Presentación

El Workshop Hispano-Luso de Agentes Físicos pretende ser un foro de encuentro que permita el intercambio de información y experiencias relacionadas con la aplicación del concepto de agente en entornos físicos, especialmente en el control y coordinación de sistemas autónomos: robots, robots móviles, procesos industriales o sistemas complejos.

Su primera edición tuvo lugar el año 2000 en la Universidad Rovira i Virgili (Tarragona). En aquella ocasión se reunieron diversos grupos de investigación de la península ibérica interesados en la aplicación de las tecnologías de agente. El gran ambiente creado por el comité organizador permitió establecer contactos entre los diversos grupos así como intercambiar ideas de forma altamente productiva. Con el mismo espíritu esperamos fomentar la colaboración entre los distintos grupos en esta ocasión.

Finalmente, me gustaría agradecer la colaboración de todos aquellos que han ayudado a hacer posible el WAF'2001. En primer lugar, mi agradecimiento al grupo de revisores, que ha realizado un desinteresado trabajo de corrección y selección de los trabajos recibidos. Tampoco quiero olvidar a los profesores del Grupo de Sistemas y Comunicaciones de la URJC que han proporcionado el soporte necesario para poder organizar este Workshop. Por último, agradecer las aportaciones económicas y de gestión a las distintas entidades que han colaborado en la organización del Workshop, en especial a la Universidad Rey Juan Carlos donde ha tenido lugar el mismo.

Marzo de 2001

Vicente Matellán
Editor WAF'2001

Preface

The Spanish-Portuguese Workshop on Physical Agents intends to be a forum to allow knowledge and experience exchange related to agent concept in physical context, mainly in control and autonomous robot coordination: robotics, mobile robots, industrial processes or complex systems.

Its first edition was held at Universidad Rovira I Virgili (Tarragona) in 2000. Several research groups coming from both countries and interested in applications of agent technology gathered there. Organization committee favoured a good atmosphere that allowed creation of links between different groups and efficient ideas exchange. This time we also hope to promote collaboration between different groups.

Finally, I would like to thank all people who helped in the WAF'2001 organization and made it possible. First, my acknowledgement to reviewers group, who generously did a great job in correction and selection of received papers. I would like also to mention teachers from System and Communication Group -GSYC- from URJC, for the teamwork in the arrangements. Last, I gratefully appreciate contributions and management support from different entities that participated in workshop organization.

Marzo de 2001

Vicente Matellán
Editor WAF'2001

Organización

El Segundo Workshop Hispano-Luso en Agentes Físicos (WAF'2001) organizado por el Grupo de Sistemas y Comunicaciones, del Departamento de Ciencias Experimentales e Ingeniería de la Universidad Rey Juan Carlos.

Comité Organizador

Coordinador: Vicente Matellán Olivera (Universidad Rey Juan Carlos)
José María Cañas Plaza (Universidad Rey Juan Carlos)
José Centeno González (Universidad Rey Juan Carlos)
Jesús María González Barahona (Universidad Rey Juan Carlos)
Pedro de las Heras Quirós(Universidad Rey Juan Carlos)

Comité de Programa

Coordinador: Vicente Matellán Olivera (Universidad Rey Juan Carlos)
Revisores: Eugénio Costa Oliveira (Universidade do Porto)
María Conceição Neves (Instituto Superior de Engenharia do Porto)
Pedro Almeida Lima (Instituto de Sistemas e Robotica, Lisboa)
Daniel Borrajo (Universidad Carlos III)
Albert Oller (Universitat Rovira i Virgili)
Josep Lluís de la Rosa (Universitat de Girona)
Bianca M. Innocenti (Universitat de Girona)
Antonio González (Universidad de Granada)
Domingo Guinea (Instituto de Automática Industrial, CSIC)
Luis-Manuel Tomás (Universidad de Murcia)
Miguel Toro (Universidad de Sevilla)
José Soler (Universidad Politécnica de Valencia)
Luis Paulo Reis (Universidade Fernando Pessoa)

Entitades Colaboradoras

- Red Temática Eurobot-IB
- Escuela Superior de Ciencias Experimentales e Ingenieria
  Universidad Rey Juan Carlos
- Comisión Interministerial de Ciencia y Tecnología (CICyT)

Organitation

The Second Spanish-Portuguese Workshop on Physical Agents (WAF'2001) has been organized by the Systems and Communications Group of the Experimental Sciences and Engineering of the Rey Juan Carlos University.

Local arrangements committee

Coordinator: Vicente Matellán Olivera (Universidad Rey Juan Carlos)
José María Cañas Plaza (Universidad Rey Juan Carlos)
José Centeno González (Universidad Rey Juan Carlos)
Jesús María González Barahona (Universidad Rey Juan Carlos)
Pedro de las Heras Quirós(Universidad Rey Juan Carlos)

Program Committee

Chair: Vicente Matellán Olivera (Universidad Rey Juan Carlos)
Referee: Eugénio Costa Oliveira (Universidade do Porto)
María Conceição Neves (Instituto Superior de Engenharia do Porto)
Pedro Almeida Lima (Instituto de Sistemas e Robotica, Lisboa)
Daniel Borrajo (Universidad Carlos III)
Albert Oller (Universitat Rovira i Virgili)
Josep Lluís de la Rosa (Universitat de Girona)
Bianca M. Innocenti (Universitat de Girona)
Antonio González (Universidad de Granada)
Domingo Guinea (Instituto de Automática Industrial, CSIC)
Luis-Manuel Tomás (Universidad de Murcia)
Miguel Toro (Universidad de Sevilla)
José Soler (Universidad Politécnica de Valencia)
Luis Paulo Reis (Universidade Fernando Pessoa)

Sponsors

- Eurobot-IB Tematic Network
- Escuela Superior de Ciencias Experimentales e Ingeniería
  Universidad Rey Juan Carlos
- Comisión Interministerial de Ciencia y Tecnología (CICyT)

Contents

Sesión 4: Agentes Físicos II

Sesión 5: Docencia

Anexo: Presentaciones de Proyectos

Conferencia Invitada

Towards Agent-based Architectures for Advanced Decision Support

Sascha Ossowski and Juan Manuel Serrano

Grupo de Inteligencia Artificial
Escuela Superior de Ciencias Experimentales y Tecnología
Universidad Rey Juan Carlos
e-mail: {s.ossowski, jserrano}@escet.urjc.es

Abstract This paper discusses the potential benefits of using agent technology in the design of Decision Support Systems. The concept of *intelligent assistant* is conceived as a personalised software agent that renders support to a decision-maker on the basis of a flexible and adaptive human-agent interaction. The paper provides an overview of design questions respecting the architecture and the knowledge endowment of such agents. In particular, it discusses how the FIPA agent communication language can be extended to provide intelligent assistants with the advanced communication facilities required by decision support domains.

1 Introduction

The outside world is full of systems which are governed by complex laws of behavior. Independently of whether these systems are made of unanimated entities, governed by the laws of physics, organizations of humans with predefined artificial process rules or a mixture of both, often there is a need to influence their dynamics and to bias their evolution into a desired direction. The consequences of natural disasters, such as floods, can be alleviated if the spill gates of dams are managed to distribute the water volume in the watershed basin, keeping rivers and channels from overflowing. In industrial plants the production processes need to be monitored and adapted in order to ensure the quality of the final product, economic efficiency and security. Faults in chemical industries require actions that restore normal conditions and prevent the formation of toxic clouds. Computer networks have to be managed in order to maintain a certain quality of service to the users, which requires upper bounds on message delays etc.. In much the same way, companies are interested in maintaining their business processes effective and robust. The flow of work in an office is to be kept smooth despite the illness of employees; material flow on car assembly lines should not be disrupted by any kind of contingencies. Traffic domains comprise a mixture of natural and artificial laws of behavior: road traffic flows have to be influenced so as to avoid traffic jams , to reduce travel times etc.. In air traffic control, the primary concern is to influence the planes' routes so as to avoid accidents. This list could be extended easily.

The above examples motivate that it is a major challenge — for both economic and social reasons — to take the adequate control decisions to maximize the efficiency of the systems and to minimize the negative impact of faults. Sometimes

one single person, but usually a group of *operators* is in charge of taking such decisions and is responsible for their effects. This implies to monitor continuously the system to be managed and requires to take decisions respecting changes in control variables, usually in *real time*. The increasing data volume and the decreasing time horizon within which control decisions have to be taken, have generated a need for computer applications that support the responsible persons in this task. These *decision support systems (DSS)* acquire data about the system state either directly or through (real-time) databases, on top of which they device an intelligent monitoring system that warns the control personnel of undesired evolution and answers their questions concerning potential reasons, effects and therapies [1]. For instance, DSSs assist hydrology engineers to decide upon actions on spill gates, chemical engineers to manage modifications on valves etc. and network administrators on configuring routers, leasing lines and identifying faulty equipment [8]; support to business managers is rendered by suggesting modifications in the usual work process, e.g. by reassigning work tasks or by modifying the route of some product through a jobshop [10]; advice for traffic engineers may consist in suggesting which road traffic signals should be set or which air corridors should be assigned to certain flights [4].

This paper discusses the potential benefits of using agent technology in the design of advanced DSSs. In particular, it conceives the concept of *intelligent assistant* as a personalized software agent that renders support to a decision-maker on the basis of a flexible and adaptive human-agent interaction. It is organized as follows: Section 2 introduces the concept of intelligent assistants and provides an overview of design questions respecting the architecture and the knowledge endowment of such agents. Section 3 gives an example of how some questions respecting the communication capabilities of an intelligent assistant can be tackled; it is shown how the FIPA agent communication language can be extended to cope with decision support problems. We conclude the paper in Section 4 providing pointers to present and future lines of research.

2 Agent-based Decision Support

Intelligent agents are becoming popular as a key abstraction in the construction of complex software systems. They are usually conceived as entities capable of achieving tasks delegated to them by their human users in a (semi-)autonomous and (individually) rational fashion. To do so, they are endowed with substantial communication facilities, that allow them to maintain flexible interactions with other agents both in a reactive as well as in a proactive manner.

In this section we discuss the implications of applying intelligent agent technology to decision support domains. Therefore, in Section 2.1, we first introduce the concept of intelligent assistants, as agents that render support to their human users in all stages of the decision-making processes that they are involved in. Subsequently, in Section 2.2, we do first steps towards an agent architecture for intelligent assistants,

analyzing the different competence models that they need to be endowed with so as to act successfully in decision support domains.

2.1 Intelligent Assistants for Decision Support

Early DSS were conceived as simple information systems that collect and facilitate decision relevant data [18]. Despite some intends to improve organization and presentation of such data by means of additional deductive facilities [5], it soon became apparent that the fundamental problem for a decision-maker is not such much to access pertinent data but rather to understand its significance. French [3] outlines that the key challenge for next-generation DSS is to help decision-makers in building up and exploring the implications of their judgements, so as to take decisions based on understanding. We believe that, due to their aforementioned characteristics, agent-based systems that serve as *intelligent assistants* for the decision maker provide a major step towards that goal.

Suppose an agent-based DSS that assists operators (the decision-makers) in a traffic control centre to generate and choose among alternative signal plans for traffic control devices (Variable Message Panels, Traffic lights etc.), so as to assure a smooth flow of traffic, as well as to avoid and/or overcome potentially critical situations. The standard way of interaction with such a system is in a reactive manner [1]. Once an operator detects some abnormal system parameters or receives alarm messages (e.g. from traffic observers), she initiates a dialogue with the DSS in order to prepare her decision. There are several questions that she will put forward to the agent, respecting the cause of the current situation (*What is happening?*), the reason for it (*Why is it happening?*), and action alternatives (*What can be done?*). She will also be interested in comparing the potential effect of different alternatives (*What may happen if ?*). These questions will trigger several processes in the course of which the agent autonomously gathers relevant data from the different information sources available. On the basis of this information, it will apply its domain knowledge to generate answers to the questions, taking into account rationality constraints with respect to the operator's objective of minimizing negative impacts on the part of the road network that she is responsible for.

Another scenario is given when the intelligent assistant monitoring the road network detects symptoms of a deterioration of the traffic situation or of dysfunctional active signal plans (initiated, or not adequately updated, for instance, by a less trained or experienced operator). In this case, it proactively issues warnings, initiating a dialogue with the decision-maker, in the course of which the latter explores the reasons and implications of that warning in a flexible and intuitive manner. To allow for this, the agent not only needs to be endowed with a domain ontology that reflects the abstraction level at which the operator conceives the system and its behavior (motorways, road sections, control devices, traffic problems, causes, consequences, and control actions etc). It also needs to possess an extended and semantically rich set of communicative actions, that allow for the emergence of complex interactions.

In the traffic management domain, such an interaction between the decision-maker (DM) and the personal assistant (PA) may have the following shape:

PA – I warn you that a potential traffic problem is emerging on section s_{21} *of motorway* M_{60} *eastbound*

DM – Why do you say that?

PA – Because the speed in sections s_{17} *and* s_{22} *is about to be classified as* low *while the occupancy is starting to be* high

DM – Ok. Which are the causes of that problem?

PA – There has been a sudden increase of traffic flow emanating from the M_{60} / N_8 *motorway junction.*

DM – Which are the alternatives for the solution of the problem?

PA – There are two possible action alternatives: either signal plan π_1 *involving control devices* δ_{28}, δ_2 *and* δ_1 *deviating traffic through area* α_{45}, or *signal plan* π_1 *involving control devices* δ_{28}, δ_2 , δ_{25}, δ_{77} *and* δ_1 *deviating traffic through areas* α_{41}, α_{42} *and* α_{43}

DM – Which effects might be expected in case of implementing the first alternative?

PA – Some minor delays at sections s_{21} *and* s_{21} *of motorway* N_8 *northbound*

DM – Execute the first alternative.

2.2 Towards an Architecture for Intelligent Assistants

A wide variety of different agent architectures have been presented that instrument different agent models in a variety of domains. Despite some work directed towards generic agent reference architectures, most approaches have been optimized to better cope with the specific characteristics of a particular domain [9]. Our aim is to come up with an architecture for intelligent assistant agents. In this section we make first steps towards that goal by describing the different competence models that an intelligent assistant is to be endowed with.

Domain Knowledge Model In order to cope with its tasks of providing advise to its human user, an intelligent assistant needs to be able to perform different types of reasoning within the problem domain. The following reasoning tasks are essential:

Classification Setting out from the available information about the state of the world, the intelligent assistant classifies the situation with respect to its desirability. As a result, it obtains a set of problematic features of the current situation. For instance, an assistant for traffic management system will receive numerical data from road sensors, on the basis of which it may classify situations as "fluid", "slight delays at junction j_{48}" or "medium congestion in area α_5".

Diagnosis On the basis of the problematic features, the "symptoms" that indicate that something is going wrong in the modelled system, the diagnosis task comes up with an explanation that identifies the causes of such undesirable behavior. For

instance, diagnosis might explain the "medium congestion in area α_5 by an incident at the outlets of that area".

Prediction The prediction task evaluates how the current state of the environment will evolve given that no actions are taken by the decision-maker, or that the action policies currently in use are not changed. For instance, this task might conclude that if no actions are taken, the congestion in area α_5 will become severe.

Planification This task generates a set of plans or alternative action policies that are considered to be adequate to overcome the problems identified previously. In the example, this will be the different plans' traffic guidance that deviate traffic from congested area α_5.

Agent models Intelligent assistants usually operate in multiagent worlds. This may be either due to the fact that there are several decision-makers present (each of them possibly endowed with a personal assistant) and whose objectives are only partially compatible, or because the agent designer deemed it advantageous to base the DSS itself on a multiagent architecture. For instance, in the traffic domain one might want to subdivide the task of managing a road network into the problems of managing smaller subnetworks so as to combat complexity. In either case, an agent needs to be endowed with a model of its acquaintances, which includes at least a model of how they may *influence* the achievements of its local goals. This comprises the capability of performing actions that make it impossible of at least hinder the effectiveness of the agents own options, but it may also include the possibility of facilitating or enabling them [12]. In the traffic domains, an agent responsible for managing subnetworks will need to know whether others may use the same control devices as itself, and whether they may deviate traffic towards its control area, thus having the possibility to influence the results of its local control plans.

An intelligent assistant is provided with at least two distinguished agent models. First, a *self-model* contains information characterizing the agent's own capabilities and tasks, so as to be able to choose the most adequate action from a set of alternatives. Second, as decision-makers require personalized advice, the intelligent assistant agent needs to be endowed with a *user model*. As in most settings the outcome of the user's action alternatives is uncertain, this model will include facts such as her attitude towards risk, as this is a primary factor in the generation of adequate advice strategies. It may also contain information respecting the user's preferred modes of interaction etc.

Interaction model One essential characteristics of an intelligent assistant for a decision-maker is its high degree of communicative competence. Therefore, it cannot rely on precompiled communication structures, but needs to be able to plan an interaction according to its communicative goals. As such, it needs to be endowed with an agent communication language (ACL) that provides an adequate set of

communicative actions, so as to be able to modify (its model of) the beliefs of others. For instance, in the above dialogue the PA warns the decision maker of certain problems, and it explains why a certain situation is problematic. In addition, the interaction model may also include different modes of interaction, depending on the communicative act.

Strategic model Strategic knowledge is necessary to enable the assistant to coherently choose among different action alternatives. This may be either the case for action relating directly to the environment (e.g. different control plans), but also to communicative actions (different types or modes of communications). As such, it constrains the agents choices, due to obligations derived from user preferences, from communication protocols etc.

3 Example: An ACL for Decision Support

In this section we will present aspects of the interaction model in more detail. In particular, we will focus on the pragmatics of the Agent Communication Language for Intelligent Assistants in the domain of Decision Support. By the "pragmatics of an ACL" we understand the catalogue of communicative acts which the agent can perform. This catalogue is built on top of the FIPA-ACL.

3.1 FIPA-ACL

FIPA-ACL [13] [2] has been designed as a domain-independent language. Its standardized CAs are generic enough to be used in virtually any possible MAS domain. The main reason for this is the generic ontology underlying its logical framework [14]: a FIPA compliant agent i is just supposed to hold beliefs respecting generic facts p ($B_i p$), or to be uncertain about some of them ($U_i p$). In addition, certain facts can be chosen as goals to be accomplished ($C_i p$), and FIPA agents can refer to actions with specific pre-conditions and post-conditions. Within this generic ontology, the FIPA-ACL catalogue is designed to allow artificial agents to speak about information and action performing. These basic functions imply the broad applicability of FIPA CAs: in any domain, agents need to exchange information, and, in the same way, they will probably need to speak about actions.

The catalogue of communicative actions for an ACL constitutes a trade-off between expressive power and simplicity. For instance, table 1 summarizes the different conditions that FIPA ACL imposes on the use of *informative* CAs, i.e. actions directed toward the modification of the addressee's beliefs. In principle, it would be enough to have a single performative, named *tell* for instance, with the following model (note the empty context-relevance preconditions):

$$<i, tell(j, p)>$$

	Context-relevance preconditions					
Ability preconditions	$B_j \neg p$	$U_j \neg p$	$U_j p$	$B_j p$	$\neg(Bif_j p \vee Uif_j p)$	**Rational effect**
$B_i p$	Disconfirm($\neg p$)		Confirm(p)		Inform(p)	$B_j p$
$B_i p$		Confirm($\neg p$)	Disconfirm(p)		Inform($\neg p$)	$B_j \neg p$

Table1. FIPA informatives CAs

$$FP: B_i p$$
$$RE: B_j p$$

However, a more complex CA set is preferred to this single performative, on the basis of a gain in expressiveness: by means of more complex context-relevance preconditions, we can design CAs which has more presuppositions than the simple *tell*. For instance, by performing a *confirm* that p, the speaker not only expresses his intention that the hearer believes in p (i.e. he is not only *telling* him that p), but his assumption that the hearer might be uncertain about p.

In a specific domain, we can describe the agent's communicative behavior in less generic terms than the ones used by FIPA-ACL: agents do not just exchange information, but maintain semantically (and pragmatically) richer conversations. For instance, in a traffic management context, an assistant *warns* the operator about same undesired events; in the automatic tutoring domain, the instructor *corrects* the student about the proper way to solve some problem; etc. These CAs are not included in the FIPA-ACL catalogue, but appear to be useful in an domain-specific ACL. Of course, the generic acts of the FIPA-ACL catalogue can still be used in the certain domains, but, in some cases, more expressive CAs are preferred. The gain in expressiveness that an extended domain-specific ACL provides, goes in line with of following additional advantages:

- Agents can convey more information in a single utterance (transport message), thus making the dialog more simple or synthetic.
- In this sense, it helps to simplify the description of the agents' communicative behavior, i.e., by using more synthetic messages the interaction protocol results simpler.
- As a consequence of the augmented expressiveness, the required communication bandwidth can be reduced.
- From a human-computer interaction perspective, the artificial agent can communicate with human users at a higher abstraction level, thus reducing the gap between the artificial messages and the human language pragmatic actions.

3.2 A domain extension to FIPA ACL for DSS

This section aims at identifying the actions that the DSS and decision-maker perform when they speak about a system's components, problems, causes, etc. As claimed

before, to say that they are just exchanging information about problems, causes, and so on, is a rather generic way to describe their activity (although, of course, they have to inform each other of the relevant problems and causes). From a global perspective, the DSS is *rendering support* to the responsible person, while the latter is *asking for support*. This joint activity is performed following a given pattern, where several stages can be identified: main objective of the initial stage of the interaction is *problem understanding*; in the following stage of interaction, the generation of a *plan of control actions* to alleviate the problems is the essential functionality. Both phases can be complemented by an *explanation analysis* stage. Each of these stages of the interaction, so as the whole interaction will be formalized with the corresponding protocols.

A more fine-grained analysis, focusing on what the agents are doing in saying, leads to the identification of the communicative acts. In the following, we will realize such a detailed analysis on the basis of the example dialogue of section 2. The interaction starts with a *warning* issued by the assistant concerning what it believes to be a severe problem in one of the main system's components. With this utterance, it does not intend the responsible to immediately act upon that problem, but rather to turn her attention on it. Next, the operator *requires* the assistant *to justify* its inference respecting the presence of that problem. She is supposed to be uncertain about the actual presence of that problem, and tries to achieve a higher level of confidence in the information conveyed by the warning. Then, the assistant *explains* the operator that inference in terms of certain patterns of object attributes, which it believes to prototypically characterize that problem. The aim of the assistant is to bring the responsible to believe that a relevant problem is emerging.

Once the operator acknowledges this, she *requests* the assistant *to report a diagnostic* of the problem. Consequently, it will *inform* about its beliefs on the state of the components, that are supposed to cause the problems in the component being analyzed. At this moment, the responsible has gained a satisfactory understanding of the situation, and passes to the analysis of the possible solutions to the problem. So, she *asks* the assistant *to propose* a possible action plan. The assistant *proposes* a double alternative that, as it believes, can bring the affected component to a problem-free state. This proposal consists of two different type of plans. To evaluate the proposed actions, the operator *requests* the assistant *to report a forecast* of the consequences expected if the first alternative was implemented. The assistant *informs* her that some minor problems are likely to occur in some secondary components. As this satisfies the operator, she finally requests the assistant to implement that action. A summary of the previous analysis can be found in table 2, It shows the relevant actions of both, the DSS and the responsible person:

3.3 Formalization of the Communicative Acts

The previous high-level analysis of the sample dialogue leads to the identification of ten communicative acts. Only one of them, the request act, is already present in the

Responsible Person	DSS
Request-for-explanation	Warn
Request-for-diagnosis	Explain
Request-forecast	Report-diagnosis
Call-for-proposal	Report-forecast
Request	Propose-action

Table2. Communicative Actions for Decision Support

FIPA-ACL catalogue. In the sequel, and as an example, will present the formalization of the new *warn* CA. In addition, some comments concerning the planning process involving this CA will be given.

To warn someone about some fact p is a special case of informing about the possible occurrence of p. So, a warning inherits the feasibility preconditions of the inform CA: the sender believes p (sincerity conditions) and that the hearer does not hold any propositional attitude over p (context-relevance preconditions). It also inherits the rational effect intended by the speaker: it is intended to make the hearer believe p is possible. However, a warning presupposes an additional context-relevance feature: the speaker believes the negation of the proposition being communicated to be in the interest of the hearer. In other terms, the speaker believes that the situation being referred to has a negative impact on the goals of the hearer. We formalize this as a restriction in the model of the *inform* CA:

$$<i, warn(j,p)> \overset{\text{restricts}}{\equiv} <i, inform(j, Possible(p))>$$
$$FP: B_i C_{\neg} p$$

So, the warning issued by the DSS in the sample dialogue can be represented by the following expression:

$<a, warn(r, holds(problem(system - component(c1), p1))))>$.

The use of this communicative action within emerging conversations can be established by an automatic deduction process. For that purpose, a set of *generic* rationality axioms (or cooperative principles) are used [15]. Still, in order for the agent to plan this specific communicative action, it is convenient to add the following more specific cooperative axiom, which provides the reasons to perform a warning.

Axiom 1 $\models B_i(Possible(q \wedge C_j \neg q) \Rightarrow I_i B_j Possible(q))$

If the agent is equipped with the FIPA ACL catalogue then there are probably other candidates to achieve the given intention: either *inform*, *confirm* or *disconfirm*. However, the agent should issue a warning instead of a (dis)confirmation or inform action, as it conveys more information. This characteristic can be endorsed by the following axiom:

Axiom 2 $\models I_i Done(a_1 | \ldots | a_n) \Rightarrow I_i Done(a_j)$ *,where* a_k*,* $k \in 1 \ldots n$*, are actions with the same RE (rational effect), and* a_j *is the more* expressive *action.*

4 Conclusions

In this paper we have discussed the potential of using agent technology in the design of advanced Decision Support Systems. In particular, we have argued that intelligent assistants can integrate different competence models necessary for providing flexible and adaptive advice to human decision-makers in an agent-based architecture. The design of an extension to FIPA ACL has been presented as an example of one such competence model.

We are currently studying the application of the aforementioned models in different domains. One of line of action relates to the construction of *Personal Trading Assistants (PTAs)*, software agents that assist non-experts in performing transactions in electronically mediated markets. Its primary focus is on small-scale investors that participate in financial markets through Internet, as the recent boom of such transactions has exceeded the capacity of many brokerage agencies to provide individualized advice through human expert brokers. We are also proceed with previous research on DSS for road traffic management. In this respect, we will apply intelligent assistants, based on multiagent models, to real-world problems of bus fleet management and road traffic control in Malaga and Bilbao respectively. Setting out from these application scenarios we intend to abstract a reference architecture, as well as a software repository, so as to allow for the principled and efficient construction of personal assistant agents for advanced decision support.

References

[1] Cuena, J.; Ossowski, S. (1999): Distributed Models for Decision Support. *Multi-agent Systems — A Modern Approach to DAI* (Weiss, editor), MIT Press

[2] Foundation for Intelligent Physical Agents. *FIPA Communicative Act Library Specification.* FIPA - http://www.fipa.org, 2000.

[3] French, S. (2000): *Decision Analysis and Decision Support.* John Wiley & Sons

[4] Hernández, J.; Ossowski, S.; García-Serrano, A. (2001): Multiagent Architectures for Intelligent Traffic Management Systems. To be published in *Transportation Research C*, Kluwer

[5] Klein, M.; Methlie, L. (1995): *Knowledge-Based Decision Support Systems.* John Wiley & Sons

[6] Kobsa, A. (2001): Generic User Modelling Systems. To be published in the *Int. Journal on User Modelling and User-Adapted Interaction*, Kluwer

[7] Maybury, M.; Wahlster, W. (1998): *Readings in Intelligent User Interfaces*, Morgan Kaufmann

[8] Molina, M.; Ossowski, S.(1999): Knowledge Modelling in Multiagent Systens — The Case of the Management of a National Network. *Intelligence in Services and Networks* (Zuidweg, editor), Springer-Verlag

[9] Müller, J. (1998): The Right Agent Architectur to Do the Right Thing.*Intelligent Agents IV* (Müller, Singh y Rao, editores), Springer-Verlag

[10] Ossowski, S.; García-Serrano, A. (1996): A Model of Co-ordination Support for Unanticipated Situations. *Advanced IT Tools* (Terashima y Altman, editores), Chapman & Hall

[11] Ossowski, S.; García-Serrano, A. (1999): Social Structure in Artificial Agent Societies — Implications for Autonomous Problem-Solving Agents. *Intelligent Agents IV* (Müller, Singh y Rao, editores), Springer-Verlag

[12] Ossowski, S. (1999): *Co-ordination in Artificial Agent Societies.* LNAI 1535, Springer-Verlag

[13] M. D. Sadek. Dialogue acts are rational plans. In *Proceedings of the ESCA/ETRW Workshop on the structure of multimodal dialogue*, pages 1–29, Maratea, Italie, 1991.

