colours to give important help to the vision system to obtain position and orientation of our team micro-robots.

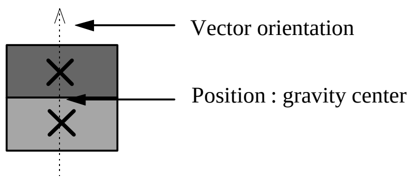


Fig. 10 Micro-robot top view colors

4.2 COMMUNICATION SYSTEM

Co-operative commands which are finally decided by the coach-agent are broadcasted to the robots by a FM radio-communication with a specially devised protocol. Because the FM emitter [15] can be connected to the serial communication port of the computer, co-operative commands can be sent directly from Simulink model.

Notice that communication problems can affect the behaviour, also the performance of the robots because the lack of this communication induces reactive actions instead of co-operative ones.

4.3 MICRO-ROBOT STRUCTURE

According to MIROSOT rules, robots have strong constraints on their size so that communication features and autonomous motion capabilities must be implemented by using small devices. Micro-robots are composed by one FM receiver [16], a microprocessor 8751, a power unit, two micro-motors [17], and batteries for motion, communication and control unit supply voltages.

When the communication unit receives information, the FM receiver converts it to RS-232C protocol and sent it to the on-board microprocessor. After programme calculations, control signals are generated and finally executed by PWM modulation.

The main program enables two interruption lines: a RS-232C port line, and an external one. With the first line, new commands from the host can be received so that new relative set-point positions are accepted. On the other hand, the external interruption line is connected to an infrared sensor in order to avoid collisions or finely detect the ball.

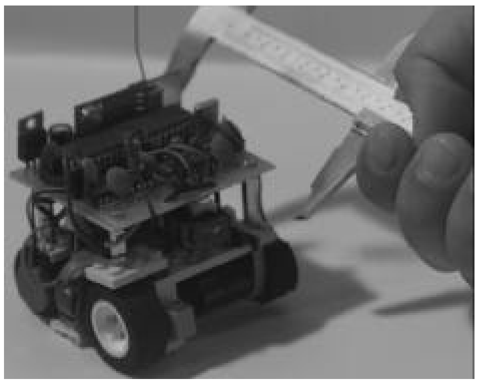


Fig. 11 The Implemented Soccer Micro-robots

5. CONCLUSIONS

This project is quite stimulating in several terms. With respect to the technical features of the proposed solution, the Matlab/Simulink integrated vision system is able to calculate position and orientation of the ball and micro-robots taking approximately 750ms. long. The micro-robots are not as autonomous as desired to apply agents approaches on stand-alone. The system was applied on-line without the immediate need of generating C code.

With respect to the research on this project, the first idea from the analysis phase is that agents are a natural way of thinking about distributed, specialised autonomous systems. To deal with this aim two interesting results have appeared :

The former, the Shoham selected AOP [2] is quite general and basically enough for our system, and though implementation of cooperation is done in the HOST the conceptual features are cooperative. Let us explain further on this: the analysis of this project is a subset of SHOHAM AGENT0, and anyway, the solution is proper. In the design phase the subset of SHOHAM was equally enough. In the implementation phase, most of the responsibility for the co-operation is supported by the HOST system mainly due to the half-duplex nature of the communication device finally implemented in our robots, and also by the limited capabilities of the chosen INTEL 8751 micro-processor. Furthermore, since these micro-robots are not fully sensorised, this is not, so far, a loss of generality of the final solution.

The latter, the necessity of implementing AOP within a CACSD framework closer to control engineers, is justified for facilitating the analysis and design phases because, first, of the continuous time dynamic simulation facilities of the selected CACSD system, Matlab/Simulink, and second, because the amount of control oriented toolboxes that are developed from current research to this environment suggest a brilliant platform for the application of AOP in automatic control and automation. However, implementing the AOP is not easy because this is not an OOP framework. One conclusion is that Matlab/Simulink could extremely help the analysis and design phases if this system was object-oriented.

Nowadays, some works try to deal with this lack [18][19]. On the other hand, in the implementation phase this tool is now quite powerful to generate C software code (both non-real and real time operation) which is extremely interesting in obtaining a final integrated automatic control-oriented system [20].

Therefore, this could be a first step to a general application of agents in the world of automatic control and robotics by using tools that are common in the former area. The drawback is that object oriented paradigms are not extended enough and agents have problems to be applied in practical implementations. This paper shows a possibly first success in practical application/implementation of agents in automatic control.

6. FUTURE WORK

Further work on Matlab/Simulink must improve support object oriented tools so that research community could go on to research supervisory control architectures, methodologies, and practical applications. A near future version Matlab/Simulink is announced to incorporate *software structures*. *Structures* will facilitate the application of these ideas to embed objects better. However, these structures are expected to become real objects in the next versions as required for AOP.

7. ACKNOWLEDGEMENTS

This work has been partially funded by the TAP96-1114-C03-03 project *Plataformas Integradas de CAD de Supervisión y Metodologías* of CICYT program from the Spanish government.

References

[1] Micro-Robot Soccer team Tournament. November 9-12, 1996. KAIST. Korea.

[2]Y.Shoham. “Agent-oriented programming”, Artificial Intelligence, 60 (1993) 51-92.

[3] T.Balch, G.Boone, T.Collins, H.Forbes, D.Mackenzie, J.C.Santamaria, "Io, Ganymede and Callisto: a multiagent robot trash-collencting team". AI Magazine, Vol.~16, No.~2, Summer 1995, pp.~39-51.