[14] M. D. Sadek. A study in the logic of intention. In William Nebel, Bernhard; Rich, Charles; Swartout, editor, *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, pages 462–476, Cambridge, MA, October 1992. Morgan Kaufmann.

[15] M. D. Sadek, P. Bretier, and F. Panaget. ARTIMIS: Natural dialogue meets rational agency. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 1030–1035, San Francisco, August 23–29 1997. Morgan Kaufmann Publishers.

[16] Serrano, J.M., Ossowski, S. (2000): A Domain Extension to FIPA-ACL — An Application to Decision Support Systems. Proc. *Simposio Español de Informática Distribuida*, Univ. de Oviedo

[17] Serrano, J.M., Ossowski, S. (2001): Illocutionary Actions for Artificial Assistant Agents. To be pusblished in the proc. of the *7th Int. Colloquium on Cognitive Science*, Univ del Pais Vasco.

[18] Silver, M. (1991): *Systems that Support Decision Makers - Description and Analysis.* John Wiley & Sons

Sesión 1: Sistemas Multiagentes

A Multi-Agent System for Planning Meetings

Santiago Macho, Marc Torrens, and Boi Faltings

Artificial Intelligence Laboratory (LIA)
Swiss Federal Institute of Technology (EPFL)
IN-Ecublens, CH-1015, Lausanne
Switzerland
e-mail: {akira,torrens,faltings}@lia.di.epfl.ch

Abstract In this work we address the problem of arranging meetings for several participants taking into consideration constraints for personal agendas, transportation schedules and accommodation availability. We have implemented a multi-agent system that solves the problem.
Building such applications implies to consider two main issues: collecting information from different sources on the Internet, and solving the problem itself. We show that multi-agent systems that use constraint satisfaction for modelling and solving problems can be very suitable for this kind of systems.

1 Introduction

In this paper we address the problem of arranging meetings among several people. The problem involves combining personal agendas with transportation schedules and accommodation availability in order to find appropriate meeting places, dates and times.

The traditional way of arranging meetings for several participants implies a negotiation by hand of dates and sometimes also places. Every participant has an agenda with some available dates for the meeting. The task of taking a decision for a meeting is not an easy problem, specially in the case that: the participants are quite busy and/or the meeting takes several days and/or they live in different places, etc. The problem becomes more complex if we consider transportation schedules and people have several meetings with different people in different places. The problem could be even more difficult to solve if we take into consideration user's preferences, where every participant has different criteria. In such situations it is mostly impossible to plan meetings in an optimal way by hand.

Basically, our problem is naturally a choice problem. Participants to meetings have to choose among several options. These choices cannot be taken freely because many elements are interconnected, i.e. there are dependencies and incompatibilities between the different choices to take. Considering these factors, the problem of arranging meetings according to transport constraints and personal preferences can be easily formulated as a Constraint Satisfaction Problem (CSP).

With the information about transportation schedules and accommodation availability by neutral travel providers and the possibility of interaction among agents on the Internet, the task can be mostly done automatically and thus it could become a

useful tool for people. On the other hand, the application demonstrates the utility of using constraint satisfaction for solving complex problems in multi-agent systems.

In order to build up a multi-agent system for planning meetings, we suppose that every participant has an agenda which is accessible by an `Personal Assistant Agent`. Another agents are the `Flight Travel Agent` that has access to the schedules and availability of flights around the world and the `Accommodation Hotel Agent` that has information about hotels in the world. Such kind of agents are described in [1]. These two agents are implied in the process of collecting information. All the information implied in the system and the problem modelling are formalised using constraint satisfaction formalism, so it is ubiquitous that the agents communicate each other using a FIPA[1] compliant language called Constraint Choice Language (CCL[2]) [2]. The solving task which is basically to solve a configuration problem is carried out by constraint satisfaction algorithms implemented in the Java Constraint Library (JCL[3]) [3].

In the next section we describe how to formalise our problem using Constraint Satisfaction Problems. Then, we show how to solve the problem using information gathering on the Web and constraint satisfaction techniques under the multi-agent framework. Next section is intended for describing the multi-agent architecture for our system, more concretely: the Constraint Choice Language (CCL), the different agents and the interaction between them. Then, we point out further work and we finish the paper giving some conclusions.

2 Problem Modelling

The problem of arranging meetings is formulated in our framework as a CSP. In the following subsection, we briefly describe CSPs and then we present a concrete way to model our problem by identifying the main components of such formulation.

2.1 Constraint Satisfaction Problems (CSPs)

Constraint Satisfaction Problems (CSPs) (see [4]) are ubiquitous in applications like configuration [5, 6], planning [7], resource allocation [8, 9], scheduling [10] and many others. A CSP is specified by a set of variables and constraints among them. A solution to a CSP is a set of value assignments to all variables such that all constraints are satisfied. There can be either many, 1 or no solutions to a given problem. The main advantages of constraint-based programming are the following:

– It offers a general framework for stating many real world problems in a succinct, elegant and compact way.

[1] Foundation for Intelligent Physical Agents: `http://www.fipa.org`

[2] Constraint Choice Language: `http://liawww.epfl.ch/CCL`

[3] Java Constraint Library: `http://liawww.epfl.ch/~torrens/JCL`

- A constraint based representation can be used to synthesise solutions of the problem as well as for verification purposes (i.e. showing that a solution satisfies all constraints).
- The nature of the representation allows a formal description of the problems as well as a declarative description of search heuristics.

Formally, a finite, discrete Constraint Satisfaction Problem (CSP) is defined by a tuple $P = (X, D, C)$ where $X = \{X_1, \ldots, X_n\}$ is a finite set of variables, each associated with a domain of discrete values $D = \{D_1, \ldots, D_n\}$, and a set of constraints $C = \{C_1, \ldots, C_l\}$. Each constraint C_i is expressed by a relation R_i on some subset of variables. This subset of variables is called the *connection* of the constraint and denoted by $con(C_i)$. The relation R_i over the connection of a constraint C_i is defined by $R_i \subseteq D_{i1} \times \ldots \times D_{ik}$ and denotes the tuples that satisfy C_i. The *arity* of a constraint C is the size of its connection.

2.2 The Problem of Arranging Meetings as a CSP

The problem of arranging meetings can be formulated as a choice problem, more specifically as a Constraint Satisfaction Problem (CSP). For simplicity, we consider that we have to plan only one meeting among several participants that lives in different places. Three phases are needed in order to model a problem as a CSP[4]:

1. *Variables:* identify the variables involved in the problem,
2. *Domains:* associate to all variables the appropriate finite domain of discrete values, and
3. *Constraints:* link the constrained variables by means of allowed/disallowed combinations of values.

In our framework, there is a set of n participants ($P = \{P_0, \ldots, P_{n-1}\}$). The meeting has to take place when all the participants P_i are available. In addition, the meeting will be in a set of m possible predefined cities ($C = \{C_0, \ldots, C_{m-1}\}$). Normally, this set of places corresponds to the places where the involved people live.

For each participant P_i we define his/her agenda as a set of k `AgendaSlot` ($AS = \{AS_0, \ldots, AS_{k-1}\}$). An `AgendaSlot` is defined as a `StartSlotTime`[5], an `EndSlotTime`, and a `SlotPlace`.

Next, we identify the variables for our model, what are the associated domains and what kind of constraints the system has to take into consideration. The model has been simplified for a better comprehension of the formalism.

---

[4] in our framework, we refer CSPs as finite and discrete CSPs.

[5] when we refer to *Time* variables, we include for each value an exact time meaning an hour, day, a month, a year, etc...

Variables The variables of the CSP depend on the solution we want to find out. In our case, for each participant (P_i) to the meeting we are interested in: an `OutgoingFlight`$_i$ and a `ReturnFlight`$_i$. For every participant P_i exists three variables for each free `AgendaSlot` in his agenda: `StartFreeTime`$_j$, `EndFreeTime`$_j$ and `Place`$_j$.

Other variables concern the meeting itself: the `MeetingPlace`, the `StartMeetingTime` and the `EndMeetingTime`.

Domains For variables `OutgoingFlight`$_i$ and `ReturnFlight`$_i$ the domains are possible flights the participant P_i can take to attend the meeting. At the beginning of the solving process, the system does not know explicitly what are the possible flights for such variables, these domains are only known once the system starts solving the problem and after querying the flight database.

Concerning the variables of the type `Start/EndFreeTime`$_j$ and `Place`$_j$, the values are retrieved from the corresponding agendas for every participant and for each free time slot.

The values of the variables concerning the meeting itself are known a priori. In some sense, these values define the problem to solve. For example, if we want to plan a meeting, normally the problem can be stated as:

> "We want to meet next month, from 15^{th} to 23^{rd} in some of the places we live. The meeting will take place during 3 days. We can meet on Saturdays but not on Sundays".

From such a formulation, the system deduces the domains of the variables related to the meeting. In other words, these variables define the problem the system has to solve.

Constraints Constraints are used for defining the search space and thus the solving algorithms will find well defined solutions. Basically, the constraints involved in our problem are:

- `OutgoingFlight`$_j$`-ReturnFlight`$_j$: The return flight has to be taken after the outgoing flight. The arrival place of the outgoing flight must be the same than the departure of return flight.
- `OutgoingFlight`$_j$`-Place`$_j$: The departure of the outgoing flight must be the same place as `Place`$_j$.
- `OutgoingFlight`$_j$`-MeetingPlace`: The outgoing flight must arrive at the place where the meeting will take place.
- `OutgoingFlight`$_j$`-StartMeetingTime`: All the participants must arrive before the meeting starts.
- `ReturnFlight`$_j$`-EndMeetingDate/Time`: All the participants must leave the meeting place after the meeting has finished.
- `StartFreeTime`$_k$`-StartMeetingTime`: Obviously, for each user, there must exist at least one `StartFreeTime`$_k$ which is before the `StartMeetingTime`.

- `EndFreeTime`$_k$`-EndMeetingTime`: For each user, the `EndFreeTime`$_k$ must be after the `EndMeetingTime`.
- `StartFreeTime`$_k$`-EndFreeTime`$_k$: `EndFreeTime`$_k$ must be after `StartFreeTime`$_k$. With this constraint we guarantee that the free time slots are well defined.

3 Problem Solving

In our framework, problem solving is mainly composed of two phases, gathering information and finding solutions:

- Gathering information: every `Personal Assistant Agent` collects information from different agents in order to model the corresponding CSP. Some domains of the variables are filled in by means of queries to the involved agents. The `Personal Digital Agent` is responsible for requesting to *information agents* the needed information in the appropriate order to build the whole CSP. *Information agents* are, for example, the agent that collects information about the availability of rooms in a hotel (`Accommodation Hotel Agent`) or the agent that requests schedules and availability of flights (`Flight Travel Agent`).
- Finding solutions: once that every `Personal Assistant Agent` involved in the meeting build the CSP, they can apply constraint satisfaction algorithms from the JCL and find solutions according to the constraints. The `Personal Assistant Agent` that proposed the meeting receives the solutions of the others `Personal Assistant Agents` involved in the meeting, and compares them, informing the user about the solution that satisfies all the constraints of the CSPs.

4 The Multi-Agent Architecture

The multi-agent system is composed by the following agents (see Fig. 1):

- `Personal Assistant Agent`: is the interface agent between the users and the multi-agent system.
- `Flight Travel Agent`: is connected to a database of flights over the world.
- `Accommodation Hotel Agent`: is the agent responsible to find an accommodation on the cities involved in the meeting.

Every agent have different data. A `Personal Assistant Agent` can organise a meeting or participe giving a partial solution based on the availability of its user. The other agents will complete the data needed to organise the meeting. They cannot interact directly with the users.

In our system, a user that propose the meeting plan is charged of inputing the main parameters of the meeting, such as:

- the users willing to attend the meeting,
- how long the meeting will take,
- in what range of dates the meeting must be planned, and

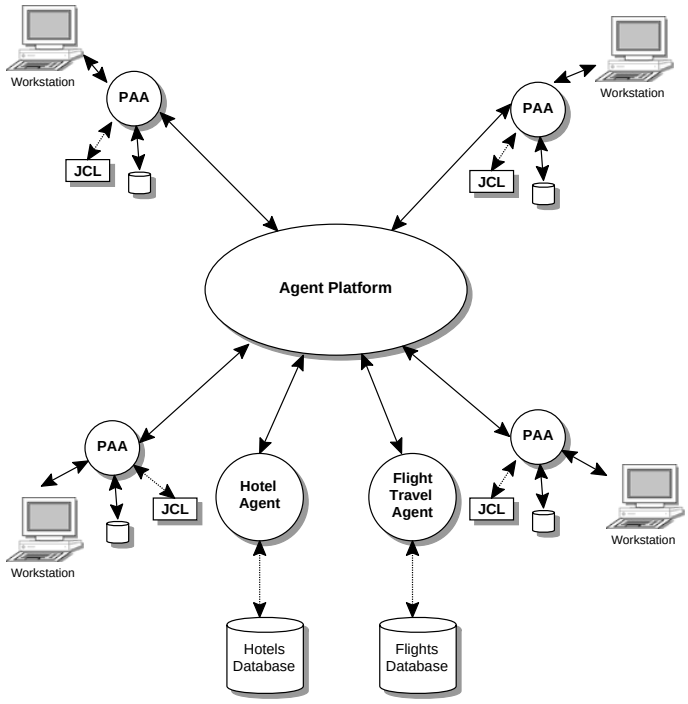


Figure1. The multi-agent architecture.

– where the meeting can take place (it is possible to give several optional places).

This user uses the `Personal Assistant Agent` to input the data into the multi-agent system. When the `Personal Assistant Agent` has all the data about the meeting we want to plan, it builds the associated CSP using CCL[6]. This agent is the responsible to add data to the CSP using the personal agenda of the user. Variables concerning the hotels availability and variables concerning the flight schedules are not yet present in the CSP. Then, the `Personal Assistant Agent` sends the CSP to the `Accommodation Hotel Agent` to get constraints about hotel availability in the city of the meeting. A similar process is carried out for getting the information about flight schedules between the cities of the user and the city of the meeting. At this point, the CSP is ready to be solved. The `Personal Assistant Agent` has to perform two phases: firstly it translates the CSP, written in the CCL, to the data structures of the JCL, and secondly it uses the JCL algorithms to find solutions to the problem. Solutions only deal with the constraints of the user that proposed the meeting without deal with the constraints of the other users involved in the meeting. The other `Personal Assistant Agents` involved in the meeting do the same process: query the agenda of their users, send the CSP to complete it with constraints about hotels and flights and finally solve it. When the `Personal Assistant Agents` have the solutions, they send them to the `Personal Assistant`

[6] CCL = Constraint Choice Language. See next section

`Agent` that proposed the meeting. This agent compares the solutions from the others agents with its solution, and show to the user solutions that satisfies the constraints of all the users. In the case that the meeting proposition is accepted, the agenda is updated according to the new meeting. It sends a message to the other agents involved in the meeting with the solution chosen by the agent that proposed the meeting.

Our multi-agent system uses ACL message for interacting. As a content language of the ACL messages, we use the Constraint Choice Language (CCL).

4.1 Constraint Choice Language (CCL) [2]

CCL is a FIPA compliant content language based on Constraint Satisfaction techniques. The CCL specification includes semantic foundations, abstract syntax and language ontology. CCL:

- is based on constraint satisfaction formalism,
- is suitable for choice problems or CSPs,
- supports communication about CSPs from modelling right through to problem solving, and
- has been incorporated in the FIPA 1999 standard as content language FIPA-CCL.

A traditional way to formulate constraints in discrete CSPs is to define the tuples by an explicit list of allowed or excluded values between the implied variables. The Constraint Choice Language deals with a slight different notation which simplifies the implementation of constraint engines. In particular, it allows us to express two types of constraints:

- *Exclusion constraints*, which act on a single variable and are specified as a *no-good list*.
- *Relations*, which act on two variables and are restricted to a closed set of seven general types ($=$, $\neq$, $<$, $>$, $=<$, $>=$, and *goodlist*), but can be formulated on tuples.

The use of tuple-valued variables allows the language to handle n-ary constraints by introducing variables whose values represent the tuples allowed by the constraint. The advantage of this formulation is that solving or consistency engines can be restricted to unary and binary constraints.

An example about how to express CSPs using XML CCL allows agents to express a CSP as we defined it in section 2.1. However, three restrictions on the CSP representation have been made to make the model minimal and more suitable for a communication language:

1. *Binary constraints*: all constraints must be binary, i.e. constraints that involves two variables. This restriction is often made in the CSP community, since most

powerful solving techniques only apply to binary CSPs. However, this is only a slight restriction because we can transform n-ary constraints to binary constraints.

2. *Discrete variable domains*: most of the real-world problems like configuration, scheduling or planning can be formulated using CSPS that have variables with domains that contain discrete values. For example, suppose that we want to write in CCL the fact that the variable `MeetingPlace` of the type `String` has as domain the values: {`Zurich, Geneva, Amsterdam`}. This variable would be expressed in CCL as follows:

```
<CSP-variable Name="MeetingCity" Type="string">
    <Domain Type="String">
        <CSP-value-list>
            <List>
                <li> Zurich </li>
                <li> Geneva </li>
                <li> Amsterdam </li>
            </List>
        </CSP-value-list>
    </Domain>
</CSP-variable>
```

3. *Intensional relations*: instead of working only with extensional relations between two variables (`good-list` or `no-good-list`) we also work with intensional relations (=, $\neq$, <, >, <=, >=). In this way, we facilitate the merge of CSP when collecting information from several sources.

A CSP expressed in CCL is composed by a zone of variables and a zone of constraints (relations), for example:

```
(request
:sender FrontBot@iiop://liasun24.epfl.ch:7999/acc
:receiver planner@iiop://liasun24.epfl.ch:7999/acc
:content  #3318<?xml version="1.0"?>
<!DOCTYPE Expression SYSTEM "CCL.dtd">
<Expression>
    <Action Name="CSP-solve">
        <CSP-solve>
            <CSP CSP-ref="id939978811875 ">
                <!-- ZONE OF VARIABLES >
                <CSP-variable Name=...
                        ...
                </CSP-variable>
                <CSP-variable>
                        ...
                </CSP-variable>
                        ...
                <!-- ZONE OF CONSTRAINTS >
                <CSP-relation Variables=...
                        ...
                </CSP-relation>
                <CSP-relation Variables=...
                        ...
```

```
                </CSP-relation>
                        ...
            </CSP>
        </CSP-solve>
  </Action>
</Expression>
:language FIPA-CSP
:protocol fipa-request
:conversation-id liasun24.epfl.ch/128.155.6312188 )
```

In the following sections, we briefly describe how each agent of the multi-agent system works.

4.2 Personal Assistant Agent

The `Personal Assistant Agent` is the agent that interfaces between the user and the multi-agent system for planning meetings and travels. With this agent, the user expresses his/her needs and preferences for the meeting. The `Personal Assistant Agent` builds first a CSP with only a few variables (`StartMeetingTime`, `EndMeetingTime`, `MeetingPlace`, etc) and some constraints extracts from the agenda of the user. Then, it asks for variables and constraints to the *information agents*: the `Accommodation Hotel Agent` (variables and constraints about accommodation availability) and the `Flight Agent` (variables and constraints about flights). When the `Personal Assistant Agent` has all the necessary variables, it uses the constraint satisfaction algorithms of JCL for solving the CSP. It also updates the agenda when a new meeting is planned. The user can modify the meeting using the `Personal Assistant Agent`. If he/she modifies the data of the meeting planified, then the system must replan the meeting starting the process with the agents involved in it.

4.3 Flight Travel Agent

The `Flight Agent` is an agent that is connected to the database of flights (schedules and availability) provided by a neutral travel provider such as Galileo. This agent offers information services about all the flights over the world. It is proactive, if a flight is cancelled due the weather or another problems, it sends a message to all the `Personal Assistant Agents` to replan meetings if necessary.

4.4 Accommodation Hotel Agent

The `Accommodation Hotel Agent` is an agent that is connected to the database of hotels in order to book a hotel in the city of the meeting. Also as the `Flight Travel Agent` the database could be provided by a neutral provider.

4.5 Interaction between agents

The agents of the multi-agent system for planning meetings are FIPA ACL compliant. In the next subsections we focus on the interaction between the agents. Firstly we show an overview of ACL messages and finally we show the following interactions:

- Personal Assistant Agent - Flight Travel Agent interaction,
- Personal Assistant Agent - Accommodation Hotel Agent interaction and
- Personal Assistant Agent$_A$ - Personal Assistant Agent$_B$

Overview of ACL messages The FIPA Agent Communication Language (ACL) is based on speech act theory: messages are actions, as they are intended to perform some action by virtue of being sent. The specification consists of a set of message types and the description of their pragmatics, that is the effects on the mental attitudes of the sender and receiver agents. Every communicative act is described with both a narrative form and a formal semantics based on modal logic [11].

In the FIPA ACL specification there is the description of some high-level protocols like request, contract net, several kinds of auctions, etc. Our multi-agent system uses the inform and the request protocol shown in Fig. 2. With the inform protocol an agent informs another agent about a situation. With the request protocol, an agent requests another agent to perform an action, and the receiver agent is able to perform it or replay that it can not do it.

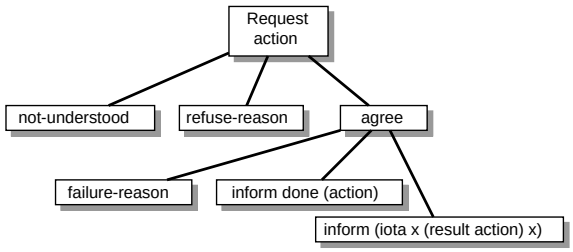


Figure2. The FIPA ACL request protocol.

The content field of an ACL message contains the expression (action, proposition or object) and the object (CSP, solution-list, etc) which is referred by the expression, all codified in CCL.

Personal Assistant Agent - Flight Travel Agent interaction The `Personal Assistant Agent` needs to add to the CSP the values and constraints about flights. When the `Flight Travel Agent` receives the message `CSP-give-constraints` with the CSP, it searches to the flight database the available flights for the user to go to the city of the meeting.

We use the FIPA-request protocol, so the possible answers from the `Flight Travel Agent` to the `Personal Assistant Agent` are the FIPA ACL messages: `not-understood`, `refuse`, or `agree`.

Fig. 3 shows the interaction between the `Personal Assistant Agent` and the `Flight Travel Agent`.

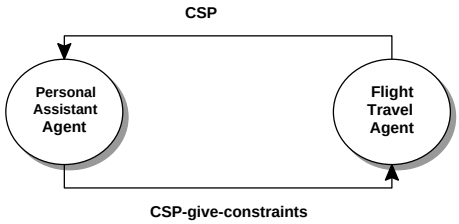


Figure3. The interaction between the `Personal Assistant Agent` and the `Flight Travel Agent`.

Personal Assistant Agent - Accommodation Hotel Agent interaction The interaction between the `Personal Assistant Agent` and the `Accommodation Hotel Agent` is similar to the interaction described above. The `Personal Assistant Agent` needs to add to the CSP values and constraint about hotels. The `Accommodation Hotel Agent` searches in the hotels database the availability of rooms in the city of the meeting.

Fig. 4 shows the interaction between the `Personal Assistant Agent` and the `Accommodation Hotel Agent`.

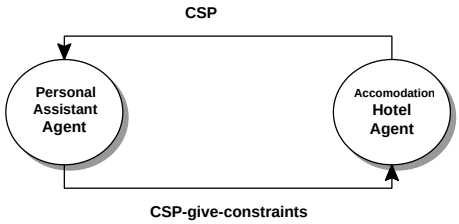


Figure4. The interaction between the `Personal Assistant Agent` and the `Accommodation Hotel Agent`.

Personal Assistant Agent$_A$ - Personal Assistant Agent$_B$ interaction The `Personal Assistant Agent`$_A$ sends a `request` message with the action `CSP-solve` to the `Personal Assistant Agent`$_B$[7]. This message starts a new conversation between the `Personal Assistant Agent`$_A$ and the `Personal Assistant Agent`$_B$.

The `Personal Assistant Agent`$_A$ sends a `request` message to the `Personal Assistant Agent`$_B$ with the action `CSP-Solve` in order to start the solving process in the `Personal Assistant Agent`$_B$. When it has the solution(s), it sends back the object `CSP-Solution` if there is only one solution or the object `CSP-SolutionList`

[7] To simplify the example we suppose that there are only 2 users involved in the meeting. If there are more users, the `Personal Assistant Agent`$_A$ sends a `request` message to every participant

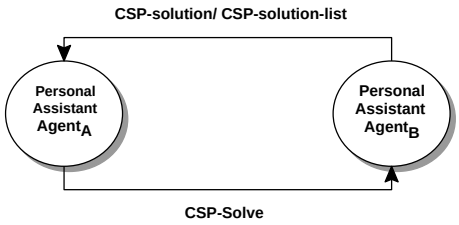


Figure5. The interaction between the `Personal Assistant Agent`$_A$ and the `Personal Assistant Agent`$_B$.

if there are more than one solution in order to compare with the solutions found by the `Personal Assistant Agent`$_A$.

Next subsection describes the Java Constraint Library.

Java Constraint Library We implemented the Java Constraint Library (JCL), which allows us to package constraint satisfaction problems and their solvers in compact autonomous agents suitable for transmission on the Internet. It provides services for:

- creating and managing discrete CSPs
- applying preprocessing and search algorithms to CSPs

JCL can be used either in an applet[8] or in a stand-alone Java application. The purpose of JCL is to provide a framework for easily building agents that solve CSPs on the Web. JCL is divided into two parts: A basic constraint library available on the Web and a constraint shell built on the top of this library, allowing CSPs to be edited and solved.

5 Further Work

Many extensions to this work are planned. An interesting future research topic will be how to combine the *gathering information* phase with the *solving problem* phase dynamically. The multi-agent system could start solving the problem without having all the information in the CSP. Then, the system would collect information when being completely necessary. This idea implies to solve the problem dynamically when searching information. The advantage would be that we will not collect unnecessary information, and the user could get some first solutions very quickly.

Another interesting issue is how to learn from previous experiences. In the `Personal Assistant Agent` we could have a user profile with a set of predefined preferences (constraints) that will be taken into consideration for next meeting plans.

[8] An *applet* is an application designed to be transmitted over the Internet and executed by a Java-compatible Web browser.

In the future, we also want to deal with several different kinds of transportation agents (not only flights). In this way, we will be able to plan travels and meetings using different transport means. In this direction, a project with train schedules is already on the track.

Once, the system finds several possible solutions to the problem, users have to choose one of them. This process can be very tedious and difficult since people tend to prefer different options. For avoiding to perform this process manually, we will study some negotiation issues related to multi-agent systems in order to apply such techniques to our framework.

A possible application of our multi-agent system is as a portable planning. Mobile phones of the next generation could be programmed and we could integrate the `Personal Assistant Agent` inside a mobile phone, having a powerful planning between different users (see Fig. 6).

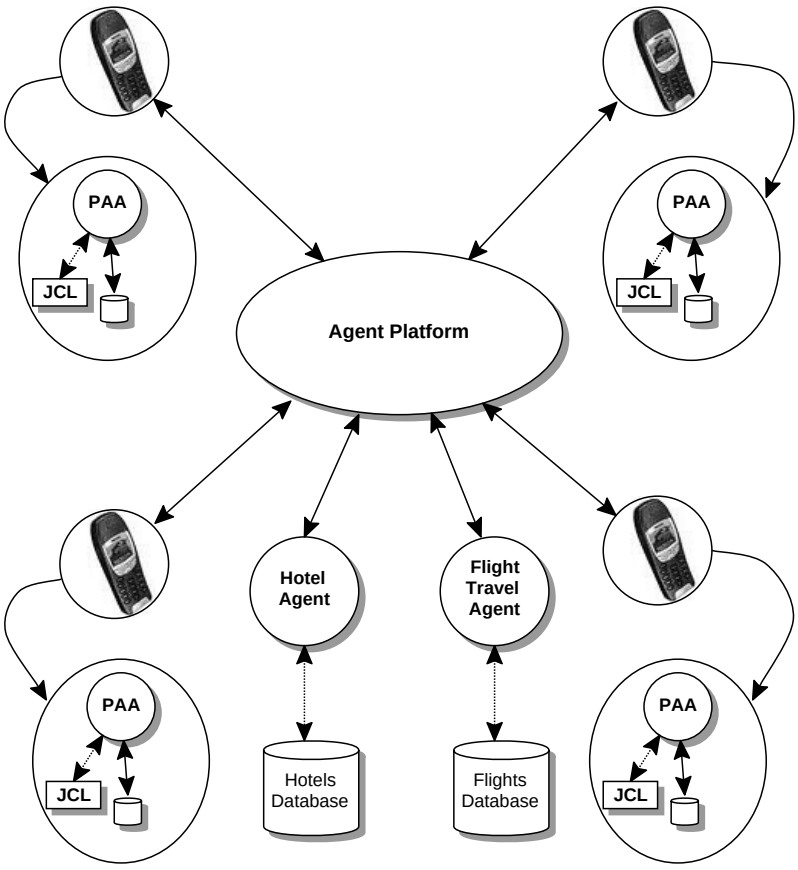


Figure6. An application of our multi-agent system.

6 Conclusions

In this paper, we have shown that constraint techniques can be very useful for solving complex problems addressed by multi-agent systems such as the problem of arranging meetings and scheduling travels.

Concretely, we have implemented a multi-agent system which is able to plan meetings using agenda's information, availability of hotels1 and transportation schedules. Agents communicate each other using the Constraint Choice Language (CCL), a FIPA compliant content language for modelling problems using constraint satisfaction formalism. With this system, we also have shown the utility of using the Java Constraint Library (JCL) for solving complex problems in multi-agent systems.

7 Acknowledgements

We thank Steve Willmott and Monique Calisti for the useful comments on modelling the problem of arranging meetings and travels as a CSP, and for building up the Constraint Choice Language in the FIPA framework.

References

[1] Marc Torrens and Boi Faltings. Smart clients: Constraint satisfaction as a paradigm for scaleable intelligent information systems. In *Working Notes of the Workshop on Artificial Intelligence on Electronic Commerce, AAAI-99*, Orlando, Florida, USA, 1999.

[2] Steve Willmott, Monique Calisti, Boi Faltings, Santiago Macho Gonzalez, Omar Belakdhar, and Marc Torrens. CCL: Expressions of Choice in Agent Communication. In *The Fourth International Conference on MultiAgent Systems (ICMAS-2000)*, Boston, USA, 2000.

[3] Marc Torrens, Rainer Weigel, and Boi Faltings. Java Constraint Library: bringing constraints technology on the Internet using the Java language. In *Working Notes of the Workshop on Constraints and Agents, Tehnical Report WS-97-05, AAAI-97*, Providence, Rhode Island, USA, 1997.

[4] Edward Tsang. *Foundations of Constraint Satisfaction.* Academic Press, London, UK, 1993.

[5] Felix Freyman Sanjay Mittal. Towards a generic model of configuration tasks. In *Proceedings of the 11^{th} IJCAI*, pages 1395–1401, Detroit, MI, 1989.

[6] Daniel Sabin and Eugene C. Freuder. Configuration as Composite Constraint Satisfaction. In *Proceedings of the Artificial Intelligence and Manufacturing Research Planning Workshop*, pages 153–161, 1996.

[7] Mark Stefik. Planning with constraints (MOLGEN: Part 1). *Artificial Intelligence*, 16(2):111–140, 1981.

[8] Berthe Y. Choueiry. *Abstraction Methods for Resource Allocation.* PhD thesis, Swiss Federal Institute of Technology in Lausanne, 1994.

[9] A. Sathi and M. S. Fox. Constraint-Directed Negotiation of Resource Allocations. In L. Gasser and M. Huhns, editors, *Distributed Artificial Intelligence Volume II*, pages 163–194. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1989.

[10] Mark Fox. *Constraint-Directed Search: A Case Study of Job-Shop Scheduling.* Morgan Kaufmann Publishers, Inc., Pitman, London, 1987.

[11] Foundation for Intelligent Physical Agents (FIPA). FIPA Agent Specification 1997. In *Technical Report*, 1997.

Examples of Dynamical Physical Agents for Multi Robot Control

B. Innocenti, J.Ll. de la Rosa, J. Vehí, I. Muñoz, M.Montaner, M. Fàbregas, A. Figueras, and J.A. Ramón

Institut d'Informàtica i Aplicacions
Universitat de Girona
e-mail: {bianca,peplluis,vehi,imunoz,mmontane,marti,figueras,jar}@eia.udg.es

Abstract Lastest trends on Artificial Intelligence (AI) lead to combine AI with the traditional Control Theory to obtain intelligent systems. The goal of this work is to find some parameters that describe dynamics of the physical body of any agent, and to use them in a decision algorithm to let the agent know about its physical limitations. As a first approach, dynamics are described for single input-single output (SISO) systems. The parameters should be generics, comparable and understandable to both, the agent (computationally treatable) and the human being.

1 Introduction

Nowadays some Artificial Intelligence (AI) techniques are being applied to control complex systems. Since Brooks [2] and Zhang [11] stated that the intelligence depends on the interactions with the environment, several researchers have been trying to combine AI tools with traditional Control Theory with the aim of developing intelligent robots.

New tendencies lead to control complex systems using agents and to consider the whole process as a multi-agent system that needs co-ordination and co-operation to obtain the desired results. One language that allows programming agents is Shoham's AGENT0 [10]. In this language the state of an agent consists of components such as capabilities (things that the agent can do), beliefs (beliefs of the world, itself or other agent), commitments (commitments with other agents or itself) and commitment rules (settle how the agent acts). A commitment rule can be as the following:

```
COMMIT(
(agent,REQUEST,DO(action, time)) ⟶ message condition
(B,[now,Friend agent] AND
 CAN(self, action) AND
 NOT[time,CMT(self, no_action)]), ⟶ mental condition
 self, DO (time, action))
```

This rule can be read as:
*I*f I receive a message from **agent** which **requests** me to **do action** at **time** and I believe (**B**) that:

- agent is currently a friend
- I can do the action

– at time I'm not committed (not cmt) to do any other action, then commit to do action at time.

With the aim of achieving its commitments, an agent must check whether they are feasible or not. So before committing it compares the required action with its beliefs and capabilities. When an agent has a physical body, not only does the performance of an action depend on the dynamics of this physical body but also that what is heuristically possible to do may result in non-desired consequences.

Recalling the capabilities represent the actions that the agent can do, they seem the appropriated *mental state* to represent the dynamics of the physical body.

The goals of this work are to find some kind of co-ordination between the AI techniques and the Control Theory and to analyse the behaviour of the whole system. The proposal here is to include some features of the dynamics of the agent physical body into the decision algorithm to get secure and reachable commitments.

This paper is organised as follows. Section 2 explains what physical agents are. Section 3 resumes the relevant aspects of the agent architecture used in this work. Atomic capabilities attributes are defined in section 4. Section 5 presents an example of fulfilling atomic capabilities. And section 6 concludes.

2 Physical Agents

According to Asada [asada], the meaning of having a physical body can be summarised as follows:

– Sensing and acting capabilities are not separable, but tightly coupled.
– In order to accomplish a given task, the sensor and actuator spaces should be abstracted under resource-bounded conditions (memory, processing power, controller, etc.).
– These abstractions depend on the interactions of the agent with the environment.
– The consequence of the abstraction is agent-based subjective representation of the environment.
– In the real world, both inter-agent and agent-environment interactions are asynchronous, parallel and arbitrary complex.
– Natural complexity of physical interactions automatically generates reliable sample distribution of input data for learning.

Based on these statements, researchers have developed several architectures to control robots. As examples, it can be mentioned among others the Brooks' Subsumption Architecture [brooks2] and Zhang and Mackworth's Constraint Net (CN) [zhang2]. In the former the reactive agents have a layered set of different behaviours that compete to take the robot control. In the later the robot, its controller and the environment are modelled as three different machines with input and output modules; based on this CN and the properties required for the controller, specified

as a set of constraints, it's possible to automatically generate a controller with the desired specifications.

There are also several hybrid architectures that include reactive and deliberative behaviours as the Oller's Dynamical Physical Agent Architecture (DPAA) [7].

3 Dynamical Physical Agent Architecture

The DPAA has been developed for physical agents and has several requirements that are built-in and that enables the agents to work in a real world, in real-time. Some of them are:

- Situated behaviour: agents must recognise asynchronous events and react both on time and in a proper way taking into account its physical body.
- Goal-oriented behaviour: agents must choose actions based on the whole system objectives and on its own.
- Efficiency: tasks must be executed efficiently considering the real physical odds that agents have to achieve them.
- Co-ordination: agents must keep in mind the positive and negative interactions with other agents.

To deal with all of these requirements, it is proposed the architecture shown in Figure 1

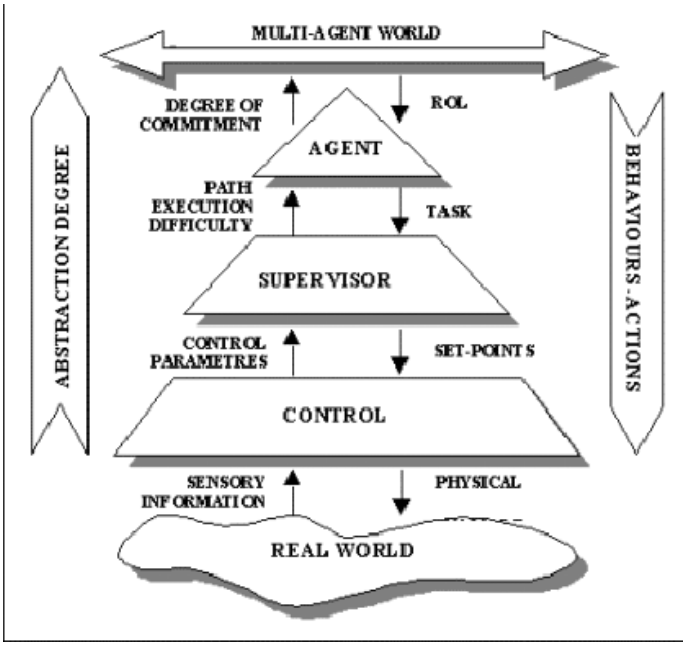


Figure1. DPAA Architecture

The DPAA is a layered architecture formed by three specialised modules:

- Control module: it's the direct connection of the agent with the real world.
- Supervisor module: it's the interface among the parameters of the agent real world with the agent logic world.
- Agent module: it's the connection with the multi-agent world.

It can be seen in the figure the different layers of the architecture, the increase of information abstraction degree as layers became logical and the decrease of the behaviour-actions as layers go to the real world.

After negotiating with other agents and in order to make a decision, an agent must check some external and internal parameters. The agent can get the external ones from the other agents by exchanging information. On the other hand, the internal ones must describe the different states of the agent physical body, both in low and high levels.

In order to include these internal parameters in the agent capabilities, three different kinds of capabilities depending on the abstraction level of the information are proposed, which are:

- Atomic capability: it contains information about the agent physical body, the perception of the environment through its body and the agent adaptation to the environment (learning).
- Basic capability: formed by several atomic capabilities, represents the knowledge of the supervisor. Information starts to be symbolic.
- Symbolic capability: it contains an abstract model of the world. Information is symbolic and depends on the agents' application.

Before deciding, an agent looks up in its set of capabilities and if they have enough information about the pretended action, the agent accepts or rejects the proposal. In case that the information is not complete, the agent communicates with the immediate lower level, and so on, to get it. In this way, when the agent accepts the action, it is aware of what its physical body can do with a high level of certainty.

In this approach, the parameters that formed the atomic capabilities turn out to be incomplete, so the contribution of this work is to complete them.

4 Atomic Capabilities

When trying to complete the atomic capabilities, two questions arise: What kind of information should the capabilities contain? Which parameters computationally treatable by the agent are the ones that best represent the dynamics of its physical body?

In order to obtain the characteristics of the dynamics of a physical agent, it has been necessary to do a complete study of the responses of different controlled

systems. Because of the complexity of studying real systems in Control Theory [4] [5] [6] [8] [9], the scope of the present work has been limited to SISO systems.

In Control Theory before designing the controller for a system, Control Engineers should know the specifications that the response of the system must achieve. These specifications describe the response of the controlled system, so they can be used to complete the atomic capabilities. But this information has been modified in order to accomplish some requirements such as:

- Knowledge contained in the capabilities must be general. That means capabilities can be completed for any controlled SISO system.
- Atomic capabilities must be comparable between them. This implies that the parameters must be independent from the input, kind of system and between them. They must also ensure that the comparison is suitable.
- Information must allow computationally treatment to be understandable by the agent.
- Capabilities must be simple in order to be understood by control and system engineers.

Having these requirements the proposal is to complete the atomic capabilities with attributes that contains information about the temporal and frequency response of the controlled system, about the controller, about the open-loop system and about the actuators and the sensors. So, the capabilities are formed by the following attributes:

1. Related to the controller:
   - Identification: Controller name, as PID, predictive, etc.
   - Controller type: whether is linear or not.
   - Controller structure: Feedforward, multi-variable, control ratio, etc.
2. Related to the open-loop system:
   - Order and type: number of poles of the open-loop system and number of poles at the origin.
   - Delay: approximate time that goes by since a different input signal is applied until a change on the output of the system is observed.
   - Gain: deviation of the output value in steady state respect to the input signal.
   - Time constant: Time that takes a first-order system to get the 63% of the output value. It indicates how fast the system temporal response is.
3. Related to actuators and sensors:
   - Sort: kind of actuator or sensor (mechanic, electric, chemical, etc.)
   - Precision: interval in which the given magnitude can be erroneous.
   - Sensibility: minimal variation of the input that can be detected by the sensor or to which responds the actuator.
   - Time constant: time that indicates how fast answers the actuator or sensor to changes on the input signal.

- Hysteresis: deviation of the magnitude value depending on whether it is reached by an increasing or a decreasing continuous change of the input.
- Temperature dependence: change on the output value due to a different temperature from nominal.
- Linearity interval: interval in which the actuator or the sensor works on its linear zone.
- Delay: delay between a change on the input and its corresponding effect on the output.
- Noise rejection: maximal power of the noise signal that does not affect the sensor or actuator output signal.

These attributes are included in atomic capabilities to be used in future applications but not in the scope of this work. The following attributes have been modified, redefined or adapted to reach the requirements above mentioned and will be immediately used in the decision algorithm:

4. Related to the temporal and frequency response of the closed-loop system:
   - Precision
   - Overshoot
   - Rapidity
   - Persistence
   - Robustness
   - Aggressiveness
   - Control effort
   - Coherence
   - Identification

4.1 Precision

This attribute is related to the deviation that the controlled system has respect to a ramp input signal with a ? slope at 2 ? times, being ? the time constant of the open-loop system.

$$Precision = 100 - \lim_{t \to 2\tau} \frac{\tau\ t - y(t)}{\tau\ t}\ 100$$

This parameter has been defined in this way because it is possible to avoid infinite or zero values (the error is calculated for a time equal to twice the open-loop time constant). And also, it is independent of the kind of input applied (therefore can be compared with its equals).

4.2 Overshoot

As in Control Theory this attribute represents the relative value of the maximal value of the output signal respect to the steady state value. It is calculated as follows:

$$M_p = \frac{y(t_p) - y(\infty)}{y(\infty)}\ 100\%$$

Where t_p is the time at which the maximum value of the temporal response is produced. If the temporal response do not present an overshoot then this parameter is 0%.

4.3 Rapidity

This attribute is a ratio between the time needed by the controlled system to get the steady state when there is a change on the set point and the same time but in open-loop. It is defined as:

$$Rapidity = \frac{t_{slc}}{t_{sla}}$$

Where:

- t_{slc}: closed-loop system settling time.
- t_{sla}: open-loop system settling time.
- Settling time: it's the time that requires the system to maintain the output between an interval of 2% or 5% of the steady state value.

The lower this value is the faster the systems responses.

4.4 Persistence

This attribute is related to the capability of the system to reject disturbances, which is to maintain the output signal within an acceptable value.

It has to be said that the disturbance rejection is sometimes a specification for designing the controller, so its evaluation will depend on the Control Engineer judgement. Anyway, a formula to calculate is provided for the two most common disturbances, which are step and pulse types.

In the case of step perturbations of amplitude A, the following way to calculate the persistence is proposed:

$$Disturbance = \left[(1 - \frac{IAE}{A\tau})percentage\_disturbances\right]\%$$

Where

- $IAE = \int_{t_1}^{t_2} \|e(t)\|dt$ is the integral of the absolute error value.
- A is the amplitude of the step.
- τ is the open-loop time constant.

The choice of τ is because it does not change as the closed-loop time constant does (depends on the controller and hence this attribute won't be independent). To calculate persistence for a pulse disturbance:

$$Disturbance = \left[(1 - \frac{IAE}{B})percentage\_disturbances\right]\%$$

Where

- IAE it's the integral of the absolute error value.
- B pulse area: pulse amplitude x pulse duration.

In both cases if the equation between parenthesis is negative, the persistence takes 0% value. That is that the system does not reject disturbances.

In case that there exists more than one kind of disturbance, this index will be the maximum value of all of them.

4.5 Robustness

This attribute represents the capability of the controlled system to maintain the output within acceptable values when there are variations in the parameters of the open-loop system or non-modelling dynamics.

The phase and gain margins give a magnitude of the system stability. They provide the maximum change that can have the parameters of the open-loop system to maintain stable the closed-loop system.

To calculate robustness it is necessary to know the phase and gain margins (PM_{nom} and GM_{nom}) of the system without variations on the open-loop parameters and both margins (PM and GM) with the maximum variations of the parameters. So the formula is:

$$Robustness = \frac{\frac{MP}{MP_{nom}} + \frac{MG}{MG_{nom}}}{2}$$

4.6 Aggressivenes

This attribute represents the system speed to respond to changes in the set point. It is defined as the percentage relation between the rising time (tr) and the settling time (ts) of the closed-loop.

$$Agressiveness = 100\% - \frac{t_r}{t_s}100\%$$

4.7 Control effort

This attribute describes the effort that the controller needs to keep the output in the desired value. Its evaluation is made as:

$$Control\_effort = \frac{IADU}{u_{max} - u_{min}}$$

Where:

- $IADU = \int_{t_1}^{t_2} \| \frac{du(t)}{dt} \| dt$ is the integral of the absolute value of the derivative of the control signal.
- u_{max} is the maximum value that can take the control signal.
- u_{min} is the minimum value that can take the control signal.

4.8 Coherence

This attribute is related to the work interval in which the designed controller satisfies the required specifications.

$$Coherence = work\_interval$$

4.9 Identification

This attribute is added to identify the controllers that deal with the same inputs-outputs units in order to compare only the capabilities of the same sort of controller. That is, if an agent has several position and speed controllers, and the currently task needs a speed controller, the comparison among capabilities should be done only for that ones that represents speed controllers. It is defined as:

$$Id = input\_units, output\_units$$

5 Example of Capability

Assuming the system open-loop transfer function as:

$$FT = \frac{1}{s^2+3s+2}$$

controlled by a PID with the approximate derivative with the following constants:

Kp = 150	Ki = 40
Kd = 50	N = 50

perturbed 95% of times by a pulse of amplitude 10 and duration of 20 sec., and with a non-modelled pole in

$$\frac{500}{s+500}$$

let's complete the atomic capability associated to this controller. The simulated response of the open-loop system to a step set point of amplitude 3 is:

The open-loop time constant of the system is:

$$\tau = \frac{1}{\zeta\,\omega_n} = \frac{1}{1.5} = 0.6667$$

and the open-loop settling time is:

$$t_{sla} = 4.6s$$

The controlled system response is depicted in Figure 3 , with the pulse disturbance affecting it.

From this simulated response, it is possible to calculate The overshoot as:

$$M_p = \frac{3.36-3}{3}100\% = 12\%$$

The closed-loop settling time:

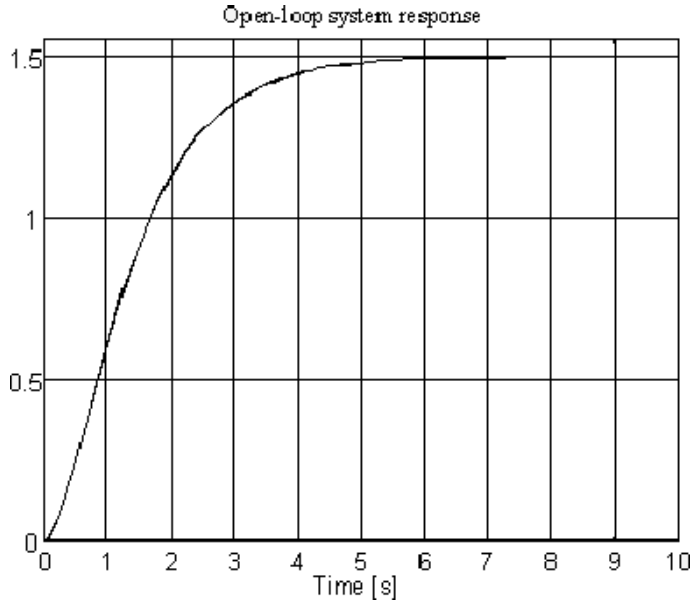


Figure2. Open-loop system temporal response

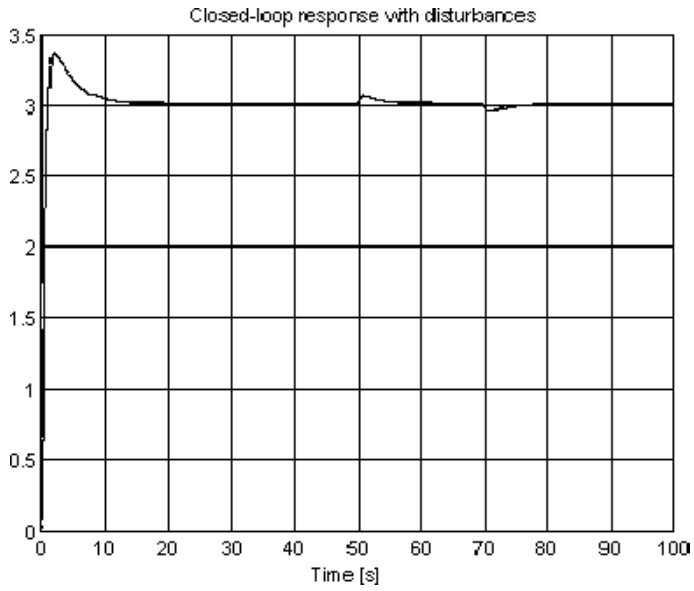


Figure3. Controlled system response with disturbances

$$t_{slc} = 8.6s$$

The rise time:

$$t_r = 0.8945s$$

And the IAE produced by the pulse disturbance:

$$IAE\_pulso = \int_{50}^{80} \|e(t)\| dt = 0.4983$$

The closed-loop response to a ramp input of τ slope is shown in Figure 4.

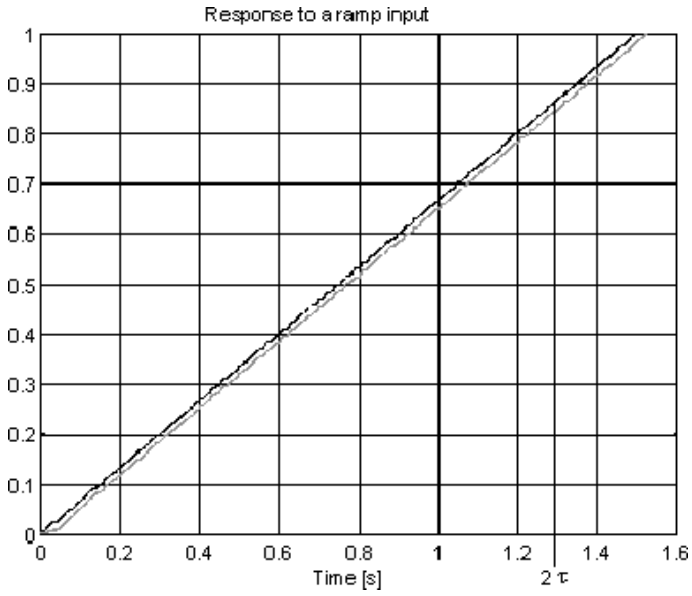


Figure4. Closed-loop system response to a ramp input

Considering this response, the system error in 2 τ is:

$$e = \lim_{t \to 2\tau} \frac{\tau\, t - y(t)}{\tau\, t}\, 100 = \frac{0.8865 - 0.8687}{0.8865}\, 100 = 2.0079$$

Apart from these figures, and applying some formulae cited in this paper is possible to reckon:

The area of the pulse disturbance:

$$B = amplitude\ duration = 10\ 20 = 200s$$

The IADU:

$$IADU\_pulso = \int_{t_1}^{t_2} \|\frac{du(t)}{dt}\| dt = 18.88$$

The nominal phase and gain margins:

$$MF_{nom} = 51.0080^o$$

$$MG_{nom} = 1.8043x10^3 dB$$

And the same margins but considering the non-modelled pole:

$$MF = 46.3911^o$$
$$MG = 10.5363 dB$$

With these values and making use of the different formulae of the attributes let's complete the atomic capability of this controller:

Precision	97.9921%
Overshoot	12%
Rapidity	1.8696%
Persistence	94.7633%
Robustness	0.4577
Agressiveness	89.5988%
Control_effort	0.6293
Coherence	0.6

The atomic capability of this controller is completed with these values and with the corresponding ones of the open-loop system, sensor and actuator and controller.

6 Conclusion

In this paper, one way to include knowledge about the physical body into the states of an agent is presented. Thus the agent has enough information about its dynamics to decide feasible actions. The idea is to have a set of controllers installed on the agent's body (and a set of atomic capabilities associated with them) that modifies its dynamics in a desired way. So, before committing itself to an action, the agent inspects its own capabilities and according to its physical constraints makes a decision.

Any commitment includes more than only its physical body constraints; it has to consider the task the agent is doing, the current state of the environment and the modifications produced by agents on it. That's why, besides the atomic capabilities, the agent has basic capabilities and symbolic capabilities. And all of them must be included in a decision algorithm.

Next step is to include knowledge encompassed in the atomic capabilities into a decision algorithm and apply it to a real physical system to verify its applicability.

Currently this idea is being applied to the soccer benchmark proposed in the RoboCup initiative [1] and in convoying of vehicles. Results are available on: `http://eia.udg.es/~bianca/phyiscal_agents`

Acknowledgements

This work is partially funded by projects TAP98-0955-C03-02 "Diseño de agentes físicos (DAFNE) / Physical Agents Design" and TAP99-1354-E, "Eurobot-Iberia: red temática en agentes físicos / Thematic Network on physical agents" of the Spanish Research Foundation CICYT, and "Laboratori de l'Equip Internacional de Robots Mòbils de Girona (RoGi), as special action 2000ACES00018 of the Catalan Government.