[4] P.J.Gmytrasiewicz, E.H.Durfee, D.K.Wehe. "The Utility of Communication in Coordinating Intelligent Agents". Proceedings of the 9th National Conference on Artificial Intelli-gence, 1991.

[5] J.LL.de la Rosa, J.Aguilar, I.Serra *“Heuristics for Cooperation of Expert Systems. Application to Process Control”*. Ed. PIAR, 1994. ISBN 84-605-0275-9.

[6] M. Wooldridge, N.R. Jennings, *“Intelligent Agents : Theory and Practice”*, Submitted to Knowledge Engineering Review, October 1994, Revised January 1995.

[7] "COORDINATION: Linguistic Support for Multiple Cooperating Agents". ESPRIT Project No 9102 in the Basic Research Action. Work Area: Basic Aspects of Multiple Computing Systems. Dates: 1.2.94 - 31.1.96.

[8] T.Finin et.al., DRAFT Specification of the KQML (The Knowledge Query & Manipulation Language). Unpublished draft, 1993.

[9] R.C.Arkin, K.S.Ali. "Integration of Reactive and Telerobotic Control in Multi-agent Robotic Systems". Proceedings of the 3th International Conference on Simulation of Adap-tive Behaviour.

[10] N.Agell, J.C.Aguado, and N.Piera, 1995, "A Qualitative Mobil Tracking", Current Trends in Qualitative Reasoning and Applications, Monograph CIMNE Nº33, pp. 102-107.

[11] J.LL.de la Rosa, A.Oller, et. al., "A Comparison of Fuzzy and Qualitative Control Techniques". IEEE International Conference on Control Applications / International Symposium on Intelligent Control / International Symposium on Computer-Aided Control System Design. pp 139-144. ISBN 0-7803-2978-3. Dearborn, USA. September, 1996.

[12] Watcom C/C++ (ver. 10.5). Tenberry Software, Inc. 1994.

[13] Real-Time video Co-processor (RTC) was developed by Vision Group of Electronics, Informatics and Automatics Department of University of Girona.

[14] PIP-1024B digitizer plug-in board is a comercial product of MATROX.

[15] Radiofrequency transmitter module, MOD: TX-433-SAW, 433,92 MHz. Totem Line. AUR·EL S.p.A.

[16] Super-Heterodyne receiver with monolithic filter oscillator. MOD: STD 433 SIL. 433,92 MHz. Totem Line. AUR·EL S.p.A.

[17] Maxon DC Motor, MOD: 2017.938-22.162-000. 12 volt, 1.5 W, 148000 rpm., and Maxon gear, MOD: 2916.804-0123.0-000, 122,6 :1.

[18] J.Melendez, J.LL.de la Rosa, J. Aguilar, "Embedding Objects into Matlab/Simulink for process Supervision", accepted paper at IEEE Control Systems Magazine to appear in October 1997.

[19] J. Melendez, J.LL. de la Rosa, J. Colomer, J. Vehí, C. Pous, "Incrustación de objetos en Simulink. Integración de herramientas de ayuda al diseño de estructuras de supervisión", in the 'II Congreso de usuarios de MATLAB'. Madrid, Spain. September, 1996.

[20] C. Pous, A.Oller, J. Vehi, J.LL. de la Rosa, "Using Real-Time Workshop in teaching control design techniques". Volume on Real Time Systems Education. Ed. by IEEE Computer Society Press, ISBN 0-8186-7649-3. pp 153-158. August, 1996.

Josep Lluis de la Rosa i Esteva obtained a degree in Sciences at the Automous University of Barcelona, Bellaterra (Spain) in 1989. In 1989 he joined Siemens-Nixdorf Software Development Center in Barcelona. In 1991 joined the Laboratoire d'Architecture et d'Analyse des Systèmes of the Centre National de la Rechercher Scientifique (LAAS-CNRS) in Toulouse (France), where he got his Ph.D. degree in Sciences by the Autonomous University of Barcelona in 1993. And from 1994 he is in the University of Girona, adscribed to the Institute of Informatics and Application (IIiA). His present research interests include artificial intelligence applied to supervisory systems and process control, and multi-agents systems. WWW: http://eia.udg.es/~peplluis

Albert Oller i Pujol obtained a degree in Sciences at the Automous University of Barcelona, Bellaterra (Spain) in 1990. From 1994 he is in the University of Girona, adscribed to the Institute of Informatics and Application (IIiA). His present research interests include artificial intelligence applied to supervisory systems and process control, and multi-agents systems. WWW : http://eia.udg.es/~oller

Josep Puyol-Gruart obtained a degree in Electronic Engineering at the Polytechnic University of Catalonia, Barcelona (Spain) in 1985. From 1987 he has worked for the Spanish Council for Scientific Research (CSIC) at the Artificial Intelligence Research Institute (IIIA). He got his Ph.D. degree in Computer Science by the Autonomous University of Barcelona in 1994. His present research interests include knowledge-based systems and multi-agents systems. WWW: http://www.iiia.csic.es/~puyol

Josep Vehí obtained a degree in Sciences at the Autonomous University of Barcelona, Bellaterra (Spain) in 1987. Since September 1987 he has worked at the Technical University of Catalonia. Since 1991 he has been working at the University of Girona, adscribed to the Institute of Informatics and Applications (IIiA). His present research interests include interval methods applied to robust control parametric design and interval simulation.