References

[1] Asada, M. et al, "The RoboCup Physical Agent Challenge", First RoboCup Workshop in the XV IJCAI-97, 51-56 (1997).
[2] Brooks, R., "Intelligence without reason", IJCAI'91, 569-595, (1991).
[3] Brooks, R., "A robust layered control system for a mobile robot", IEEE Journal of Robotics and Automation RA-2, 14-23 (April 1986).
[4] Dorf, R., Bishop, R. "Modern Control Systems 7th Edition", Addison-Wesley Publishing Company, (1995).
[5] Kuo, B. "Sistemas de Control Automático 7ª Edición", Prentice Hall Editors, (1996).
[6] Ogata, K. "Ingeniería de Control Moderna 3ª Edición", Prentice Hall Editors, (1998).
[7] Oller, A. et al., "DPA: Architecture for co-operative dynamical physical agents", MAMAAW'99, (June 1999).
[8] Phillips Ch., Harbor, R. "Feedback Control Systems 2nd Edition", Prentice Hall International, (1991).
[9] Shinskey, F. "Process Control Systems 3rd Edition. Application, Design and Tuning", McGraw-Hill Publishing Company, (1988).
[10] Shoham, Y., "Agent-oriented programming", Artificial Intelligence 60, 51-92 (1993).
[11] Zhang, Y, Mackworth, R., "Will the robots do the right thing?", Technical Report TR 92-10, UBC, (1992).
[12] Zhang, Y, Mackworth, R., "Constraint Nets: A semantic model for hybrid dynamic systems", Theoretical Computer Science 130, 211-139, (1995).

Sesión 2: Robótica móvil

La integración de sistemas basados en el conocimiento y sistemas de tiempo real por medio de un robot móvil.

M. Henao[1], J. Soler[2], and V.Botti[3]

[1] Departamento de Sistemas Informáticos y Computación
Universidad EAFIT, Medellín - Colombia E-mail: mhenao@sigma.eafit.edu.co
[2] Universidad Politécnica de Valencia, España
e-mail: {jsoler, vbotti}@dsic.upv.es

Resumen El desarrollo de sistemas inteligentes de tiempo real es importante tanto para el campo de la inteligencia artificial como para el industrial, debido a su potencial en la solución de problemas complejos del mundo real. Para hacer esto, es importante tener tanto métodos que permitan modelar los requerimientos del problema y plantear buenas alternativas de solución, como herramientas que permitan llevar a cabo la construcción del sistema resultante. Este artículo presenta la metodología CommonKADS-RT para desarrollar sistemas inteligentes en tiempo real para problemas que requieren del manejo de conocimiento con variables temporizadas y su aplicación para un problema de un agente de control de un robot móvil.

1 Introducción

En la actualidad se están construyendo muchos tipos de sistemas, que por lo general se identifican con alguna de las áreas de investigación de la ciencia de la computación. Es así, como algunos han estado interesados en crear sistemas que manejan el conocimiento de un dominio y otros en sistema que tienen el tiempo como una de las variables trascendentales para realizar sus tareas, surgiendo las áreas de los sistemas basados en el conocimiento y de los sistemas de tiempo real, respectivamente. Como cada vez surgen problemas más complejos que requieren de soluciones flexibles, adaptativas, con un comportamiento inteligente y con unos razonamientos y respuestas acotadas por el tiempo, han emergido las técnicas de los Sistemas Inteligentes de Tiempo Real - SITR [MHA95], [VHB97].

Para desarrollar cada uno de estos tipos de sistemas se requiere tener una metodología que defina las técnicas, herramientas y procesos más importantes para llevar a cabo el desarrollo del proyecto en una forma exitosa. Estas metodologías no deben ser ad hoc sino generalizadas para poder garantizar ciertos estándares necesarios para asegurar tanto el éxito del proyecto como la calidad del sistema de software desarrollado.

En este artículo se presenta la adaptación de varios métodos y metodologías conocidas, para que permitan modelar tareas temporales o con restricciones de tiempo, con el fin de tener una metodología que pueda ser utilizada en la construcción de sistemas inteligentes de tiempo real en el entorno industrial. El artículo está estructurado de la siguiente forma: la sección 2 presenta en forma general la metodología

CommonKADS. En la sección 3 se muestran algunos de los modelos utilizados en CommonKADS-RT y en la 4 se aplica la metodología al control de un robot móvil. Por último, en la sección 5 se plantean algunas conclusiones.

2 Metodología CommonKADS

CommonKADS [SAA+98] es una metodología que reúne una serie de métodos, técnicas y herramientas para el desarrollo de Sistemas Basados en el Conocimiento - SBC. En ella se plantea el hecho de que hacer un sistema basado en el conocimiento es esencialmente una actividad de modelado. Desde este punto de vista, el sistema es un modelo operacional que exhibe los comportamientos deseados que se han especificado u observado en el mundo real. La metodología refleja algunos métodos del análisis y el diseño estructurado, del paradigma de objetos y de algunas teorías gerenciales como la planeación estratégica, la reingeniería, entre otras.

Como en un sistema de conocimientos no debe modelarse únicamente los conocimientos expertos, sino también otros sistemas del mundo real tales como la organización, el usuario y la interacción entre él y el sistema, para conseguir un sistema viable comercialmente, entonces se plantean los siguientes modelos:

1. Modelo de la Organización, describe la organización en la cual el sistema será implantado.
2. Modelo de Tareas, describe la tarea asignada al sistema y las relacionadas con la organización.
3. Modelo de Agentes, una descripción de alto nivel de los involucrados en la tarea, por ejemplo los usuarios del SBC y los sistemas de cómputo relacionados con él.
4. Modelo de Comunicación, describe la interacción entre los agentes que participan e interactúan en la realización de la tarea.
5. Modelo de Conocimientos, muestra el conocimiento usado por el sistema para solucionar la tarea. En este modelo se puede utilizar una librería de componentes de modelación genéricos, como por ejemplo métodos de solución de problemas de una tarea específica y ontologías del dominio. [BrV94].
6. Modelo del Diseño: Es el enlace entre los modelos conceptuales y la implementación del ordenador. Describe la arquitectura y la funcionalidad detallada del sistema a ser implementado.

Adicionalmente, para estos modelos se incluye una serie de formularios que permiten registrar las especificaciones y los requerimientos necesarios en cada uno de ellos, dándose que todos son una vista diferente del mismo problema. Además, el ciclo de vida que se sigue en CommonKADS es por espiral con las etapas comunes a cualquier sistema de información por ordenador [WSB92].

Una de las cosas importantes para resaltar de esta metodología es el hecho que es una de los más utilizadas para el desarrollo de SBC, tomándose incluso como el estándar europeo.

3 Aproximación Metodológica de CommonKADS para SITR CommonKADS-RT

SiUn software de tiempo real es típicamente diseñado como una serie de procesos concurrentes comunicados. La descomposición del sistema en relación con el tiempo, el espacio, el flujo de control, el tamaño, el flujo de datos, las subfunciones, los objetos y las estructuras de datos ha sido ampliamente investigado por los ingenieros de software en las últimas décadas, surgiendo así métodos y metodologías específicas para el diseño de Sistemas de Tiempo Real - STR como las presentadas en [BuW94], [Cal97], [Del97], entre otras.

Ya que CommonKADS no fue planteada para modelar estas características de tiempo real y dado que se requiere utilizar para la creación de un Sistema basado en el conocimiento de tiempo real - SBCTR, entonces es necesario hacerle ciertos cambios y adiciones que permitan su utilización efectiva, originándose CommonKADS-RT (CommonKADS-Real-Time).

Esto es así porque la propuesta que hemos trabajado está fundamentada en CommonKADS por ser consistente y robusta, porque integra los conceptos más importantes de la ingeniería del software siguiendo con el paradigma de objetos y porque además ha sido ampliamente utilizada y validada para crear sistemas inteligentes. Para modelar las características de tiempo real se escogió como base RT-UML [Dou98], [Dou99] ya que en éste se ha definido una serie de técnicas que se pueden aplicar para realizar el análisis de un sistema de tiempo real e incluso de un sistema basado en el conocimiento de tiempo real. Seguidamente, presentamos una breve descripción de la metodología a través de la Figura 1 y posteriormente se detalla por medio de un ejemplo práctico.

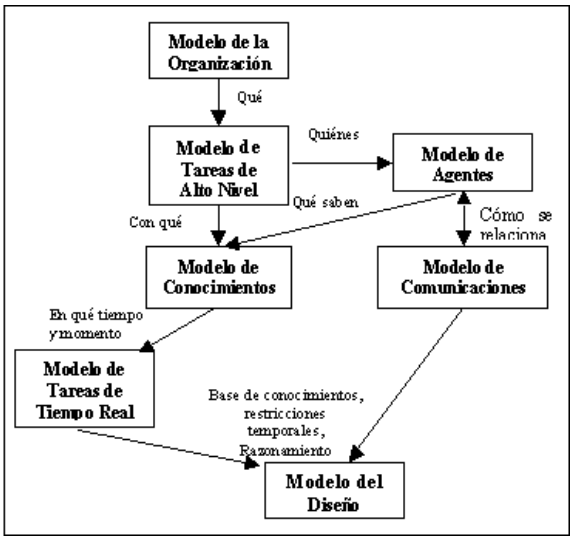


Figura1. Modelos de CommonKADS-RT

Al Modelo de la Organización original, se le han introducido una serie de cambios en la estructura de sus formularios y se han añadido los casos de uso para representar las funciones del sistema desde el punto de vista del usuario. Después de esto, se comienza con el desarrollo del Modelo de Tareas de Alto Nivel (TAN), el Modelo de Conocimiento y el Modelo de Tareas de Tiempo Real. En este caso, primero se construye un diagrama de contexto y un diagrama de entidades (conceptos). Luego, se hace la lista de eventos externos y algunos diagramas de secuencia y de colaboración. Se determinan las características de las tareas de tiempo real y se modelan a través de UML para tiempo real. Se terminan de especificar los modelos de TAN y de Conocimiento a través del Lenguaje de Modelado del Conocimiento de CommonKADS (CML) y de los formularios apropiados. Se utilizan los diagramas de transición de estados para analizar algunos de los conceptos que resulten de este punto. Se especifica el Modelo de Agentes y el de Comunicación, utilizando los diagramas de secuencia y de colaboración. Se completan las tareas de tiempo real. Y por último, se pasa al Modelo del Diseño con los diagramas de componentes de UML para definir la arquitectura del sistema. Algunos de los procesos del análisis se pueden hacer en forma paralela.

3.1 Modelo de la Organización

Como ya se dijo anteriormente, CommonKADS plantea un Modelo de la Organización y una serie de formularios que sirven de guía para analizar la organización en donde está el problema y se implantará la solución.

Cuando se va a hacer el análisis de un SBCTR se debe considerar si el problema que se está estudiando es exhaustivo en conocimientos y si se comporta como un sistema que tiene que responder correctamente a unos eventos de entrada (periódicos o aperiódicos) en ciertos plazos de tiempo aceptables y previamente determinados, es decir bajo unas restricciones temporales. Si es así, entonces desde el comienzo, en el análisis de requerimientos se debe definir que el sistema tendrá unas tareas que tienen comportamiento de ese estilo y que por lo tanto deben tener unas características propias del concepto de tiempo real.

Siguiendo los formularios de este modelo, se inicia con la identificación de las oportunidades para hacer un SBCTR y de la información requerida para saber qué es lo más importante, necesario o urgente para la organización. Con esto se podrá establecer una forma de asignación de prioridades a los problemas y sus soluciones, de acuerdo con ciertos criterios definidos por la misma empresa. Adicionalmente, para las situaciones basadas en conocimiento y con el recurso tiempo como factor determinante se estima el tiempo en que el proceso se realiza en la actualidad (al menos para aquellos procesos que son críticos) y aquel que se consumiría en el caso de adoptar una solución de SBCTR. Esto último podría llegar a ser un factor competitivo para la organización.

Es importante resaltar que en este modelo se hace la diferencia explícita entre lo que es el análisis del problema y lo que es el análisis de la solución. Se incluye en

estos formularios la especificación de ciertas variables temporales de alto nivel, como por ejemplo la prioridad que tiene asociada el proceso, su tiempo de activación y de finalización, el tiempo máximo de ejecución que podría en un momento tener, la especificación del equipo apropiado de adquisición de datos (sensores) y de actuación sobre el entorno (actuadores), la comunicación que se puede tener con otros sistemas, la determinación de si el proceso es periódico o no, y el impacto que tendrá el sistema informático dentro del área de la organización. Todos estos cambios tienen que ver con la consideración y el manejo apropiado de las características de los procesos asociados con el tiempo como recurso crítico.

3.2 Modelo de Tareas de Alto Nivel

Este modelo está basado en el problema seleccionado, en el dominio específico y en los criterios o variables que se identificaron en el análisis del problema y de la solución y que quedaron reflejados en el Modelo de la Organización. Para CommonKADS una tarea es una función compleja que se realiza como parte de un proceso del negocio y que tiene asociado un método que describe cómo la tarea es efectuada a través de una descomposición de funciones. Ésta a su vez puede estar formada por otras tareas, por inferencias o por funciones de transferencia. En cambio en tiempo real, una tarea se asocia a un proceso de ejecución que puede tener incorporadas unas restricciones temporales. Por tanto, lo primero que se debe hacer es establecer una diferencia entre esos tratamientos de la tarea, surgiendo el concepto de Tarea de Alto Nivel - TAN para las funciones complejas y Tarea de Tiempo Real - TTR para un proceso de más bajo nivel. Para las TAN se debe analizar todo lo que se plantea en CommonKADS, es decir la descripción del proceso de la tarea, sus componentes, los agentes que están involucrados en ella, entre otras cosas. Y además, como el entorno en que se encuentra es de tiempo real, entonces se debe determinar su comportamiento periódico y su plazo máximo de ejecución dentro del cual se deben realizar las acciones. Por esto, se utilizan los escenarios del dominio de [Deu88] que reflejan las situaciones posibles en que puede estar el sistema en un momento dado, y más específicamente la TAN que se está analizando. Para las TTR se plantea el Modelo de Tareas de Tiempo Real.

3.3 Modelo de Conocimientos

Este Modelo está dividido en tres componentes importantes: El conocimiento del dominio, que son las estructuras estáticas del dominio; el conocimiento de inferencia que describe cómo las estructuras estáticas pueden ser usadas para hacer el razonamiento; y el conocimiento de la TAN para definir cuáles son las metas que se intenta alcanzar para aplicar el conocimiento. Relacionando esto con el paradigma de objetos, se tiene que los Conceptos son análogos a las clases pero sin métodos (es el conocimiento de la decisión), los Esquemas de Reglas son las relaciones de dependencia causal, el Modelo del Dominio son las instancias de los tipos de conocimiento

y los Roles de Conocimiento son los nombres abstractos de los objetos de datos que indican su rol en el proceso de razonamiento.

Para definir esta jerarquía se cuenta con el lenguaje CML2 (Conceptual Modelling Language) que permite modelar dicho conocimiento a través de una notación semi-formal. Entonces, para continuar involucrando las variables de tiempo real se deben hacer algunos cambios en este lenguaje para que se pueda modelar el conocimiento temporal. Como se dijo en 3.3 la estructura de la tarea de tiempo real de igual forma debe tener su representación en este modelo.

Por ejemplo, para expresar las funciones que se encargan de obtener los datos o presentar la información a través de un sensor o de un actuador respectivamente, es posible tener una función de transferencia de tiempo real, lo cual se ve en la Figura3.3.

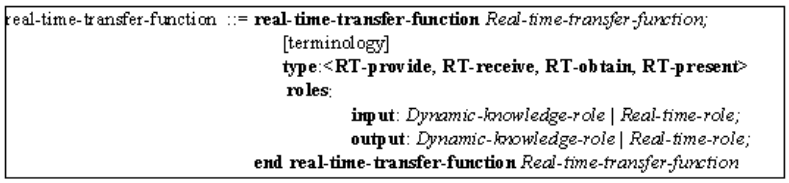


Figura2. Función de Tiempo Real para transfrencia de datos o información

Así mismo, como en los conceptos se especifican las propiedades temporales de la clase (el período de ejecución, el tiempo de cómputo, el tiempo de finalización, el plazo de respuesta - deadline, etc.) [Bot96], entonces a los tipos de datos primitivos definidos se le adiciona los tipos denominados absolute-time y relative-time que representan el tiempo absoluto (determinado por los pulsos de un reloj) o el relativo (orden parcial estricto impuesto en los conjuntos de todas las ocurrencias de las transacciones) [FHL+96]. Ver figuras 3 y 4.

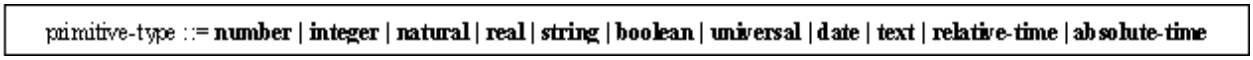


Figura3. Tipos de datos

Figura4. Tipos de tareas de Tiempo Real

Asimismo, cuando se está construyendo este modelo se utilizan los diagramas de secuencias que son una representación temporal de los objetos y sus interacciones,

los diagramas de transición de estados para caracterizar las vidas de los objetos, y una variación de la lista de eventos externos de [Dou98] que relaciona dichos eventos con las respuestas del sistema y sus tiempos. Algunos de estos se muestran en la sección siguiente utilizados para un robot móvil.

3.4 Modelo de Tareas de Tiempo Real

Para la parte específica de las estructuras de tiempo real se ha tomado como base ARTIS, una arquitectura de agentes para sistemas de tiempo real crítico [BCJ+99], y más específicamente en lo que se refiere al modelo de tareas de tiempo real [GTB+97] según el cual una tarea está formada por tres partes, obligatoria, opcional, y final, de las cuales tan sólo las partes obligatoria y final pueden tener restricciones temporales críticas. Esto implica tener una representación en el Modelo de Conocimientos que refleje dicha estructura.

3.5 Los otros modelos

En el Modelo de Agentes, un agente representa el ejecutor de una TAN, puede ser un humano, un sistema de información computarizado o cualquier entidad que sea capaz de realizarla. Así, en este modelo se define cómo son los agentes, cómo se caracterizan, qué conocimiento tienen, con quién se comunican y cuáles son las condiciones temporales de sus funciones. Además, el modelo proporciona el enlace entre los Modelos de Tarea de Alto Nivel, Comunicación y Conocimiento.

En un SITR es muy importante determinar cómo es el patrón de llegada de los mensajes, periódicos o aperiódicos y cómo es el patrón de sincronización de los mismos, y por eso en el Modelo de Comunicaciones se modelan las transacciones de comunicación entre los diferentes agentes involucrados en el sistema, con estas restricciones.

Por último, se pasa al Modelo del Diseño que es la construcción misma del sistema en la herramienta apropiada. En este caso, ya todos los requerimientos temporales se han incluido, así que no se cambia nada de esto.

4 Aplicación de la Aproximación de CommonKADS con Tiempo Real para un robot móvil.

En este ejemplo se plantea el desarrollo del software de control para un robot móvil que se encarga de repartir paquetes dentro de una planta de oficinas. El robot recibe, a través de una radio ethernet, peticiones para pasar a recoger un paquete en un despacho concreto y entregarlo en otro despacho dentro de la misma planta. Estas peticiones pueden ser hechas en cualquier momento, para lo cual el robot deberá hacer una planificación del trabajo a llevar a cabo en función de su localización, de las peticiones pendientes y de la prioridad de éstas, tratando de minimizar el tiempo en los envíos.

Este sistema se ha planteado para el robot móvil "Movil Pionner 2" conectado mediante un puerto serie RS-232 a un PC de abordo. Este robot dispone de un anillo de 16 sensores de ultrasonidos que le permiten reconocer todo el horizonte, de bumpers para detectar contactos o colisiones y proporciona la información de odometría necesaria para su localización.

La resolución de este problema se ajusta perfectamente a los sistemas inteligentes de TR pues nos encontramos ante la necesidad de controlar un proceso físico como es la conducción del robot con una serie de restricciones temporales propias de la dinámica de éste y debido a la complejidad de la aplicación, tenemos subproblemas como la planificación del trabajo o de la trayectoria, entre otras, en donde son adecuadas técnicas propias de la IA.

4.1 Tareas de alto nivel

En el comportamiento global del robot se han identificado una serie de tareas de alto nivel Figura 5Figura :

- evitar_obstáculos. Es necesario asegurar la seguridad del robot y de los elementos de la oficina, así pues esta TAN se encarga de realizar todas las acciones necesarias para evitar colisiones. Es la tarea más prioritaria del sistema pues debe asegurar la integridad del robot y de las instalaciones. Fundamentalmente, se encarga de adaptar la trayectoria planificada de antemano para evitar contratiempos.
- chequeo_malfuncionamiento. Esta TAN se encarga de verificar el buen funcionamiento del robot, detectando cualquier anomalía. En particular revisa el estado de las comunicaciones del robot y en el caso que éstas fallaran, se realizarán las acciones necesarias para reanudarlas o para llevar el robot a un estado seguro. También se encarga del mantenimiento de las baterías, activando los mecanismos de alerta para devolver el robot al lugar de carga con la suficiente antelación.
- localización. El robot debe de tener en todo momento una estimación de la posición en la que se encuentra dentro de su escenario de trabajo. Es pues necesario disponer de un mecanismo que le permita obtener esa estimación y además su verificación a lo largo del tiempo. El robot dispone de un mecanismo de localización local y para poder verificar esto, utiliza la información de los sensores y la estimación de los valores previstos a partir de un mapa reticular de cada estancia.
- navegación. Esta TAN determina la trayectoria que debe de seguir el robot para alcanzar los objetivos planteados. La TAN se encarga de calcular esta trayectoria y de determinar las acciones sobre los motores del robot para alcanzar los objetivos planteados.
- planeación. Esta TAN se encarga de activar el funcionamiento del robot, analizar las peticiones que llegan por radio y planificar los objetivos que se tienen que desarrollar. Utiliza el mapa topológico del entorno para determinar los nuevos objetivos que después comunicara al resto de tareas.

– conducción. Las acciones básicas que puede realizar el robot Pionner son: - desplazarse a cierta velocidad, - girar cierto ángulo, - detenerse. Así pues es necesario. Esta tarea por lo tanto, se encarga de controlar los movimientos del robot, es decir la conducción propia del robot hasta que se cumplen los objetivos locales asociados a éstas.

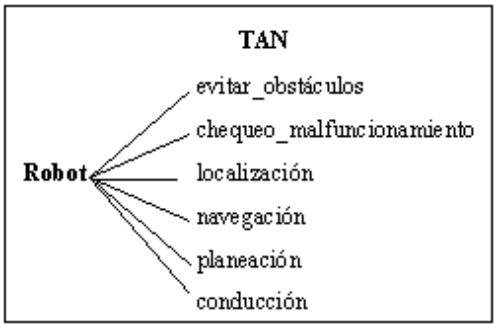


Figura5. Tareas de alto nivel del problema

4.2 Descripción de tareas

Debido a la limitación en este articulo, solamente se describirán dos TAN con detalle, una de tiempo real y otra basada en el conocimiento. Estas son conducción y evitar_obstáculos respectivamente.

La figura 4.2 representa el diagrama de estados del robot que muestra los eventos que se deben realizar para pasar de un estado a otro y las tareas que los generan.

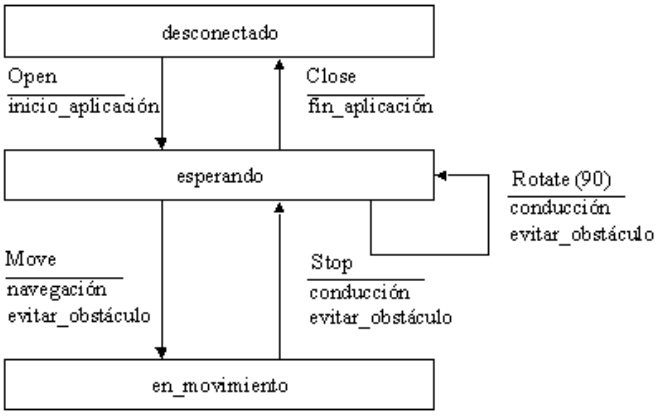


Figura6. Diagrama de transición de estados.

Siguiendo con la especificación mediante el lenguaje propuesto, las tareas anteriormente mencionadas quedan definidas de la siguiente forma: en la figura 7 se

presenta la tarea de conducción, la cual es una tarea de tiempo real, que se repite cada 100 milisegundos y tiene un tiempo límite de ejecución de 8 milisegundos.

```
real-time-task conducción;
    rt-task-type: periodic;
    from: conducción;
    relative-time: yes;
    deadline: 8; /* milisegundos */
    period: 100; /* milisegundos */
end real-time-task conducción;
```

Figura7. Definición de la tarea de Tiempo Real: Conducción

En la figura 8, se muestra la especificación general de la tarea evitar_obstáculo basada en el conocimiento.

```
task evitar_obstáculo;
    task-definition
    goal: "adaptar la trayectoria para evitar colisiones;
     roles:
        input: valor-de-los-sensores, velocidad;
        salida: conjunto-de-acciones;
     specification: "ejecuta las acciones necesarias para asegurar la integridad del robot";
end task evitar_obstáculo;
```

Figura8. Definición de la TAN: evitar_obstáculo

Después de hacer la especificación completa de cada una de las TAN y de las Tareas de Tiempo Real se pasa a la construcción del Modelo del Diseño en el entorno de ejecución apropiado.

5 Conclusiones

En este artículo se ha presentado un método para modelar sistemas inteligentes de tiempo real a través de la construcción de un sistema de control de un robot móvil. Esta aproximación ofrece facilidades para el desarrollo de sistemas complejos y muestra además, que las diferentes áreas de investigación dentro de la informática tienen que entrar a compartir sus conocimientos, para crecer mucho más y evitar la pérdida de tiempo al hacer trabajos ya realizados. Así, cuando se habla de sistemas inteligentes en tiempo real, lo que se busca es que se compartan los resultados del área de sistemas de tiempo real y del área de la inteligencia artificial para poder hacer

sistemas más rápidos, complejos, confiables y mucho más cercanos a la realidad. Por esto, el área de SITR ya tiene muchos adeptos que están trabajando en la validación y creación de algoritmos, lenguajes, teorías formales, metodologías, entre otros, para crear sistemas de este estilo.

Además, como el futuro es muy amplio se posibilita la creación de muchos dominios de investigación, en especial en el planteamiento de guías que permitan que el desarrollo del sistema se haga de una forma apropiada.

Referencias

[BCJ+99] V. Botti, C. Carrascosa, V. Julian, J. Soler. Modelling Agents in Hard Real-Time Environments. Proceedings of the MAAMAW´99. Lecture Notes in Computer Science, vol. 1647. Springer - Verlang, Valencia. 1999. pp 63-76

[Bot96] V. Botti. Notas de Clase, curso doctoral Sistemas de Tiempo Real. Universidad Nacional de Colombia, sede Medellín. 1996.

[BrV94] J. Breuker, W. Van de Velde, W. CommonKADS Library for Expertise Modelling; Reusable problem solving components. The Netherlands: IOS Press. 1994. 359 p.

[BuW94] A. Burns. A.J. Wellings. A Design Method for Hard Real-time Systems. Real-Time systems. Vol. 6, No. 1, pp. 73-114. 1994.

[Cal97] J.P. Calvez. Specification and Design of Embedded Real-time Systems; MCSE Methodology. http://www.irestre.fr/mcse/htmlan/. 1997

[Del97] J.A. de la Puente. Diseño de sistemas de tiempo real - Introducción a HRT-HOOD. ftp://ftp.dit.upm.es/str/software/mine.tar.gz. Madrid. 1997.

[Deu88] M. Deutsch. Focusing Real-Time Systems Analysis on User Operations. IEEE Software, Sept. 1988. 39-50

[Dou98] B. P. Douglass. Real-Time UML. Addison-Wesley, United States of America. 1998, 365 p.

[Dou99] B. P. Douglass. Doing Hard Time; Developing Real-Time Systems with UML, Objects, Frameworks, and Patterns. Addison-Wesley, United States of America. 1999, 749 p.

[FHL+96] E. Falkenberg, W. Hesse, P. Lindgreen, et al. FRISCO, A Framework of Information System Concepts. The IFIP WG 8.1 Task Group FRISCO. 1996. 221 p.

[GTB+97] A. García-Fornes, A. Terrasa. V. Botti. A. Crespo. Analyzing the Schedulability of Hard Real-Time Artificial Intelligence Systems. Engineering Applications of Artificial Intelligence. Pregamon Press Ltd. pp 369-377. 1997.

[MHA95] D. Musliner, J. Hendler, A. Agrawala, et al. The Challenges of Real-Time AI. Computer. January 1995. 58-66.

[SAA+98] Schreiber, A. Akkermans, Anjewierden, et al. Engineering of Knowledge: The CommonKADS Methodology [version 0.5]. The Netherlands: Department of Social Science Informatics, University of Amsterdam. 1998.

[VHB97] E. Vivancos, L. Hernández, V. Botti. Construcción y análisis temporal de sistemas basados en reglas para entornos de tiempo real. Actas de la VII Conferencia de la Asociación Española para la Inteligencia Artificial, CAEPIA,97. Málaga, España. 1998. 675-684.

Eldi's Activities in a Museum

M. Castrillón Santana, J. Cabrera Gámez, D. Hernández Sosa,
A. C. Domínguez Brito, J. Lorenzo Navarro, J. Isern González, C. Guerra Artal,
I. Pérez Pérez, A. Falcón Martel, M. Hernández Tejera and J. Méndez Rodríguez.

Centro de Tecnología de los Sistemas y de la Inteligencia Artificial (CeTSIA)
Campus Univ. de Tafira, 35017 Las Palmas, SPAIN.
e-mail: modesto@dis.ulpgc.es.
Universidad de Las Palmas de Gran Canaria - SPAIN

Abstract In this paper we will present Eldi, a mobile robot that has been in daily operation at the Elder Museum of Science and Technology at Las Palmas de Gran Canaria since December 1999. The system that controls Eldi and the rest of the installation has been conceived as a set of agents that interact by means of discrete events. This is an ongoing project that was organized in three different stages of which only the first one has been accomplished termed *The Player*. This paper will describe briefly the physical robot, the environment and demos developed. Finally, we will summarize some important lessons learnt.

1 Introduction

Last years have revealed Education and Entertainment as promising, though demanding, new application scenarios for robotics with a great scientific, economic and social potential [1]. The interest raised by products like Sony's Aibo or the attention deserved by the media to projects as Sage [2], Rhino [3], Kismet [4] and, more recently, Minerva [5] demonstrate the fascination of the general public for these new fashion robotics *pets*.

In this paper we will present Eldi, a mobile robot that has been in daily operation at the Elder Museum of Science and Technology at Las Palmas de Gran Canaria since December 1999. This is an ongoing project that was organized in three different stages of which only the first one has been accomplished. The initial phase, termed *The Player*, was devoted to design and build the physical robot, obtain a scalable and extensible software control architecture and put all this into operation in a number of shows and demos that should be offered to visitors. The second stage, actually under development, has been called *The Cicerone* and aimed at adding better navigational capabilities in the robot such that it can give tours through some of the Museum's halls. In a final phase, termed *The Vagabond*, Eldi will be allowed to move erratically across the Museum during its *spare* time (i.e. while not required to give a tour, attend a show or recharge batteries) and it will be possible for a visitor to demand its attention and services through a multimodal interface (gesture, voice and a touchscreen). This paper will focus on the accomplished first stage to succinctly describe the physical robot, the environment and demos developed. Finally we will summarize some important lessons learnt.

Figure1. Front and side views of Eldi.

2 ELDI system anatomy

2.1 Hardware description

As stated above, the first phase of the project, carried out during 11 months, was devoted to build the robot and accomplish a first level of capabilities. The main goal is to attract visitors' interest towards Science and Technology. Physically, the robot's body has two main components (see figure 1). The lower part integrates a commercial Nomadic's XR4000 mobile platform that gives the robot its basic mobility and sensor capabilities. On top of this platform, it integrates a *torso* that hosts:

- a second processor,
- several radio communication systems that offer a 802.11 network interface with off-board systems
- transmission system of color video and sound from the robot,
- a touchscreen,
- loudspeakers
- and a two degree of freedom head.

The robot is equipped with an active vision system that comprises a pair of Sony EviG21 motorized color cameras housed in the head, a Directed Perception pan-tilt

that articulates the neck, and a PCI frame grabber. Basically, the processor installed in the mobile platform controls the motion, localization, obstacle avoidance, selects the video input for transmission and power resources of the whole system using a digital Port Control Board. It runs under the Linux O.S. The second *upper* system, that runs under Windows NT 4.0, controls the whole robot and develops all the interaction with user through a number of devices that include the vision system. Communications with off board systems are routed through the Linux system. The robot is also capable of recognizing voice commands from a number of people.

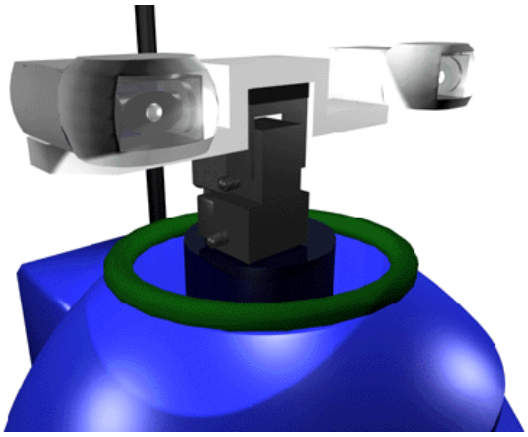


Figure2. A closer look to the head and neck of Eldi.

2.2 Control Systems

Three main subsystems control the robot (Eldi). In the upper body a 350 MHz Pentium II running NT takes care of vision, communication, interaction and high-level robot control. In the base, a micro-controller network manages the power and sensor systems (ultrasonars, infrared, bumpers) at low level, and a 233 MMX Pentium under Linux is in charge of platform sensor control, obstacle avoidance, localization, and low level motion control.

Several external machines complete the system (PCs connected using a local net with two segments being Eldi the net gateway): GameController dual Pentium 350Mhz under NT, BoardController Pentium 300 under W98 and ConacPC Pentium 100 under W95. These external machines controls also the active floor (in terms of illumination) used in different games, the videoboard, the sound, lights and so on. The whole system (figure 3) is not just the robot but a complete system designed for being able to offer different performance combining multimedia.

Global control is achieved by means of CAV [6], a software architecture that provides a substract for combining different machines in an asynchronous manner, concretely as a network of asynchronous weakly coupled agents modeled as Discrete Event Systems (DES), in particular as Port Automata [10] [11].

2.3 Sensors

In its upper body, the robot incorporates an active color vision system (SONY EVI-G21 and Imaging PCI frame grabber) mounted on a pantilt by Directed Perception (figure 2) for color detection and tracking (faces, robot games pieces), and a 14"SVGA color touchscreen by ELO Touchsystems for direct interaction with visitors (information, screen games). A laser beacon is also included as part of the location system (CONAC).

In its lower body (XR4000 Nomadic Technologies), there are microswitches for door opening detection and temperature probes for motor overheating checking. The robot base has two rings with 24 sensor modules, each one consisting of: one ultrasonic sensor for long range obstacle detection, one infrared sensor for short range obstacle detection and a bumper for contact detection (there are additional contact sensors on doors).

External sensors include a pair of color vision cameras mounted on the ceiling of the robot area to help players and robot location using a PCI Imaging frame grabber, a wireless microphone (TOA) for voice recognition, and laser detectors for accurate robot localisation.

2.4 Degrees of freedom

The robot head is mounted on a neck (PTU-Directed Perception) with 2 degrees of freedom (pan and tilt). The robot eyes are constituted by two motorized cameras that contribute with 2 mechanical degrees of freedom (pan, tilt) and 2 optical degrees of freedom (zoom, focus).

An holonomic system allows for the movement of the base with 4 wheels driven by 2 motors each (wheel rotation and translation).

2.5 Power Systems

A microcontroller based system is in charge of power distribution. The robot has a main battery set with four 33 Amp.h batteries and an auxiliary battery set with four 18 Amp.h batteries, both sets located in the base. Two DC-DC conversors and a devices power control board supplier upper body systems from battery sets.

2.6 Communication Systems

All the machines compose a two segments local network connected by means of a wireless network interface (Lucent Technologies Wireless IEEE 802.11 interface in 2.4 GHz using DS) that uses the lower body as gateway. Internal robot communication systems include a 100 MB/s Fast Ethernet linking the robot's main processors (upper and lower body) and an Arcnet network for information transmission between microcontrollers and the platform main processor. External systems are connected using a classic ethernet.

Audiovisual data are transmitted from the robot using a video-audio transmitter (Eagle 2.4 GHz PAL video and audio transmission).

3 Functional description: CAV in Eldi

A major breakthrough accomplished during this first phase has been the software architecture and associated methodology used to control the robot and off-board systems. The system that controls Eldi and the rest of the installation has been conceived as a set of agents that interact by means of discrete events. Eldi has been built using an extended version of CAV [6], a tool that enormously eases the definition and implementation of distributed systems modeled as a DES, specifically each agent is formalized as a Port Automaton [10] [11] and the whole system is in this way a network of Port Automata. CAV had been used previously in the design and implementation of active vision systems to facilitate the development and reduce the integration effort of such systems [7]. CAV allows modeling a robotic system as a set of distributed asynchronous weakly coupled active entities or agents [8] making tasks in parallel or concurrently, and interacting among them by means of events or signals, in this way the system is viewed from an agent perspective, as a network of agents sharing the same control and communication schemes. Thus therefore, in Eldi, all agents both on-board and off-board, share the same control and communication structure. This fact proved to be crucial during the integration of the whole system and sets the basis for system extension or modification along future phases of the project.

In figure 6 are depicted the main CAV agents involved in the current phase of project Eldi (figure 3), some other subsidiaries agents are not shown. In the diagram circles are agents, the cylinders are memory storages, arrows are data flows among agents, bi-directional arrows implies a protocol of command request and response. Functionally, as it is displayed on the diagram, the dashed contours, three different computer systems are distinguished:

The ROBOT: That physically includes:

- the Platform, the Nomadic XR4000 which also contains some extra power supply units for some devices (a speaker, TopRobot, the CONAC emitter and a video transmitter),
- and TopRobot, which is the upper part of the robot, the torso and the head.

The Platform is in charge of controlling low level platform movements, video transmission to external machines, the platform sensors (ultrasonars, infrared, bumpers, motor temperature sensors and odometry) and the extra power supply units.

TopRobot is responsible for head movements, vision behaviors, the voice, the multimedia application, the touch screen and high level monitoring and control of the Platform.

The GAME CONTROLLER: It is responsible for attending user commands through speech recognition (using IBM ViaVoice), executing agenda commands, controlling the evolution of board games and robot and player visual localization through the external ceiling cameras.

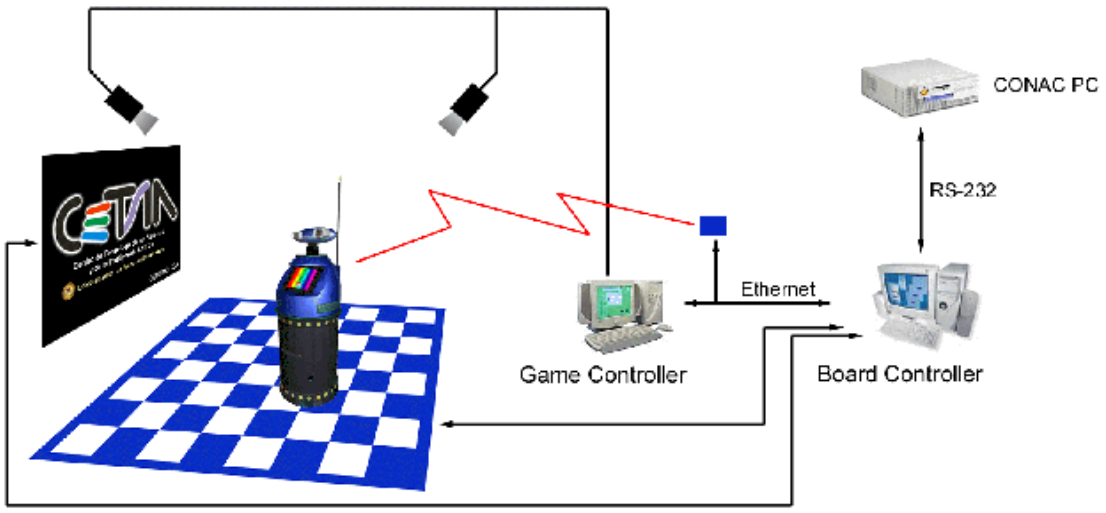


Figure3. Complete system schema. The robot moving on the chess-like board (Realboard), a videoboard (Virtualboard) that presents synthetic feedback of the game. The external machines and the cameras for controlling the game.

The BOARD CONTROLLER: In charge of controlling the game board (the Real Board) 3, the video board (the Virtual Board), and also hosts an agent collecting information about robot localization from the CONAC system.

4 Daily activity

Actually, the daily activity of the robot cycles between a show where robot commanding is performed by means of its voice interface. The show is carried out over a backlighted 8x8 glass board, figure reffig:wholesystem, where it develops several shows and plays different games as *Treasure Search* against a human player, figure 4, detected using vision. Eldi also can interact with a visitor to solve an 8-puzzle using vision, figure 5. Additionally it performs a choreography combining music, video and game board light effects. Furthermore, there is a resting period during which Eldi recharges batteries and offers the public the opportunity either to play different games as mastermind, chess or four-in-a-line, or to learn more about robotics using the multimedia information system.

Some of the available games have been programmed to offer the opponent explanations of the actions taken by the robot during the course of the game or even comments that reflect the judgement of the robot about player's actions. These comments are constructed using a set of phrase patterns that are randomly selected and offered via the voice synthesizer available on board, besides they are accompanied by movements of the head and eyes that track the face of the player.

Daily activity goes further from just the show, there are several background activities that should be always working, like weak obstacle detection and semi-automatic

system checking, as for example obtaining from time to time the histogram of the images captured by Eldi's cameras could provide the system a cue about when there is a malfunction with the cameras.

Typical Eldi's script in the current stage, i. e., *The Player*, is as follows (translated from spanish):

1. Introduction
   - Monitor: Hi Eldi and hi everybody, I am Eldi's assistant, I would like to welcome you to Eldi's Show. Eldi introduce yourself.
   - Eldi introduces the video that explains its design and abilities
2. Abilities
   - M: We will start showing you the kind of movements Eldi can do attending to my commands. *Eldi, are you ready?*
   - Eldi answers.
   - M: *Eldi be quiet.*
3. Moving
   - M: Eldi is able to move in any direction. *Eldi go backwards*, *Eldi go ahead*,*Eldi turn right*, *Eldi look ahead*, Eldi you are so quiet ... *Eldi say something.*
   - Eldi says something
4. Dancing
   - M: Eldi will now show you how he likes dancing ... *Eldi dance.*
   - Eldi goes to the dance starting point and 'dances'
5. Playing

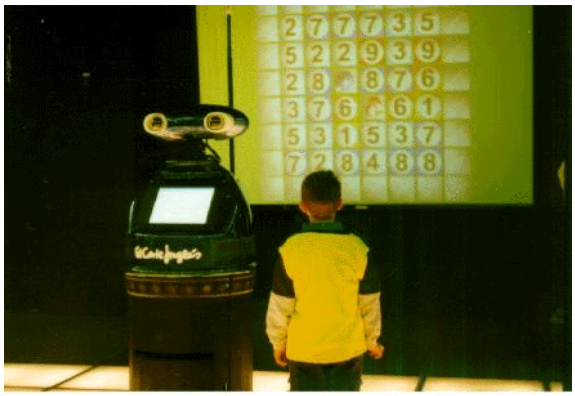


Figure4. Frame of the Treasure Search game.

   - Treasure Search (figure 4)
     - M: It is enough now let's play against Eldi. It is a simple game, you only need to know how to add. Now look at the floor, it is like a huge chesslike board where each cell has a value, and wins the first that reaches 40 without breaking the rules. The game status can be seen on the video board. *Eldi play.*

- Eldi goes to its starting point
- M: I need a volunteer ...
- The game goes on

– 8-Puzzle (figure 5)
  - M: Any of you can move the cells to test if Eldi can solve it, later Eldi will tell us the movements to solve the puzzle. *Eldi solve the puzzle*
  - Eldi says if he could solve the puzzle or if there is a problem
  - *Eldi next movement*, ...
  - Eldi says for example 'move x left'

Figure5. Eldi solving the puzzle.

6. Finish
   – M: Eldi is now hungry, so let's him go to rest. *Eldi go to bed*
   – Eldi goes to charge area

5 Conclusions

First we have to emphasize that Eldi is an ongoing shuttle which is still facing its early stages and that has been acting daily for eight hours at the Elder Museum of Science and Technology at Las Palmas de Gran Canaria since December 1999. With Eldi the objectives prefixed initially in this first stage, *The Player*, has been attained, the integration of an entertainment robot under a reliable and extensible control scheme in a real human environment.

Regarding to normal operation, after the installation and initial verification of the system by the developers, Eldi is operated on a daily basis by personnel not specialized in these systems. Therefore reliability needs to be addressed not only during the automatic startup checkouts but specially during operation. This demands a

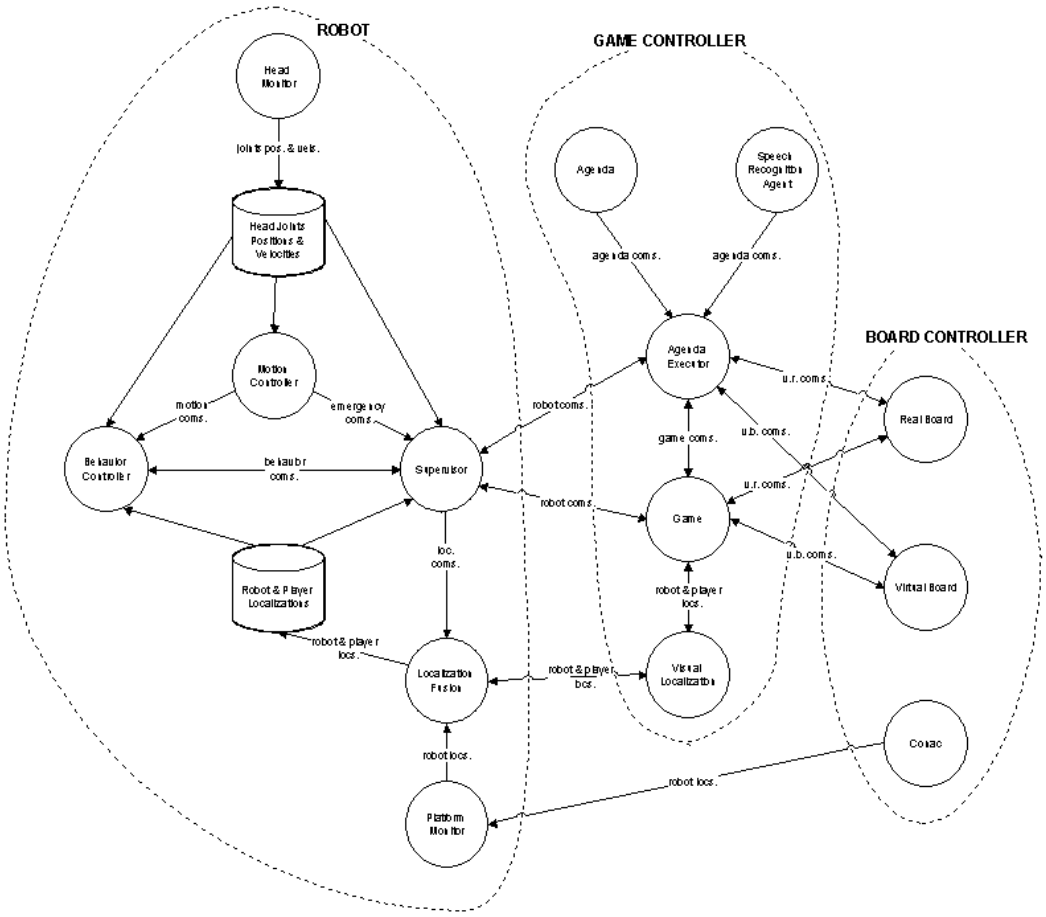


Figure6. Software architecture under CAV.

software control architecture that must guarantee the correct functioning of the different parts of the system both hardware and software. In our opinion, these facts emphasize the importance of a suitable control architecture and associated design implementation that must hold the extensibility and easy integration demands of these systems.

Related to the hardware, it has proven to be the weakest point of the system, and in fact broken hardware has been the only reason that has avoided that Eldi's show could take place. Some critical elements are devices that are not easily obtained due to they have a low demand, and additionally the future of the hardware supplier companies can also affect the evolution of the system drastically.

The first and perhaps most important lesson learnt from the Eldi project is that a museum robot must be conceived as a living being. Indeed, this situation must be considered in the light that these *pieces* normally capture a great deal of attention from the visitors and it is not unusual that they end up being considered as the *flagship* of the exhibition. A logical consequence is the staff demand to constantly update the shows or add new capabilities to the robot to renew the interest of the

public and attract new visits. Surprisingly, we have observed, as other authors [9], that it is the emotional and expressive abilities of the robot what captures much of people's attention and not its navigational or obstacle avoidance capabilities. Most people do not realize (and do not mind) if the robot is avoiding obstacles or not. People enjoy frequently catching robot attention (cameras) and seeing themselves on videoboard or on Eldi's screen. In our opinion, a clear indication of the type of expectations these type of robots poses on the general public.

References

[1] Toschi Doi citation, excerpt from Computists' Weekly, AI Brief section in Intelligent Systems, December 1999.

[2] The Sage project's URL: http://www.cs.cmu.edu/~illah/SAGE/index.html

[3] The Rhino project's URL:http://www.informatik.uni-bonn.de/~rhino/

[4] The Kismet project's URL: http://www.ai.mit.edu/projects/humanoid-robotics-group/kismet/kismet.html

[5] The Minerva project's URL: http:// http://www.cs.cmu.edu/~minerva/

[6] A.C. Domínguez-Brito, F. Mario Hernández-Tejera and J. Cabrera-Gómez, "A Control Architecture for Active Vision Systems", Proc. of VIII Spanish National Symposium on Pattern Recognition and Image Analysis, Vol. I, 395-402, 1999.

[7] M. Hernández, J. Cabrera, M Castrillón, A. Domínguez, C. Guerra, D. Hernández, and J. Isern, "Deseo: An Active Vision System for Detection, Tracking and Recognition", in Computer Vision Systems, H. I. Christensen (Ed.), Vol. 1542 of Lecture Notes in Computer Science, Springer-Verlag, 1999.

[8] M. Wooldridge and N. R. Jennings, "Intelligent Agents: Theory and Practice", Knowledge Eng. Review, 10(2), 1995.

[9] W. Burgard, A.B. Cremers, D. Fox, D. Haehnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "Experiences with an Interactive Museum Tour-Guide Robot", Technical Report CMU-CS-98-139, Carnegie Mellon University, Pittsburgh, PA, 1998. Available through URL: http://www.cs.cmu.edu/~thrun/papers/thrun.tourguide_tr.pdf

[10] M. Steenstrup, M. A. Arbib and E. G. Manes, "Port Automata and the Algebra of Concurrent Processes", Journal of Computer and System Sciences, Vol. 27, 29-50, 1983.

[11] Antonio C. Domínguez-Brito, Magnus Andersson and Henrik I. Christensen, "A Software Architecture for Programming Robotic Systems based on the Discrete Event System Paradigm", Technical Report CVAP 244, Centre for Autonomous Systems, KTH - Royal Institute of Technology, S-100 44 Stockholm, Sweden, September 2000.

Fuzzy Logic-Based Robot Navigation in Unknown Environments

Humberto Martínez, Antonio G. Skarmenta, and Miguel A. Zamora

Depto. de Ingeniería de la Información y las Comunicaciones
Universidad de Murcia
e-mail: {humberto,skarmeta,mzamora}@um.es

Abstract This paper outlines the problem of a mobile robot navigating in an unknown environment. This is solved in a two step process by the application of two navigation techniques. In the first step a fuzzy grid map is applied for the map-building problem, and an A* search method is applied for the path-planning problem. In the second step a fuzzy segments map is applied for the map-building and localisation problems. Some results in both simulator and real robot are presented.

1 Introduction

For a mobile robot, the ability to navigate is one of the most important of all. Staying operational, for instance avoiding dangerous situations such as collisions, come first. But if any tasks which relate to specific places in the robot's environment are to be performed navigation is a must. Navigation can be defined as the combination of three fundamental competences: map-building,self-localisation and path-planning.

As can be noted, these competences are tightly coupled. If one of them fails, the robot performance is greatly degraded or reduced. Often, one or more of those competences may not be present in a given robot, usually when it is working in known and structured environments and when the robot is provided with a priori information or special sensor systems. One of the objectives of our work is to work on unknown environments, and thus, the robot is provided with the three competences. The following sections describe some basic techniques to develop and implement these competences.

2 Map building and path planning

The first navigation procedure relies on the assumption of reliable odometry for short time and distance operations. With this assumption in mind, the robot is provided with a map building method and a path planning algorithm. The robot does not know the environment a priori, and thus, it has to perform both tasks simultaneously. A fuzzy grid map is used for map building, while the path planning is based on the A* algorithm. As opposed to other navigation schemas, the robot is required to build the maps while it is moving, and thus approaches based on non-continuous movement (the robot maps the area, then generates a short distance map, and then moves that distance) are not desired. The rationale for this requisite

is to exhibit some kind of reactivity so that the robot seems to perform more animal or human like.

The proposed navigation procedure is as follows. Each time new sensor values are available, the map is updated. This step, which is not very time consuming, is performed at a given frequency (approximately every 300 ms). The robot simultaneously is moving and with a certain periodicity a new path is computed based on the current map. This step is performed at a slightly lower frequency (approximately every 1200 ms), because the path planning is comparatively more time consuming. The different algorithms are explained below.

2.1 Fuzzy grid map

The choice of internal map representation depends on various characteristics but mainly on the environment itself (obstacles density, rooms dimension, etc.). Lee (Lee, 1996) makes a classification attending to the range of geometric properties that can be derived from the map. Considering the properties of an indoor high density environment, a metric map may be considered as the most appropriate choice because of the compact representation of the environment. The square cells grid maps are the most widely used metric maps (Moravec, 1985), (Borenstein, 1988), (Lim, 1992). Although they have many shortcomings, the advantage of employing grid maps is that path planning is quite direct and simple. In recent years, different tools based on neural networks (Kurz, 1996), (Thrun, 1998) or fuzzy logic (Oriolo et alt., 1998) have been introduced to build grid maps based.

The method used for map building is based on fuzzy grid maps (Oriolo et alt., 1998). Each cell in the map has two values associated: the degree of certainty that the cell is empty ε_{ij} and the degree of certainty that the cell is occupied ω_{ij}. These values are calculated for each cell in the beam of a single sensor. As sonar and infrared sensors have different aperture widths, that of the wider sensor (sonar) is used.

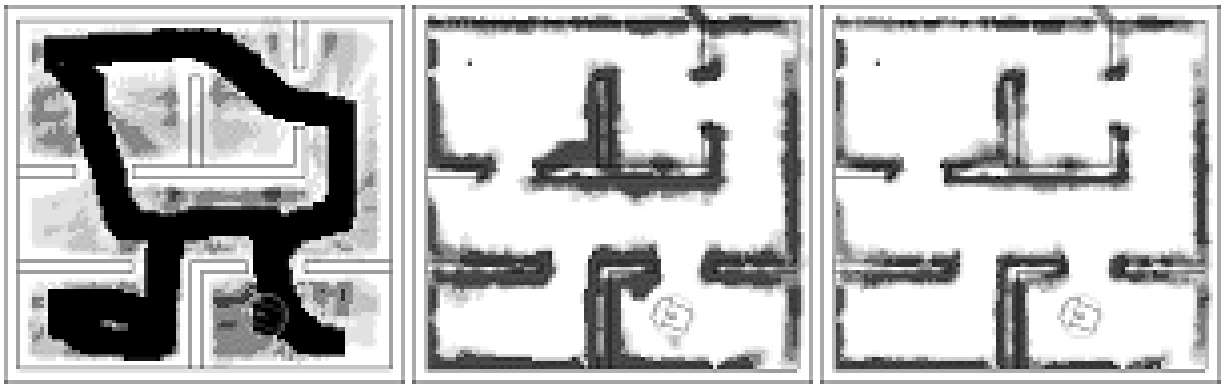
The ε_{ij} and ω_{ij} values are then aggregated to the previous values of each cell, to obtain the aggregated degree of certainty that the cells are empty *E(i,j)* or occupied *O(i,j)*.

As these *E(i,j)* and *O(i,j)* correspond to fuzzy degrees of membership, such information is combined in an additional value for each cell *M(i,j)*, which represents, in a conservative way, if the cell is occupied. The meaning can be defined precisely based on its definition: the degree of certainty that the cell is empty, is not occupied, and is not ambiguous.

Using this method, the robot uses and maintains three different maps:

- The map of cells that are likely to be empty (Figure 1(a)), which is constructed using the *E(i,j)*. Darker cells correspond to a higher possibility of being empty, while lighter cells correspond to a lower possibility.

- The map of cells that are likely to be occupied (Figure 1(b)), which is constructed using the *O(i,j)*. Darker cells correspond to a higher possibility of being occupied, while lighter cells correspond to a lower possibility.
- The navigation map (Figure 1(c)), which is constructed using the *M(i,j)*. This map is the one that is later used for path planning. Darker cells correspond to a higher certainty of being an obstacle, while lighter cells correspond to a higher certainty of being empty.



(a) (b) (c)

Figure1. (a)Empty cells map, (b)Occupied cells map, (c)Navigation map.

One important point to be noted here is derived from the fact that the original method (Oriolo et alt., 1998) only takes into account sonar sensors, modelling the uncertainty directly on the map. In this case, virtual sensors are used to build the map instead of using sonars directly. These virtual sensors fuse sonar and infrared sensors to obtain a more reliable measure . The fusion method (Martínez Barberá et alt, 2000) works as a sonar with distances bigger than 80 centimetres. With smaller distances it works as an infrared (we currently use Sharp GP2D02 infrared sensors, which are quit robust against lighting conditions). In this case the infrared beam is more or less similar to the sonar one if it is projected on a cell map, because the beam is discretised on 10-20 centimetres cells. Thus, the map model is still valid.

Another important difference is the map initialisation. The original method was considered for use in a sense-map-plan-move approach, and the unexplored area is modelled as an ambiguous ($\varepsilon_{ij} = 1.0$, $\omega_{ij} = 1.0$) or occupied area ($\varepsilon_{ij} = 0.0$, $\omega_{ij} = 1.0$). As has been stated above, this behaviour is not desired and consequently the path-planning procedure is different. This requires that the unexplored area is modelled as empty area ($\varepsilon_{ij} = 1.0$, $\omega_{ij} = 0.0$) so that tentative paths may cross unexplored areas if necessary.

Finally, all the cells occupied by the robot are set to the maximum values of certainty that the cells are empty ($\varepsilon_{ij} = 1.0$) and not occupied ($\omega_{ij} = 0.0$). Trivially,

if the robot is over these cells, there is not any obstacle there. The influence of this strategy can be noticed in the maps shown above (Figure 4.2) as very dark cells over the robot's trajectory.

2.2 A*-based planning

The most direct method for path-planning using grid maps is to consider each cell as a node in a graph and then apply a graph search algorithm. The A* algorithm (Hart et alt., 1968) allows heuristic information to be incorporated when available, resulting in an efficient search. Let $h^*(n)$ be an estimate of the cost of an optimal path from n to the goal t, and let $g^*(n)$ be the cost of the path from s to n with minimum cost so far found. Taking a fuzzy grid map $M(i,j)$, as described in the previous subsection, the path planning problem can be stated in the following way: the robot has to navigate from an initial cell s' to a goal cell t'. Instead of looking for a path starting in the robot location, the A* is applied to look for a path starting at the goal cell. In this way, the starting position s' corresponds to the A* goal t, and the goal position t' corresponds to the A* start s. There are approaches to solve this problem taking into account sensor noise, the robot size and minimum length paths (Oriolo et alt., 1998) but not simultaneously. In this paper a different approach has been taken to find minimum length paths while providing safe clearance of obstacles. Using the previous problem statement, the $h^*(n)$ function (Eq. 1) is defined as follows:

$$h^*(n) = \sqrt{(s_x - n_x)^2 + (s_y - n_y)^2} \tag{1}$$

where $(s_x,\ s_y)$ and $(n_x,\ n_y)$ are the coordinates of the cells s and n respectively. Trivially, the condition holds true as do the local consistency criteria, because this $h^*(n)$ is the euclidean distance of the most direct path between s and n. The definition of the $g^*(n)$ is more complex as it is computed in a two step process: first, the fuzzy map is dilated locally around the cell a, and second, the resulting cost is augmented. The dilation step $dil(a)$ is performed as follows (Eq. 2):

$$dil(a) = \frac{\begin{array}{c} min \\ i \in [a_x - \delta, a_x + \delta] \\ j \in [a_y - \delta, a_y + \delta] \end{array} \{M_{ij}\}}{k_{dil}} \tag{2}$$

where δ is the size of the dilation window, and k_{dil} is the dilation weight. The augmentation step $aug(x)$ is performed as follows (Eq. 3):

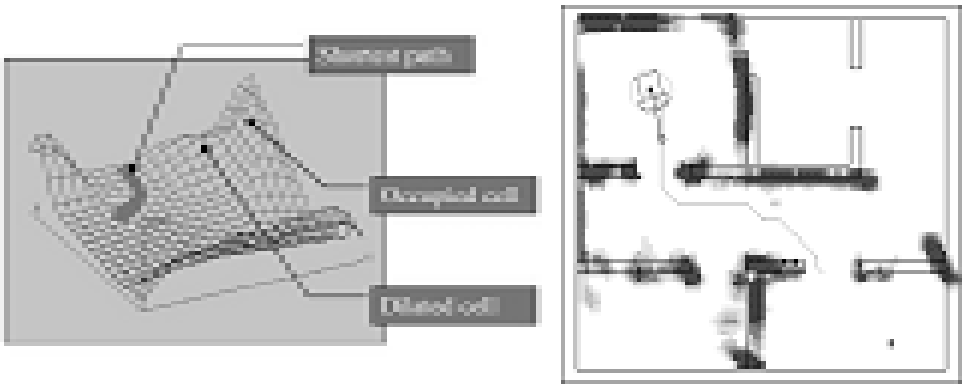
$$aug(x) = e^{k_{aug} \cdot x} \tag{3}$$

where k_{aug} is the augmentation weight. Finally, the $g^*(n)$ function is recursively computed as follows (Eq. 4):

$$
\begin{aligned}
&v(n) = aug(dil(n)) \\
&g^*_{inc}(n, n+1) = \begin{cases} v(n) & n_x = (n+1)_x \wedge n_y = (n+1)_y \\ \sqrt{2} \cdot v(n) & otherwise \end{cases} \\
&g^*(n) = \begin{cases} g^*(s) = 0 \\ g^*(n+1) = g^*(n) + g^*_{inc}(n, n+1) \end{cases}
\end{aligned} \tag{4}
$$

where *v(n)* is the composition of the dilation and augmentation steps, and $g^*_{inc}(n, n+1)$ is the incremental step for the computation of $g^*(n)$. The incremental function $g^*_{inc}(n, n+1)$ computes the cost of traversing from cell n to $n+1$, where $n+1$ is the next cell in the path to the goal. Basically, the function returns *v(n)* if both cells are in the same horizontal or vertical, or $\sqrt{2}v(n)$ if the cells are diagonally aligned.

The dilation function serves to fill small gaps in the maps so that tentative paths do not cross walls or small openings. By choosing different values of δ, different robot radii or sizes can be taking into account. The augmentation function exponentially spreads the degrees of certainty that cells are occupied or empty so that small certainty values give similar results while high certainty values give much bigger results. The idea is to avoid the use of occupied cells for path planning, and thus, this augmentation helps the use of lower certainty values *M(i,j)* when possible. The dilation function is applied only locally in orer to avoid applying the transformation to the whole map, because plans usually involve small portions of the map and the dilation process is time consuming. A sample of the effects of the evaluation function $f^*(n)$ is shown below (Figure 2(a)). The dilation effect is observed as peaks with half the height of their surrounding. The augmentation effect is observed as the difference between cells with different certainty values.



(a) (b)

Figure2. (a)Evaluation function $f^*(n)$, (b)A* based path.

Using the previous A* based planning scheme, the navigation strategy uses the computed path to set a look-ahead point which is used as a short-term goal location. This look-ahead point is defined as the point of the path from s to t which is at a distance of δ_{look} cells from s. The distance δ_{look} depends on both the robot's average speed and the navigation control cycle frequency. For the Quaky-Ant robot, the following navigation parameters are used: $k_{dil} = 2.0$, $\delta = 2$, $k_{aug} = 5.0$, $\delta_{look} = 10$. A sample path is show above (figure 2(b)), where the dilation effect (the path leaves enough clearance near the walls) and the look-ahead point (small circle in the front of the robot) can be noticed.

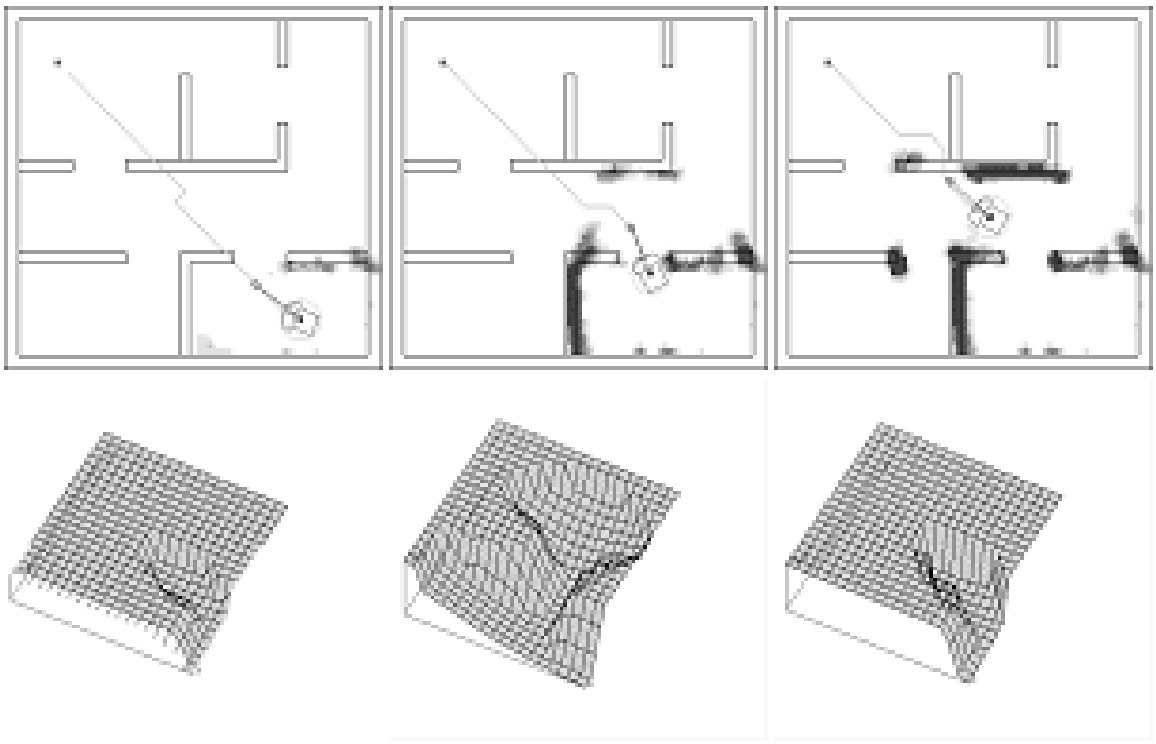
2.3 Experiments and results

Different experiments have been conducted to test the simultaneous map-building and path-planning. One of these tests is shown below (Figure 3). The Quaky-Ant robot is simulated in autonomous mode, and has to navigate from a given home position (bottom-right corner) to a goal location (top-left corner) in an a priori unknown environment. The different slides show the evolution of the fuzzy grid map (top image) and the A* evaluation function $f^*(n)$ (bottom image). The world map is superimposed on map images for reference. Cells that are not expanded are represented with the maximum value of $f^*(n)$ found so far.

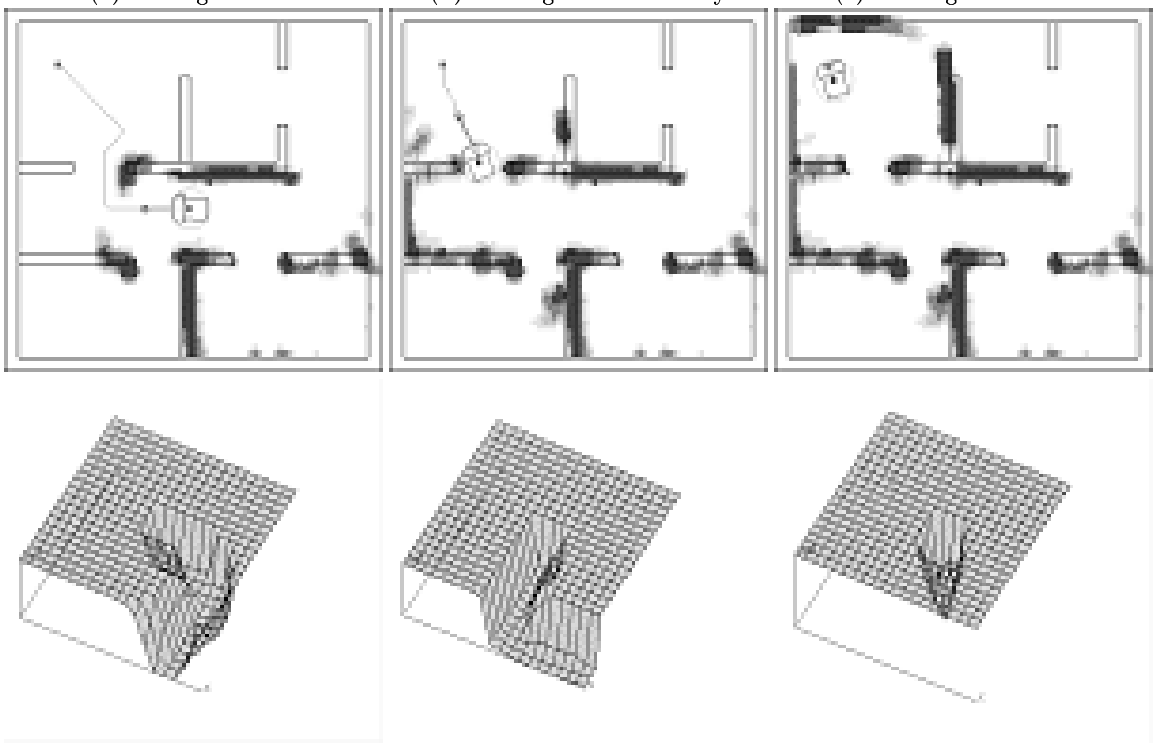
The robot starts (position "a") with no previous knowledge, and thus the path to the goal is an almost straight line. As the robot adds new features to the fuzzy grid map the path consequently avoids the modelled walls (position "b"). Due to sensor noise, parts of the walls are not modelled, and in this way paths can be generated through them if the corresponding holes are bigger than the robot (position "c"). If more evidence about the wrong walls is acquired, even if they are not completely continuous, the dilation step prevents it from crossing the walls with a path (position "d"). When the local environment is almost fully modelled (position "e"), the path is traced over safe positions until the robot reaches the goal (position "f").

The experiment area has a dimension of 5.4 x 4.95 metres, and it is tessellated in 0.075 metres wide cells, resulting in a grid map that contains 4,752 cells. The following table (Table 1) shows the number of cells in the path, the number of expanded nodes and the time taken to expand the nodes in each of the previous navigation slides (Figure 3). The test has been performed in a Mac PowerPC G3 running at 300 MHz on top of the BGen environment, which has been developed in Java.

In addition to simulated experiments, the navigation approach has also been tested in real world environments. One of these is shown below (Figure 4). The mapped area has a dimension of approximately 8 x 8 metres, and it is composed of wooden walls (bottom-left room) and standard-building walls (top-right corridor). The cleanness of the map and the safe path generated are points to notice.



(a)Starting the run (b)Crossing the 1° doorway (c)Entering the corridor



(a)In the corridor (b)Crossing the 2° doorway (c)Reaching the goal

Figure3. Simultaneous map-building and path-planning (simulated).

Position	Path	Expanded	Time(ms)
a	52	52	5
b	53	814	100
c	48	579	148
d	40	374	23
e	20	99	7
f	5	5	1

Table1. Number of expanded cells

Figure4. Simultaneous map-building and path-planning (real world).

3 Map building and localisation

The first navigation procedure relied on the assumption of reliable odometry for short time and distance operations. This is clearly an important constraint in large environments or in operations requiring long time. To overcome this limitation, the previous system is complemented with a localisation method based on a fuzzy segments map. Global grid maps are not suitable for localisation, while local grid maps (based on sonar signature matching) require a precise compass reference to obtain reliable results (Duckett and Nehmzov, 1998). In addition, the previous approach if provided with a reliable localisation performs fast and produces safe paths. Taking into account these facts, it was decided to improve the system adding a separate technique for localisation which did not need an a priori environment model.

The new navigation procedure is as follows. The fuzzy grid map and the A* path planner are executed in a similar way as described in the previous section, but instead of using positions returned by the odometry they use the corrected localised positions. Simultaneously, a local fuzzy segments map is updated each time a sensor buffer is full (approximately every 1200 ms). This is compared (approximately every 2500 ms) to a global fuzzy segments map to compute the differences and thus obtain the current odometry drift, which is then corrected. Both the local and global maps are constructed from scratch, without any prior knowledge. The different algorithms are explained below.

3.1 Fuzzy segments map

The method used for map building is based on the fuzzy segments map (Gasós and Martín, 1996b)(Gasós, 2000), which is a geometric representation composed of line segments only. By carefully managing the uncertainty related to the robot's position and sensor noise, the method tries to overcome some of the problems typically related to geometric models. In this way, a fuzzy segments map vastly reduces the lack of stability by decreasing the expressive power of the line fittings keeping the uncertainty all the way along the process

The original work included a method to generate segments based on single sensor scans. The idea was to obtain some measurements form a single sensor. These measurements were then preprocessed to eliminate easily detectable misreadings and to group together smoothly changing measurements. Finally, each group of measurements was fitted to a straight line to obtain fuzzy segments. This approach (Figure 5) combines the information from different sensors at the segment level. If two segments from two different sensors are collinear, the segments are combined and merged into a new segment.

While the original approach generates good maps, it lacks the ability to combine measurements from different sensors at a sensor level. This makes the map generation process slow, because much time is needed to obtain good sensor measurements to fit a straight line. The previous example (Figure 5) shows the trajectory followed by the robot (yellow line), the fuzzy segments (blue and grey lines), and some sonar

scans (orange dots). As can be clearly seen, the bottom-right measurements could be fitted with a line, but as the points come from different sensors, the previous approach does not generate a fuzzy segment there. To overcome this problem, a new segment generation method has been developed.

The new method builds and maintains a circular sensor buffer in a heuristic way, which is described below. Let n be the number of sensors, and let m be the number of measurements stored in the buffer for each sensor. The number of entries in the sensor buffer is $t = n$ x m. When a series of new values for the n sensors is available, those measurements that are smaller than a given threshold δ_{len} overwrite the oldest previously stored values sequentially. The sensor buffer is shown below (Figure 6), where light grey cells are the old values and dark grey cells are the most recent sensor readings. The buffer thus constructed serves to extract candidate points for the line fitting procedure.

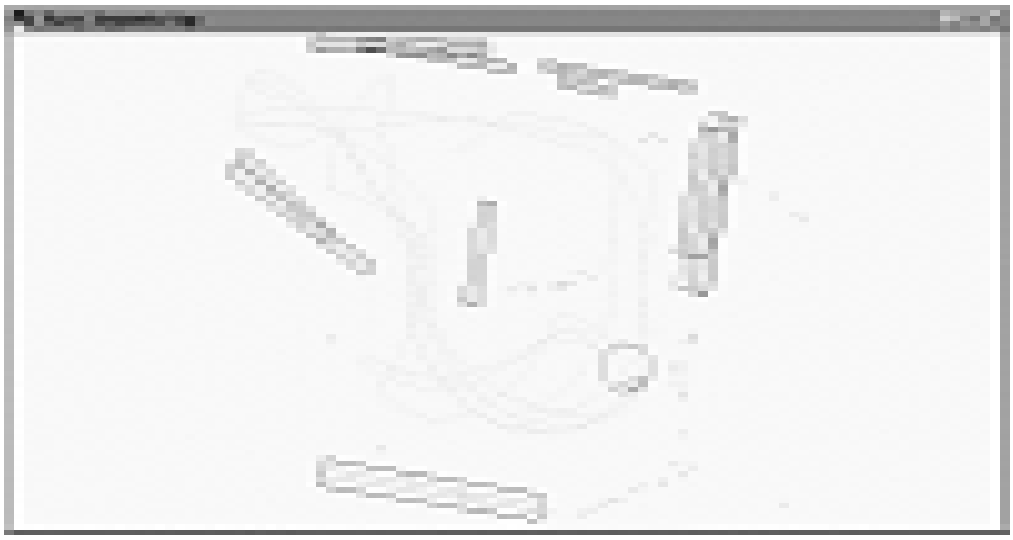


Figure5. Fuzzy segments map from single sensors.

To generate a fuzzy segment, at least four points are required to be considered them as a line. The algorithm to generate a line fitting is as follows:

s1 *Initialisation.* Let $O = \{o_1, o_2, \ldots, o_t\}$ be the sensor buffer. Let P be the set of candidate points that constitute a segment. Let i be an arbitrary cell in the sensor buffer, which for simplicity is set to 1. Let δ_{max} be the maximum distance between consecutive points on a line. Let δ_{gap} be the maximum number of outliers in a segment, that is, the number of cells in a sequence that do not belong to a line. Let δ_{ang} be the maximum difference in orientation between two lines for them to be considered as collinear.

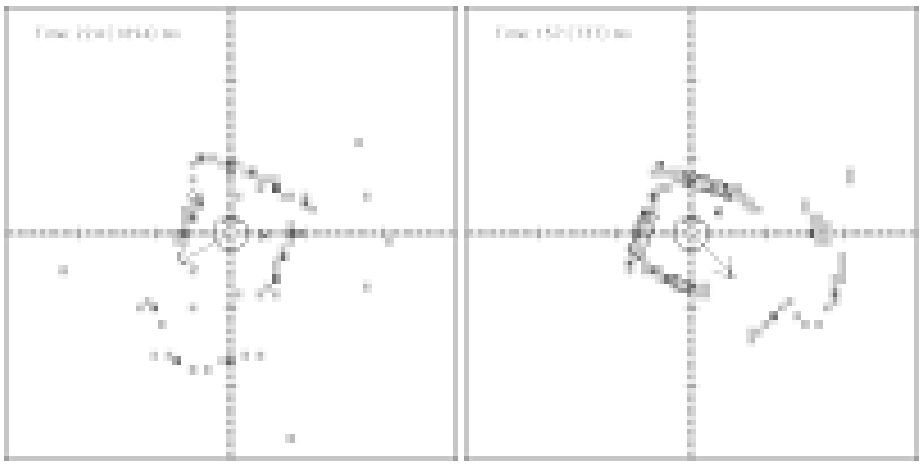


Figure6. Raw sensor buffer (left) and sensor buffer with segments (right) .

s2 *First point.* Let i_{last} be equal to i, let i_{next} be equal to $i + 1$, and let g be equal to 0. The value o_i is set as the first candidate point to form the line, and consequently $P = \{o_i\}$. Increment the index i.

s3 *Second point.* If $\|o_i, P_1\| < \delta_{max}$ then let i_{last} be equal to i, let $P = P \cup \{o_i\}$, increment i, and go to step s4. Otherwise, increment i and g. If g ¿ δ_{gap} then let i be equal to i_{next}, and go to step s2. Otherwise, repeat step s3.

s4 *Third point.* Let α be the orientation of the line $\overline{P_1P_2}$, and β be the orientation of the line $\overline{P_2o_i}$. If $\|o_i, P_2\| < \delta_{max}$ and $|\alpha - \beta| < \delta_{ang}$then let i_{last} be equal to i, let $P = P \cup \{o_i\}$, increment i, and go to step s5. Otherwise, increment i and g. If g ¿ δ_{gap} then let i be equal to i_{next}, and go to step s2. Otherwise, repeat step s4.

s5 *Extra points.* Let u be the number of point in P. Let α be the orientation of the line $\overline{P_{u-1}P_u}$, and β be the orientation of the line $\overline{P_uo_i}$.

- **s5.1** If $\|o_i, P_u\| < \delta_{max}$ and $|\alpha - \beta| < \delta_{ang}$then let i_{last} be equal to i, let $P = P \cup \{o_i\}$, increment i. If $i = t$ then generate a fuzzy segment with P and go to step s2. Otherwise, repeat step s5.
- **s5.2** Otherwise, increment i and g. If g ¿ δ_{gap} and $u = 4$ then generate a fuzzy segment with P and go to step s2. Otherwise, if g ¿ δ_{gap} then let i be equal to i_{next}, and go to step s2. Otherwise, repeat step s5.

This method obtains heuristically more consistent segments in a given environment than the original approach making fewer movements. The method has been tested with the following values: $n = 16$, $m = 10$, $\delta_{len} = 2$ metres, $\delta_{max} = 0.2$ metres, $\delta_{gap} = 10$, $\delta_{ang} = 15^o$. An example shows below (Figure 7) the trajectory followed by the robot (yellow line), the fuzzy segments (blue and red lines), and some sonar scans (grey dots). As can be seen, the trajectory followed by the robot is much smaller than in the previous example (Figure 5) given the same environment, while

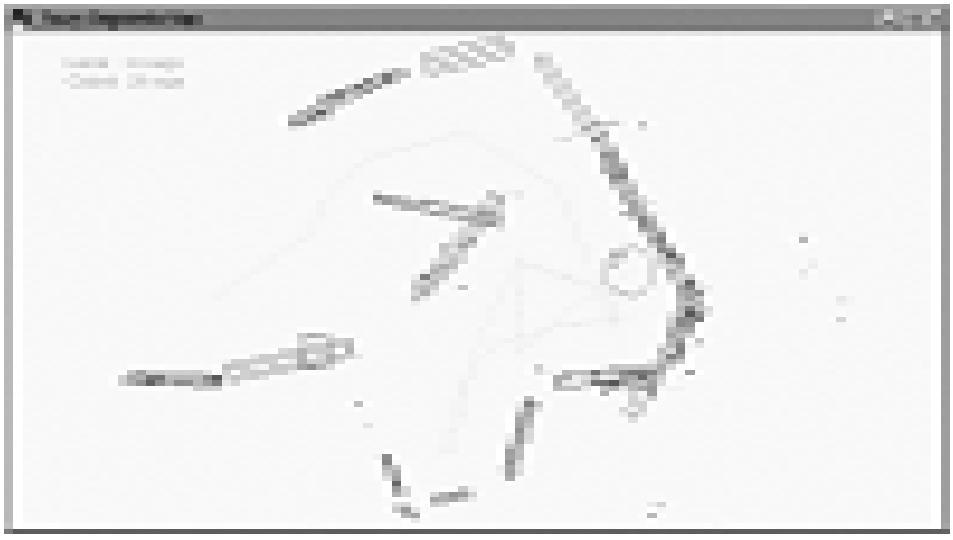


Figure7. Fuzzy segments map from sensor buffer.

there are many more consistent segments. This is a key point for achieving a good localisation (see next subsection).

3.2 Fuzzy segments localisation

The fuzzy segments map representation is used for self-localisation (Gasós and Martín, 1996a). While the robot is navigating in an environment in order to accomplish a task, the sensor observations are used to build a *partial* fuzzy segments map of the robot surroundings. This map is compared to the *global* fuzzy segments map, which has been previously constructed. When the matching results provide enough evidence of the coincidence of both maps, the robot location is updated based on their differences. This localisation task is run as a continuous process during all the time the robot is moving, in order to continuously update its position.

The global map is built from scratch while keeping track of the areas that have been already visited. For this task a coarse binary grid (1 x 1 metre cells) represents the areas visited by the robot. While the robot moves through areas not yet visited, it adds new segments to the global map and the cells are marked as visited. When the robot arrives at visited areas, it adds the new segments to the local map. The local and global maps are compared until a significant number of local segments match those in the global map. Then, the transformation that brings both maps together is used to estimate the dead reckoning errors accumulated in the time span between the construction of the maps. In this case, the local map is added to the global map, allowing changes in the environment to be completed and updated, and the process continues in the same way.

One key point in the localisation process is the correct matching between the local and global map. If there are not enough perpendicular segments, the matching

is not considered.In other words, if only parallel segments match there is not enough certainty in the position and the localisation is not considered. Clearly, the greater the number of correct segments that are generated, the better the localisation results. This is the main benefit achieved with the segment generation scheme described in the previous subsection

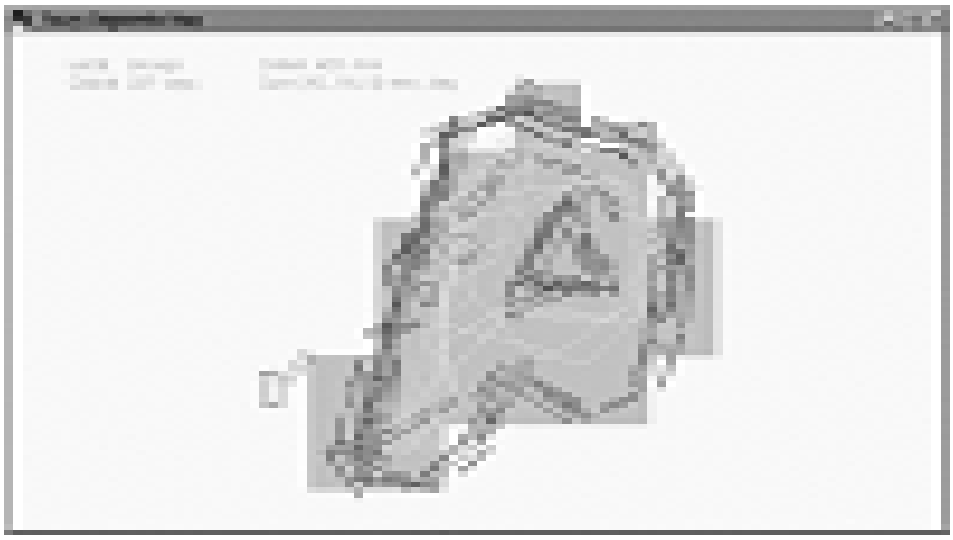


Figure8. Fuzzy segments localization.

An example is shown above (Figure 8), which represents the robot trajectory (yellow line), the current local map (blue segments), the visited areas map (light grey boxes), and the global map (dark grey segments). The important odometric error in the fifth complete round in the environment can be noticed. Previous errors were in the same magnitude. The localisation succeeds in finding the transformation between the local map and the global map.

3.3 Experiment and results

Different experiments have been conducted to test the simultaneous map-building and self-localisation. One of these tests is shown below (Figure 9). The Quaky-Ant robot is simulated in autonomous mode, and has to navigate from a given home position (bottom-right corner) to a goal location (top-left corner) and then return to the home position in an a priori unknown environment. The different slides show the evolution of the sensor buffer (top image) and the local and global fuzzy segments maps (bottom image).

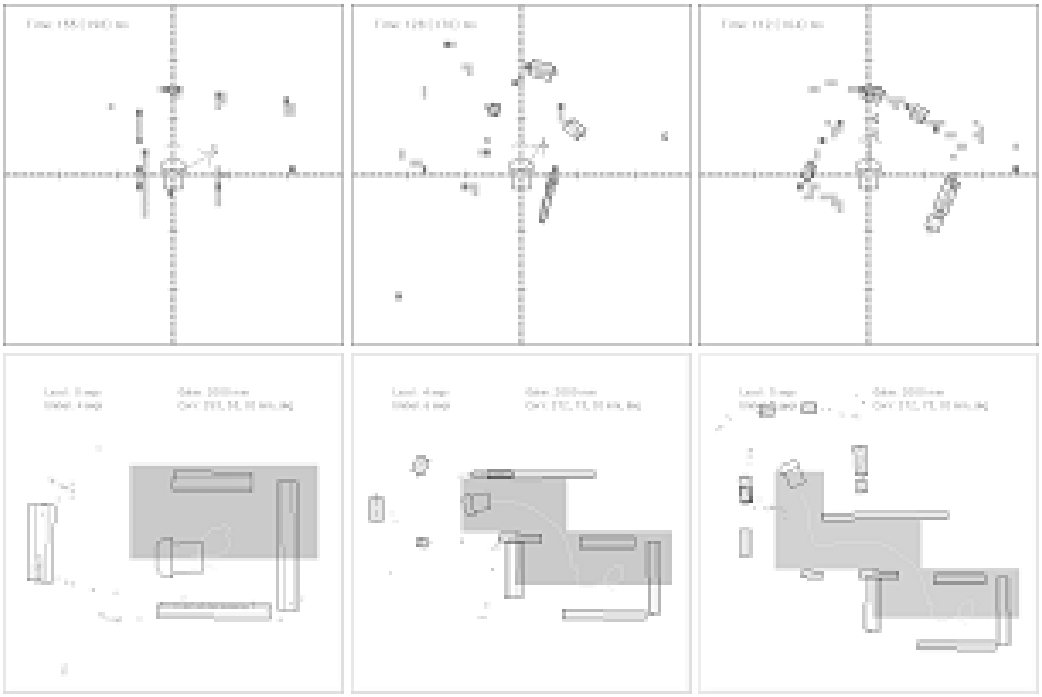
The robot starts (position “a”) with no previous knowledge, and updates the global map while it is moving through unexplored areas (positions “b” and “c”). When the robot reaches a visited area it starts matching the local and global map

(position "d") and correcting the odometry (position "e") until it reaches the home position (position "f").

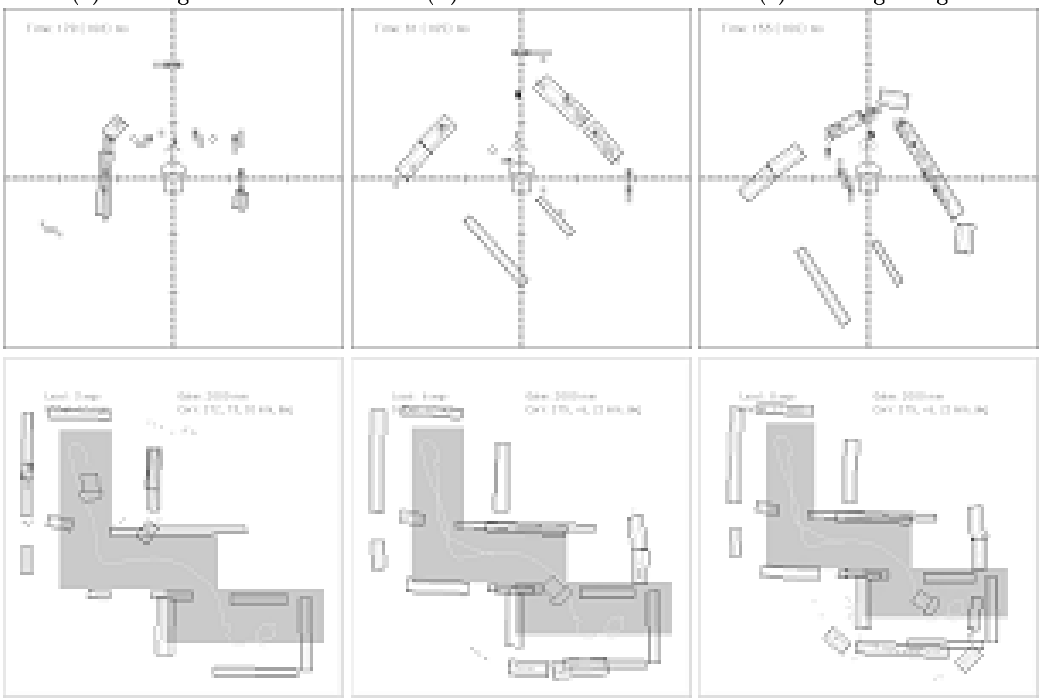
4 Conclusions

Two navigation schemes have been developed to allow the Quaky-Ant robot to navigate in a priori unknown environments. The first procedure relies on the assumption of reliable odometry for short time and distance operations. Bearing this assumption in mind, the robot is provided with a map building method and a path planning algorithm, which are run simultaneously and thus the robot is required to build the map while it is moving resulting in a continuous movement. This approach is based on a fuzzy grid map for map-building and an A* based algorithm for path-planning. The fuzzy grid map has been modified to accomplish the required task: the virtual sensors are used instead of sonars, the initialisation procedure has been changed, and some heuristics have been incorporated to modify the certainty of visited cells. The A* based planner has been defined with a new evaluation function to generate paths that leave enough space between the robot and the obstacles, and also to be efficient in time so that the node evaluation is fast enough to run the planner at a high frequency. The navigation scheme has been successfully used in different, both simulated and real world, scenarios.

The second procedure overcomes the limitation of the previous approach by adding a localisation method. This is accomplished by building fuzzy segments map to model the previously visited environment and the current sensed environment. The transformations needed to match these maps are used to correct the current robot's position. The performance of the localisation method is greatly affected by the number and quality of the fuzzy segments. The segments generation procedure has been modified to make use of a sensor buffer, which produces more consistent segments. Although the motivation behind navigation procedure was to make the overall work of the system more robust, it also proved to be capable of solving the SLAM (Simultaneous Localisation And Map-building) problem in the different tests carried out. More tests are still needed to verify or prove in which situations or environments this set-up is valid for the SLAM problem.



(a)Starting the run (b)In the corridor (c)Reaching the goal



(a)Returning to the home (b)Crossing a doorway (c)Reaching the home

Figure9. Simultaneous map-building and self-localisation.

References

[1] Borenstein, J. & Koren, Y. (1988). *Obstacle Avoidance with Ultrasonic Sensors*, IEEE J. Robotics and Automation, 4(1):213-218

[2] Duckett, T. & Nehmzow, U. (1998). *Mobile Robot Self-Localisation and Measurement of Performance in Middle-Scale Environments*, Journal of Robotics and Autonomous Systems, 24(2):57-69

[3] Gasós, J. & Martín, A. (1996a). *Mobile Robot Localization using Fuzzy Maps*, in Lectures Notes in Artificial Intelligence (A. Ralescu and T. Martín eds.), Springer-Verlag, Germany

[4] Gasós, J. & Martín, A. (1996b). *A Fuzzy Approach to Build Sonar Maps for Mobile Robots*, Computers in Industry, 32(1):151-167

[5] Gasós, J. (2000). *Integrating Linguistic Descriptions and Sensor Observations for the Navigation of Autonomous Robots*, in Fuzzy Logic Techniques for Autonomous vehicle Navigation (D. Driankov and A. Saffiotti eds.), Springer-Verlag, Germany

[6] Hart, P.E.; Nilsson, N.J. & Raphael, B. (1968). *A Formal Basis for the Heuristic Determination of Minimum Cost Paths*, IEEE Trans. on Man, Systems, and Cybernetics, 4(2):100-107

[7] Kurz, A. (1996). *Constructing maps for mobile robot navigation based on ultrasonic range data*, IEEE Trans. on System, Man, and Cybernetics, 26(2):233-242

[8] Lee, D. (1996). *The Map-Building and Exploration Strategies of a Simple Sonar-Equipped Robot*, Cambridge University Press, Cambridge

[9] Lim, J.H. & Cho, D.W. (1992). *Physically Based Sensor Modelling for a Sonar Map in a Specular Environment*, IEEE Intl. Conf. on Robotics and Automation, pp 1714-1719

[10] Martínez Barberá, H.; Gómez Skarmeta, A.; Zamora Izquierdo, M. & Botía Blaya, J. (2000). *Neural Networks for Sonar and Infrared Sensors Fusion*, 3^{rd} Intl. Fusion Conference, Paris, pp 18-25 (WeB4)

[11] Moravec, H.P. & Elfes, A. (1985). *High Resolution Maps from Wide angle Sonar*, IEEE Intl. Conf. on Robotics and Automation, Washington, USA, pp 116-121

[12] Thrun, S. (1998). *Learning Metric-Topological Maps for Indoor Mobile Robot Navigation*, Artificial Intelligence, 99(1):21-71

[13] Oriolo, G.; Ulivi, G. & Venditelli, M. (1998). *Real-Time Map Building and Navigation for Autonomous Mobile Robots in Unknown Environments*, IEEE Trans. on Systems, Man and Cybernetics-Part B: Cybernetics, 3(28):316-333

Sesión 3: Visión artificial

Murphy: hacia un robot con visión estereoscópica

P. Bustos, P. Bachiller, J. Vicente, M. Broncano, C. Fernández

Departamento de Informática
Universidad de Extremadura
e-mail: pbustos@unex.es

Resumen La percepción de un entorno tridimensional en un robot móvil es aún un problema abierto para la comunidad científica. Hay dos cuestiones claves que motivan la dificultad para resolver este problema. La primera es la relación existente entre la percepción de un espacio sin objetos y la percepción de los objetos. La segunda cuestión hace referencia al concepto de objetos y a la manera de llegar a ellos. Recientemente se han realizado avances sobre la teoría geométrica de la visión estereoscópica y la visión en movimiento, que permiten iniciar nuevas líneas de exploración de estas cuestiones. La propuesta que se presenta en este artículo se basa en las teorías anteriores para generar la percepción del espacio en un robot móvil (Murphy). El sistema desarrollado es capaz de extraer información 3D visualmente y convertirla en una representación interna, que permite generar el comportamiento físico del robot.

1 Introducción

La construcción de un robot móvil capaz de percibir su entorno tridimensional es un reto que sigue resistiéndose a la comunidad científica. La razón de estas dificultades no es, en absoluto, evidente. Como es habitual existen varias lineas de trabajo, basadas en diferentes hipótesis acerca de cómo se realiza la percepción del espacio. Desde nuestro punto de vista hay dos cuestiones claves: la primera, es la relación entre la percepción del espacio crudo, sin objetos, y la percepción de los objetos. ¿Tiene sentido esta misma distinción? y, si lo tiene, ¿es una anterior a la otra?, ¿es una distinta de la otra?. Tradicionalmente, la neurofisiología ha trabajado con la distinción entre el qué y el dónde tomando posiciones claras ante esta cuestión. Sin embargo, recientemente han surgido nuevas propuestas que sugieren una distinción entre el dónde y el cómo, reemplazando a los objetos por lo que ahora denominamos comportamientos [11]. La segunda cuestión se deriva directamente de ésta y plantea qué es un objeto o un comportamiento y cómo se llega a ellos. Desde el punto de vista de la Visión y, en especial, de la Visión en Robots Móviles la situación es tan poco alentadora como en el caso anterior. En un extremo, los objetos estaban en una base de datos 3D, en el otro, no hacen falta objetos, sólo comportamientos. El problema es que a los comportamientos les podríamos aplicar la misma pregunta sobre su origen.

Recientemente, se han realizado avances muy importantes sobre la teoría geométrica de la Visión Estereoscópica y la Visión en movimiento [6][10]. Estos resultados, junto con el incremento de la capacidad de proceso disponible, con 3DNow [8] por ejemplo, permiten iniciar líneas de exploración de estas cuestionesStandardLa postura que planteamos aquí supone una simplificación metodológica: asumimos que existe

un proceso de percepción del espacio crudo, de la geometría 3D, independiente de los posibles objetos, pero no de ciertos comportamientos básicos. En concreto, la hipótesis es que el robot genera de forma activa una representación 3D del espacio que le rodea en ciclo continuo. En este ciclo, fragmentos de información 3D se integran con la representación existente y, a su vez, ésta genera acciones sobre las articulaciones del robot y sobre los elementos de proceso 2D para mantener, refinar y validar esta representación. Este planteamiento requiere una realimentación continua entre el espacio 2D y el espacio 3D, dentro de la cual se genera comportamiento físico en el robot e interno en los procesos. También, este planteamiento supone que no es posible una reconstrucción completa de la escena, a un nivel de detalle suficiente y en condiciones reales, partiendo de una única pareja de imágenes. Si bien, esto se ha conseguido en el laboratorio con proceso off-line, las técnicas empleadas distan todavía mucho de ser aplicables a un sistema de tiempo y condiciones reales, como es un robot móvil en un entorno desconocido.

En el estado actual de nuestro trabajo, el robot móvil Murphy es capaz de extraer información 3D visualmente y convertirla en una representación interna. Esta representación es, por el momento, únicamente útil para navegación o manipulación en escenas muy simples, pero los resultados obtenidos nos permiten creer que es posible llegar a una reconstrucción densa utilizando el planteamiento antes descrito.

El resto del trabajo se estructura de la siguiente forma: en primer lugar se describe el robot móvil Murphy, dotado de una torreta estereoscópica y que nos sirve como base experimental para estas investigaciones. A continuación se describe la arquitectura de proceso de todo el sistema. En el siguiente apartado se resumen las técnicas básicas utilizadas en la extracción de puntos del espacio, para concluir con una descripción del trabajo en curso sobre el refinamiento y actualización de la representación 3D.

2 El Robot Murphy

Murphy es un robot móvil dotado de seis grados de libertad que se distribuyen de la siguiente forma: dos en la base móvil con un motor en cada rueda motriz, uno en el cuello permitiéndole girar toda la torreta en bloque, uno común a las dos cámara para el movimiento de balanceo y, finalmente, uno en cada cámara para un giro por un microcontrolador PIC que se conecta por el puerto serie a un PC. Las cámaras son de tecnología CMOS, de bajo coste y con un gran campo de visión.

El control de alto nivel del robot se realiza desde el PC a través de un módulo software que contiene la cinemática inversa de la torreta y de la base. De esta forma es posible mandar consignas de posición en, por ejemplo, coordenadas esféricas, de tal forma que ambas cámaras converjan al punto objetivo o sigan una trayectoria 3D.

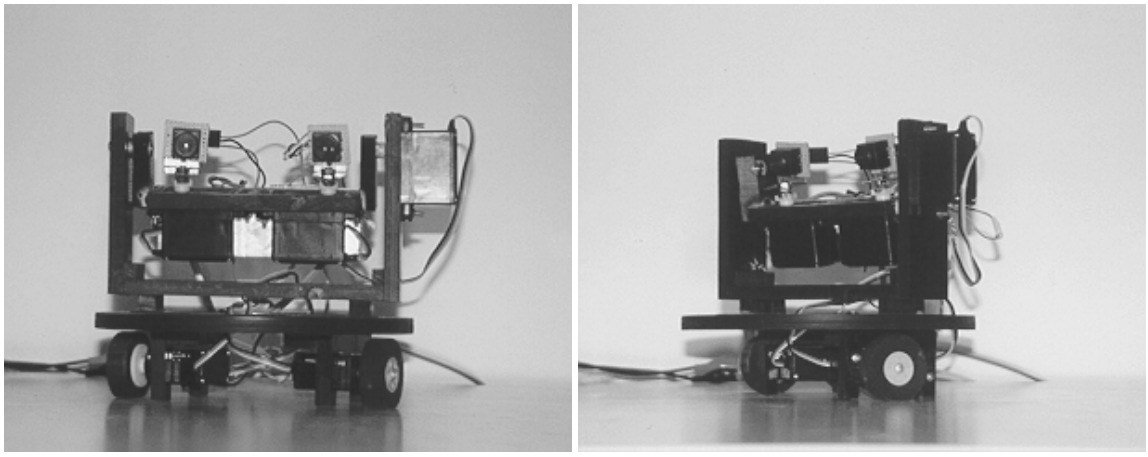


Figura 1. Vista frontal y lateral de Murphy

3 Arquitectura de Proceso

La arquitectura de proceso de Murphy está estructura en tres clases, implementadas en C++. Cada una de estas clases contiene, a su vez, otras clases asociadas a subestructuras del robot y del proceso visual y motriz. La Figura 2 muestra esta estructura.

- Vision2D: recoge la estructura de la torreta estereoscópica: la traslación y También contiene una instacia de la clase Homologos.
  - Homologos: en este objeto se realiza la búsqueda de correspondencias generando una lista de objetos asociados a parejas de esquinas, la triangulación de esquinas formando una malla
  - ElemHomologo: las instancias de esta clase se organizan en una lista. Cada instancia contiene dos punteros a las esquinas correspondientes y atributos donde se almacenan las coordenadas 3D.
- Camara: contiene una instancia del driver de adquisición (v4linux2) y del detector y seguidor de esquinas Correl.
  - Correl: crea y mantiene una lista de objetos, cada uno de ellos asociado a una esquina detectada en la imagen. Para detectar las esquinas en tiempo real[8].
    - Elemcorr: objeto asociado a una esquina con métodos para predecir y buscar su nueva posición en la lista de esquinas suministrada por Correl.
- Rep3D: recibe conjuntos de triángulos 3D y construye y mantiene la representación del espacio exterior.
  - Visor: es un widget para OpenGL en el que se visualiza la lista de triángulos 3D que mantiene Rep3D. El punto de vista se ajusta a la inclinación de las cámaras respecto al suelo.
- Robot: contiene la cinemática inversa del robot y accede a la clase de control Futaba.

- Futaba: esta clase permite controlar hasta ocho servos comunicándose a través del puerto serie con la tarjeta de control.

El programa se ejecuta en un único bucle de proceso que comienza y acaba con la adquisición de las imágenes. En la implementación se ha utilizado la librería de clases Qt que permite la comunicación directa entre objetos mediante señales (signals y slots) [3].

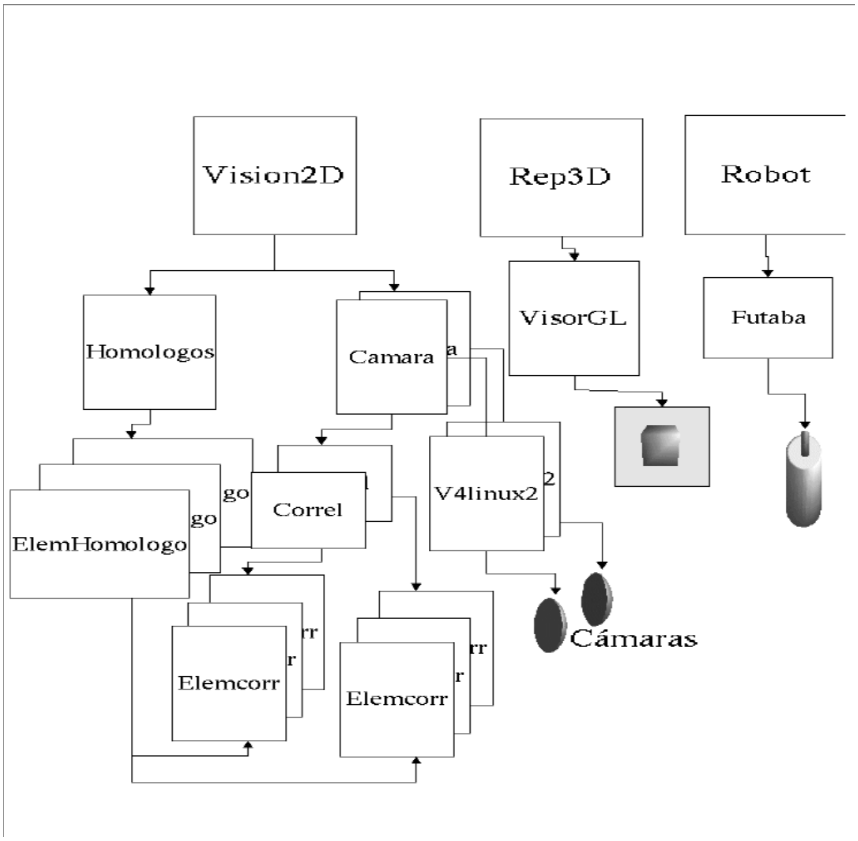


Figura 2. Estructura de clases

4 Técnicas de reconstrucción 3D

Cualquier sistema estéreo debe resolver dos problemas. El primero, conocido como correspondencia, consiste en determinar qué punto de la imagen derecha se corresponde con un punto de la imagen izquierda. El segundo problema es la reconstrucción, es decir, a partir de pares de puntos de las dos imágenes determinar la estructura 3D de la escena visualizada [7]. Para resolver estos dos problemas, el sistema propuesto lleva a cabo las siguientes fases:

1. Detección de esquinas: encontrar puntos de alto contraste en la imagen.
2. Calibración de las cámaras: determinar los parámetros intrínsecos y extrínsecos de cada cámara del sistema estéreo.

3. Corrección de imágenes: eliminar las distorsiones introducidas por las cámaras.
4. Cálculo de pares de homólogos: correspondencia entre puntos de las dos imágenes del sistema.
5. Recuperación de las coordenadas 3D de los pares homólogos.
6. Representación 3D de la escena.

4.1 Detección de esquinas

La búsqueda de puntos homólogos en las dos imágenes puede simplificarse considerablemente si, previamente, se extrae un conjunto de puntos cuyo entorno sea los más variado posible. De esta forma la comparación entre imágenes de estos puntos con una función de correlación obtendrá resultados mucho más precisos. A estos puntos se los suele denominar esquinas debido a que, muchas veces, coinciden con las esquinas de los objetos de la escena. Para la detección de esquinas utilizamos el filtro de Harris [7], cuyo funcionamiento se puede resumir en los siguientes pasos:

1. Suavizado de las dos imágenes mediante una convolución con una máscara gausiana de tamaño 9x9 y varianza 1.5.
2. Cálculo de la matriz de autocorrelación (2x2) del entorno 9x9 de cada pixel.
3. Obtención de los 2 autovalores de esta matriz.
4. Selección de aquellos pixels cuyo menor autovalor asociado es superior a un cierto umbral predefinido. Si hacemos que el resultado de esta umbralización sea una imagen en la que cada pixel valga 0, si no superó el umbral, o el valor del menor autovalor, si lo hizo, podemos pasar al último paso del proceso:
5. Supresión de puntos que son máximos locales mediante una búsqueda en el entorno local de cada posición. El resultado de este último paso es una imagen binarizada donde un valor distinto de cero indica la presencia de una esquina.

A continuación pasamos a la instanciación de objetos seguidores de esquinas según el esquema de clases descrito en el apartado anterior. La idea es tratar las esquinas como entidades que persisten en la secuencia de imágenes y asociar a cada una de ellas una instancia de la clase. Esto nos permite seguir la posición y velocidad de estas esquinas en el tiempo mediante un filtro de Kalman [1]. Además, esta estructura permite almacenar toda la información referente a cada esquina sin esfuerzo añadido, como las coordenadas normalizadas y rectificadas. Finalmente, y aunque no se utiliza aquí, al almacenar la trayectoria en el tiempo de una esquina, es posible asignarle un valor de fiabilidad y ordenarlas según este criterio para poder seleccionar subconjuntos más robustos en procesos de autocalibrado.

4.2 Calibración de las cámaras

Aunque existen técnicas que permiten inferir información 3D de una escena sin necesidad de conocer los parámetros de las cámaras, llevar a cabo una fase previa de calibración en un sistema de visión artificial, nos abre la posibilidad de utilizar

un gran número de métodos de reconstrucción 3D y de reconocimiento. Asimismo, los modelos ópticos de los dispositivos de visualización reales obligan a tener en cuenta las distorsiones producidas por los defectos de la cámara. De esta forma, antes de realizar la reconstrucción de una escena es necesario llevar a cabo una fase de corrección de las imágenes para eliminar las distorsiones introducidas por las cámaras. Afortunadamente, dichas distorsiones pueden ser modeladas como simples distorsiones radiales, de acuerdo a las siguientes relaciones:

$$x_d = x_n(1 + k_1 r^2 + k_2 r^4) \tag{1}$$

$$y_d = y_n(1 + k_1 r^2 + k_2 r^4) \tag{2}$$

donde $(x_n,\, y_n)$ son las coordenadas normalizadas de la proyección de un punto en el plano de la imagen, $(x_d,\, y_d)$ las coordenadas del punto distorsionado y $r^2 = x_n^2 + y_n^2$. En situaciones reales, k_2 es normalmente mucho menor que k_1, por lo que el único parámetro de distorsión que debe ser estimado es k_1. Teniendo en cuenta este modelo, las coordenadas del pixel resultante de la proyección de un punto en el plano de la imagen se obtienen como:

$$\begin{bmatrix} x\ y\ 1 \end{bmatrix}^T = KK \begin{bmatrix} x_d\, y_d\, 1 \end{bmatrix}^T \tag{3}$$

donde KK es la matriz formada por los parámetros intrínsecos de la cámara.

El sistema propuesto utiliza un procedimiento de calibración explícito. Es decir, realiza una estimación de los parámetros a partir de la observación de una escena en la que las coordenadas 3D de los puntos son conocidas. De este modo, el método de calibración utilizado resuelve primeramente la transformación lineal de la ecuación anterior ignorando las componentes de distorsión y, posteriormente, optimiza la estimación, incluyendo el coeficiente de distorsión radial (k_1), mediante técnicas no lineales. Para resolver este último problema el método utilizado ha sido el de Levenberg-Marquardt, ya que proporciona una convergencia rápida.

4.3 Corrección de las imágenes

Una vez estimados los parámetros de las cámaras, es necesario corregir las imágenes para eliminar las distorsiones introducidas. Para acelerar el proceso, la corrección no se realiza sobre toda la imagen, sino sólo sobre aquellos puntos que serán utilizados en la fase de reconstrucción de la escena.

Dado un determinado punto de la imagen, con coordenadas $(x,\, y)$, estamos interesados en calcular las coordenadas corregidas $(x_c,\, y_c)$ que vienen dadas por:

$$(x_c,\, y_c) = (x_n,\, y_n) \tag{4}$$

donde

$$x_n = x_d/(1 + k_1 r^2) \tag{5}$$

$$y_n = y_d/(1 + k_1 r^2) \tag{6}$$

y

$$\begin{bmatrix} x_d \; y_d \; 1 \end{bmatrix}^T = K K^{-1} \begin{bmatrix} x \; y \; 1 \end{bmatrix}^T \tag{7}$$

Puesto que $r^2 = x_n^2 + y_n^2$, de las ecuaciones (5) y (6) tenemos que

$$r^2 = (x_d^2 + y_d^2)/(1 + k_1 r^2)^2 \tag{8}$$

Si denotamos $x_d^2 + y_d^2$ como r_d^2, llegamos a la siguiente ecuación:

$$k_1 r^3 + r = r_d \tag{9}$$

Por lo tanto, el cálculo de las coordenadas corregidas $(x_c,\ y_c)$ se reduce a resolver el polinomio de grado 3 de la ecuación (9).

4.4 Cálculo de puntos homólogos

El rendimiento de cualquier método de correspondencia puede verse empeorado por la existencia de puntos que sólo son visibles desde una de las cámaras y por la asociación de falsos pares de puntos debido al ruido o a la falta de iluminación. Los efectos de estos fenómenos pueden reducirse si se tiene en cuenta la geometría epipolar del sistema estéreo para realizar la búsqueda de pares de homólogos.

Dado un par estéreo de cámaras, cualquier punto P del espacio 3D define un plano a través de los centros de proyección de las dos cámaras. Dicho plano (plano epipolar) corta los dos planos de imagen en dos líneas epipolares. De este modo, si para cada punto p_I de la imagen izquierda se determina la correspondiente línea epipolar en la imagen derecha, se puede restringir la búsqueda de los homólogos de p_I a aquellos puntos que estén en su línea epipolar. El problema de la correspondencia se simplifica aún más si rectificamos cada imagen de forma que los pares conjugados de líneas epipolares sean colineales y paralelas al eje horizontal de la imagen. Así, el conjunto de posibles homólogos de un punto $(x_I,\ y_I)$ de la imagen izquierda estará formado por aquellos puntos de la imagen derecha cuya coordenada y sea igual a y_I.

La rectificación se lleva a cabo mediante una rotación de las cámaras alrededor de los centros ópticos que consigue transformar las líneas epipolares en rectas paralelas al vector de traslación T del sistema. Una posible formación de dicha matriz de rectificación (R_{rect}) es la siguiente:

$$R_{rect} = \begin{pmatrix} e_1^T \\ e_2^T \\ e_3^T \end{pmatrix} \tag{10}$$

donde e_1, e_2 y e_3 son vectores definidos de la siguiente forma:

$$e_1 = \frac{T}{\|T\|} \tag{11}$$

$$e_2 = \frac{\left[-T_y\ T_x\ 0\right]^T}{\sqrt{T_x^2 + T_y^2}} \tag{12}$$

$$e_3 = e_1 \times e_2 \tag{13}$$

Esta matriz de rotación permite rectificar $(x_c,\ y_c,\ 1)$ de la siguiente forma:

$$\left[x'\ y'\ z'\right]^T = R_{rect}\left[x_c\ y_c\ 1\right]^T \tag{14}$$

A partir de la ecuación anterior, las coordenadas rectificadas del punto $(x_r,\ y_r,\ 1)$ vienen dadas por:

$$(x_r,\ y_r,\ 1) = (x',\ y',\ z')/z' \tag{15}$$

Para los puntos de la imagen derecha hay que construir la matriz de rectificación (Rd_{rect}) teniendo en cuenta la matriz de rotación R del sistema. De esta forma, Rd_{rect} se expresaría como:

$$Rd_{rect} = R_{rect}R^T \tag{16}$$

La rectificación de estos puntos se lleva a cabo de la misma forma que para los puntos de la imagen izquierda (ecuaciones 14 y 15) utilizando la matriz de rotación Rd_{rect}.

Una vez rectificados los puntos de las dos imágenes, se crean los conjuntos de posibles homólogos para cada punto p_I de la imagen izquierda. Así, para un punto de la imagen izquierda con coordenadas rectificadas $(x_{Ir},\ y_{Ir})$, su conjunto de homólogos H_I estará formado por aquellos puntos de la imagen derecha $(x_{Dr},\ y_{Dr})$ que verifiquen $y_{Dr} \approx y_{Ir}$. Si el conjunto de homólogos H_I no es vacío, el punto homólogo de p_I será aquel que maximice el criterio de similitud para una ventana de la imagen definida alrededor del punto. La similitud entre p_I y cada punto p_D de H_I se mide como la correlación[9] entre los puntos de imagen contenidos en las ventanas de ambos puntos:

$$NCC(p_I, p_D) = \frac{\sum_{i=-b}^{b}\sum_{j=-b}^{b}[I_I(p_{Ix}+i, p_{Iy}+j) - \overline{I_I}][I_D(p_{Dx}+i, p_{Dy}+j) - \overline{I}_D]}{\sqrt{\sum_{i=-b}^{b}\sum_{j=-b}^{b}[I_I(p_{Ix}+i, p_{Iy}+j) - \overline{I}_I]^2 \sum_{i=-b}^{b}\sum_{j=-b}^{b}[I_D(p_{Dx}+i, p_{Dy}+j) - \overline{I}_D]^2}} \tag{17}$$

donde I_I e I_D son las imágenes izquierda y derecha, respectivamente, $\overline{I_I}$ e $\overline{I_D}$ los valores medios de los pixels de las dos imágenes, $(p_{Ix},\ p_{Iy})$ las coordenadas originales de p_I, $(p_{Dx},\ p_{Dy})$ las coordenadas originales de p_D y $2b+1$ el ancho en pixels de la ventana de correlación.

Para regiones de puntos similares, el coeficiente de correlación de la ecuación (17) dará un valor cercano a 1, por lo que se considera que p_D es homólogo de p_I si maximiza dicho coeficiente en relación con los restantes puntos de H_I, pero además verifica $NCC\,(p_I,p_D) \approx 1$.

Podemos resumir el procedimiento de cálculo de pares de homólogos en los siguiente pasos:

1. Rectificación de puntos: transformación de las coordenadas corregidas $(x_c,\, y_c)$ de un punto de imagen a coordenadas rectificadas $(x_r,\, y_r)$
2. Calcular el conjunto de posibles homólogos para cada punto de la imagen izquierda: dado un punto p_I con coordenadas rectificadas $(x_{lr},\, y_{lr})$, construir el conjunto de homólogos H_I de dicho punto con los puntos p_D de la imagen derecha que verifiquen $y_{Dr} \approx y_{Ir}$.
3. Determinar los pares de puntos homólogos:

– Para cada punto p_I de la imagen izquierda, obtener el punto $p_D \in H_I$ que maximiza $NCC\,(p_I,p_D)$.
– Insertar en la lista de homólogos el par $p_I - p_D$ si se verifica $NCC\,(p_I,p_D) \approx 1$.

4.5 Reconstrucción 3D

Dado que los parámetros intrínsecos y extrínsecos del sistema estéreo son conocidos, es posible llevar a cabo la reconstrucción 3D de la escena utilizando un método de triangulación. Fijando el eje de referencia del sistema en el centro de proyección de la cámara izquierda, podemos considerar para cada par de puntos homólogos p_I-p_D, las rectas definidas por O_I-p_I y O_D-p_D, donde O_I y O_D son los centros de proyección de las cámaras izquierda y derecha, respectivamente. La intersección de ambas rectas proporciona las coordenadas 3D de un punto P de la escena. Puesto que los parámetros del sistema sólo son conocidos de manera aproximada, dicha intersección no es exacta y hay que calcular P como el punto más cercano a las dos rectas [2][7].

Podemos definir las rectas O_I-p_I y O_D-p_D de la siguiente forma:

$$O_I - p_I \Rightarrow ap_I \tag{18}$$

$$O_D - p_D \Rightarrow bR^T p_D + T \tag{19}$$

donde R y T son la matriz de rotación y el vector de traslación del sistema y p_I y p_D están expresados en coordenadas corregidas ($p_I = (x_{Ic},\, y_{Ic},\, 1)$; $p_D = (x_{Dc},\, y_{Dc},\, 1)$). Si definimos un vector w (ecuación 20) perpendicular a las dos rectas, el punto más cercano a O_I-p_I y a O_D-p_D es el punto medio de un segmento paralelo a w que une ambas rectas.

$$w = p_I \times R^T p_D \tag{20}$$

En definitiva, el problema se reduce a resolver el siguiente sistema lineal de ecuaciones:

$$ap_I + cw = bR^T p_D + T \tag{21}$$

Este sistema proporciona los puntos de unión del segmento con las dos rectas, que vienen dados por ap_I y $bR^T p_D + T$. Así, el punto P puede determinarse como:

$$P = ap_I + cw/2 \tag{22}$$

El resultado de aplicar este proceso a cada par de homólogos obtenidos de la fase anterior es un conjunto de puntos 3D, que puede ser utilizado para inferir las características de la escena visualizada y llevar a cabo las acciones correspondientes.

4.6 Triangulación

Un conjunto de esquinas en las dos imágenes no es suficiente para que la reconstrucción del espacio sea útil al robot. Por otro lado, obtener una reconstrucción densa partiendo de puntos requiere un postproceso de las imágenes extremadamente costoso, por ejemplo utilizando Programación Dinámica. La alternativa que estamos explorando se basa en hacer un conjunto de hipótesis sobre la escena que nos proporcionen una estructura completa del espacio 3D. Con este punto de partida, podemos utilizar el movimiento del robot y la reproyección 3D a 2D para refinar esta representación.

La técnica utilizada consiste en hacer una triangulación de Dealunay sobre las esquinas de una de las cámaras que han sido emparajedas con sus homólogas de la otra cámara. Para ello utilizamos el software[4][5]. El resultado de este proceso es una malla de triángulos 2D que cubren la imagen. La suposición que se hace es sobre la existencia real de los lados de los triangulos como contornos visibles de la escena. Muchos de ellos no lo serán y, por tanto, la representación 3D diferirá de la escena real. Lo que resta es convertir estos triángulos a 3D y comenzar a refinar la representación obtenida. Para la conversión no hay que hacer nada más, puesto que los vértices son el conjunto de homólogos y, para ellos ya se han calculado las coordenadas 3D. El ajuste y validación de la representación se discute a continuación.

5 Representación del espacio

Murphy representa la escena visualizada como una malla de triángulos en el espacio, esto es, como una superficie continua aproximada por planos. Esta representación tiene varias ventajas, frente a otras técnicas como NURBS, para el esquema que planteamos. Por ejemplo, es inmediato calcular homografías planares, reproyectar puntos o regiones en las cámaras y asignarles una textura directamente desde la imagen. Además, su visualización es muy eficiente y es fácil de refinar, añadiendo o eliminando triángulos.

Aunque, de momento, el sistema se limita a crear y mantener esta representación, el objetivo inmediato es que este módulo genere movimientos en el robot y reproyecciones sobre las imágenes para validar y ajustar la representación. La idea es partir de la hipótesis de que cada plano representado existe en el mundo y utilizar acciones que validen o refuten esta hipótesis. Las opciones que estamos explorando se basan en comparar las texturas que se llevan desde las imágenes a los triángulos, con la reproyección de estos sobre las mismas imágenes. Por el lado del movimiento, la validación consiste en hacer converger las cámaras hacia puntos significativos de la reconstrucción 3D y comprobar lo que aparece en los centros de las imágenes, después del movimiento.

Cada rectificación de la existencia o tamaño de los triángulos se incorpora a la representación como un refinamiento sobre la anterior, imponiendo las restricciones necesarias. La principal dificultad de este proceso es que el refinamiento tiene que coexistir con los movimientos de traslación y orientación del robot, así como, con la posibilidad de que haya regiones de la escena con movimiento propio. La necesidad de que todas estas fuentes de cambios se integren en una representación consistente, da lugar a lo que denominamos representación activa. Frente a una versión pasiva de este concepto, las representaciones activas generan acciones en el robot y predicen internamente sus resultados en términos de percepción. Las acciones se seleccionan para mantener la representación dentro de un compromiso entre estabilidad e innovación. Por ejemplo, si la velocidad del robot y el efecto que ésta tiene en la percepción de la escena hacen que no sea posible mantener una representación al nivel de detalle necesario para un desplazamiento seguro, las acciones posibles son: disminuir la velocidad o bien, restringir el área de la imagen a una zona central.

6 Experimentos

En la imagen siguiente se muestra una captura de pantalla del software desarrollado. En el centro de la figura puede verse una ventana con las dos imágenes capturadas por las cámaras. Sobre ellas, las cruces muestran la posición de las esquinas detectadas.

A la izquierda aparece el panel de control de los motores del robot. En la parte derecha, se muestra la triangulación de Dealunay de las esquinas detectadas en la imagen correspondiente a la cámara izquierda. Los vértices de estos triángulos planos son las coordenadas 3D calculadas, que se representan en el visor OpenGL de la esquina superior izquierda. Finalmente, el panel de la parte superior derecha muestra la velocidad de captura y proceso.

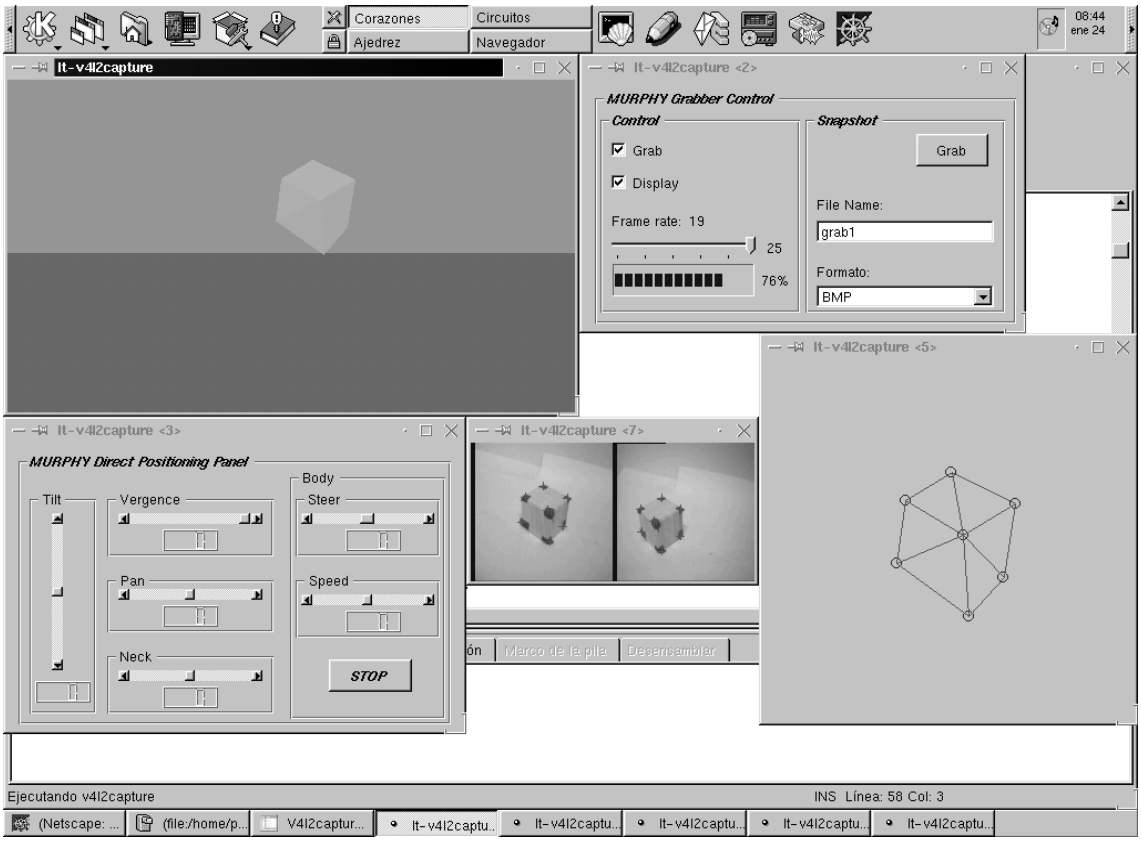


Figura 3. Escenario de trabajo del robot

En escenarios de mayor complejidad los lados de los triángulos planares no coinciden con las líneas de la escena. Esta situación hace que la calidad de la reconstrucción degenere rápidamente. Una mejora inmediata es incorporar un extractor de segmentos en la imagen y forzar a que la triangulación planar incluya todas las esquinas y líneas detectadas. Esta estrategia junto con el mecanismo de refinamiento incremental comentado en el apartado anterior, deben permitirnos obtener reconstrucciones de mayor calidad que puedan ser utilizadas por el robot en tareas de navegación complejas.

7 Conclusiones

Se ha presentado un robot móvil como base de un proyecto de investigación en curso, cuyo objetivo es crear Esta representación debe permitir al robot navegar y manipular de forma robusta y fiable. Los resultados presentados son los primeros pasos en esta dirección y muestran las posibilidad del enfoque planteado.

En resumen, las características del robot Murphy son las siguientes:

- Plataforma móvil con 6 gdl.
- Torreta estereoscópica.
- Procesamiento remoto en tiempo real utilizando una librería propia para 3DNow.

- Software para calibrado no lineal offline.
- Software orientado a objetos sobre Linux usando la librería Qt.
- Utilización de la geometría epipolar para la obtención de puntos homólogos.
- Triangulación Delaunay de las esquinas 2D utilizando el software triangle.c de J.R.ItemizeObtención de las coordenadas 3D de los triángulos.
- Representación planar 3D del espacio y visualización con OpenGL

Referencias

[1] Grewal M. S., Andrews A. P. Kalman Filtering. Theory and Practice, Pretince-Hall, 1993.
[2] Hartley R., Sturm P. Triangulation. Proceedings ARPA IUW pp. 957-966, 1994
[3] Qt Library `http://www.trolltech.com`
[4] Shewchuk J.R. Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. First Workshop on Applied Computational Geometry (Philadelphia, Pennsylvania), pp. 124-133, Association for Computing Machinery, May 1996. `http://www.cs.cmu.edu/ quake-papers/triangle.ps`
[5] Shewchuk J.R. Software triangle.c `http://www.cs.cmu.edu/ quake/triangle.html`
[6] Torr P. H. S., Beardsley P. A., Murray D. W. Robust Vision, British Machine Vision Conference, 1994, pp. 145,155, 1994
[7] Trucco E., Verri A. Introductory Techniques for 3D Computer Vision. Prentice-Hall, 1998
[8] Vicente J. A 3DNow Library for Image Processing. En preparación. 2001
[9] Vicente J., Guinea D. Differential Image Movement for 3D Reconstruction, SPIE Proceedings 2000
[10] Zissermann A., Hartley R. Multiview Geometry. Cambridge University Press. 2000
[11] Zeki S. A Vision of the Brain, Blackwell Scientific 1993.

Multiple Object Detection and Tracking in a Non-constrained Environment

Antonio Sanz[2], Iñigo Martín[1], Araceli Sánchez[1], and Enrique Cabello[2]

[1] Departamento de Informática y Automática
Universidad de Salamanca
Plaza de la Merced s/n
37008 Salamanca
[2] Escuela Superior de Ciencias Experimentales e Ingeniería
Universidad Rey Juan Carlos
C/Tulipán s/n
28933 Móstoles (Madrid)

Resumen This paper presents preliminary results in multiple object detection and tracking. Our approach has been tested in unconstrained, outdoor scenes. Research is focused in a robust method capable of detect multiple objects but with simple geometric shape (cars or robots).

1 Introduction

Computer vision is a useful tool that could be applied in several research fields, for example in robotics and video surveillance. In both cases we have to face similar problems in the detection and tracking of several objects (robots or cars) moving in a scene. Techniques needed in computer vision to detect a car or a robot could be very similar or even the same (depending of the geometry or movement of the objects). Even more, in both cases similar assumptions could be done, for example: a ground plane could be defined, because cars and robots move in a bidimensional plane. A camera looking the scene in an overview position could detect and track the objects. In mobile robot case, this information could be useful to allow that all robots could perform a common task (each robot will know the position and speed of the others). In video surveillance for example, the information could be processed in a control centre to monitor traffic.

In recent years, mobile and co-operative robots has decreased its price and increased its processing power. Till now, co-operative robots has develop its work mainly in controlled environments. Now research is moving to work in less controlled environments. So, it is expected that in few years, they could perform this task in open environments. In this scenario, a surveillance camera will perform the same task in case of cars.

This article presents initial results in an on-going project. This project is focused in the development and set up of computer vision techniques for a multiple object detection and tracking. Our work starts with one overview camera in an uncontrolled environment. In our case this is a traffic camera looking to a highway. Changes in illumination, small camera displacement, overlapping objects and object shadow

are allowed. One additional constraint is that colour could not be used to segment scenes, i.e., background does not have uniform colour or very different colour from objects.

2 State of the Art

A review of the literature on motion detection and multiple object tracking over video image processing shows that depending on the kind of scene we could find document analysis [6], medical images, images of people [2,5,11,13] or mobile robotics and traffic [1,3,6,7,8,9,12,14]. Most problems can be solved assuming a well-known information of the concrete scene. This "closed-world"is often used to reduce the redundant information and so, increasing the speed of the algorithms, but these methods are highly tied to the studied environment.

Such "closed-world"assumptions are proposed for tracking players in football [2] or kids in a known room [13], or even vehicles in traffic scenes [14] where the ground-plane constraint reduces the problem of localisation and recognition as much as we all know that road vehicles stand on the ground-plane.

Attending to other classification, several processing techniques can be found. Some papers presents a pre-processing step to avoid noise (by median filters [1], time convolution filters,...), others avoid this step [3,5]. Once this is done the segmentation part is taken into account. Edge and local feature detection [5], extraction of boundaries, lines and curves [9], morphological processes such as dilation and erosion [1], background subtraction [3,13,15], other motion detection techniques (gradient based [1,5], optical flow [4],...) and a-priori scene knowledge: ground plane [14], white lines in a road [9,12], basic rules and painted marks on a football field [2],... are most valuable methods.

Template matching is often used when the objects have a well-known shape. T.N. Tan et al [14] reduce the number of degrees of freedom using ground plane constraint and then, applying template matching to identify the best pose.

As it is shown in [2], there are some tracking techniques as correlation tracking where a template is extracted from the first image and is compared with the objects in next frames. The main problem is that it is very sensitive to occlusion, light conditions and changing size objects.

Motion tracking, specially optical flow methods, are also quite sensitive to occlusion because of its dependence on smooth changes in brightness that is not possible in the case of sudden occlusion.

In order to avoid the size changing restriction of the correlation tracking, deformable template matching permits small, constrained, changes in the template over time. It is usually used when the template matches an object, which is moving along a perspective trajectory, and so it has a geometric distortion between consecutive frames.

Kalman Filter is another common tracking method used by authors [3,8,11,12]. As [18] indicates "this filter is a set of mathematical equations that provides an efficient computational (recursive) solution of the least squares method".

3 Technical set-up

The experiment presented in this paper is an initial stage in a computer vision research. In this case only one camera is analysed. This camera offers an overview of the scene, so, it is possible the detection and tracking of the different objects. As the scenes analysed in this paper are taken watching a traffic highway, only cars are present. Changes in illumination are allowed. Also, there are huge variations in speed, colour and size of the objects. The camera offers a lateral view of the highway, so occlusion is also present.

Later experiments will be developed with three cameras. One will be an overview camera, similar as the one present here. In the case of robotics application, two cameras will be placed in the robots. These cameras will be used to achieve more effective co-operation between robots developing the same task.

4 Algorithm Description

Motion detection algorithm operates with two frames applying optical flow to the three colour components RGB (Red-Green-Blue). We set a threshold to avoid noise detection, this threshold depends on the input frames. Those pixels which optical flow value is greater than threshold are represented in blue colour (fig.2) and we called them candidate pixel, everyone else is set to white. Once we have analysed the full image, only regions with enough pixels in blue are considered. These regions will be processed later. We reject, without loosing significant information, regions with few blue pixels due to illumination changes or to objects too small or too far.

Those extracted windows or regions of interest (ROI) contain the supposed moving object. If occlusion is present and two or more ROIs overlapped, we integrate all the ROIs into a single one as can be seen in Fig. 2.

Several classical edge detectors (Roberts, Prewitt and Sobel) have been applied and best one for our purposes was the 3x3 Sobel edge detector. This detector offers us good results in an acceptable time, so we have selected it. Edge detector is used as input to next step (Hough Transform).

For recognition of well defined linear objects, such as cars or vehicles in general, it is proposed the Hough Transform (using polar coordinates to avoid the difficulties presented in vertical line detection). It is a heavy process so we are studying a more efficient alternative method (neural nets, PCA,...).

5 Results

Our algorithm has been tested on a variety of image sequences with promising results. Images size is 384x288 pixels, colour is coded in RGB, and compression technique is AVI. Figures 2 to 7 show processing techniques applied to one-frame.

Fig. 2 shows the K and K+1 frames of an image sequence, motion is detected using optical flow. Fig. 3 shows image after optical flow process. It could be noted the overlapping ROIs due to occlusion among different vehicles.

Next image is same as Fig. 2 except windows of interest are drawn to show motion detection. Extracted ROIs can be seen in Fig. 4, which are the input images to the Sobel Edge Detector (Fig. 4).

Thresholds used in this work depends on the average value of optical flow, and a manually value set for the size of the window in which movement will be considered.

These threshold values can be modified in order to get better results. For instance the optical flow tolerance could be adapted to noisy sequences, where many points have high values, or we can set a lower size to the minimum ROI if we expect small moving objects in scene.

6 Conclusions and Future Work

Since very preliminary results, a useful computer vision technique is presented in this paper. Extending the capabilities of robots to operate in non-constrained environments will result in a huge number of fields in which robots could operate. Computer vision detection and tracking of robots will increase co-operative tasks in which visual feedback is needed.

Promising results in video surveillance shows that car detection and tracking in non-constrained environments is a very interesting field of research.

In both cases, using computer vision sensors is possible to offer an "intelligent"video sensor. This video sensor could offer highly added value, because information could be processed distributively and information flow is decreased. So, information transmitted contains not only images, but also richer information about the objects in scene.

Our work will continue, improving identification and tracking of the objects (car or robots). Addition of more cameras will result in an effort of co-ordination between images but more information could be extracted and presented.

7 Acknowledgements

This paper shows preliminary results for a project granted by the Spanish Traffic Directorate (DGT: Dirección General de Tráfico) and the Universidad de Salamanca, its support is kindly appreciate. Also, authors wish to thank the help of Antonio Guzmán and Laura Agudo.

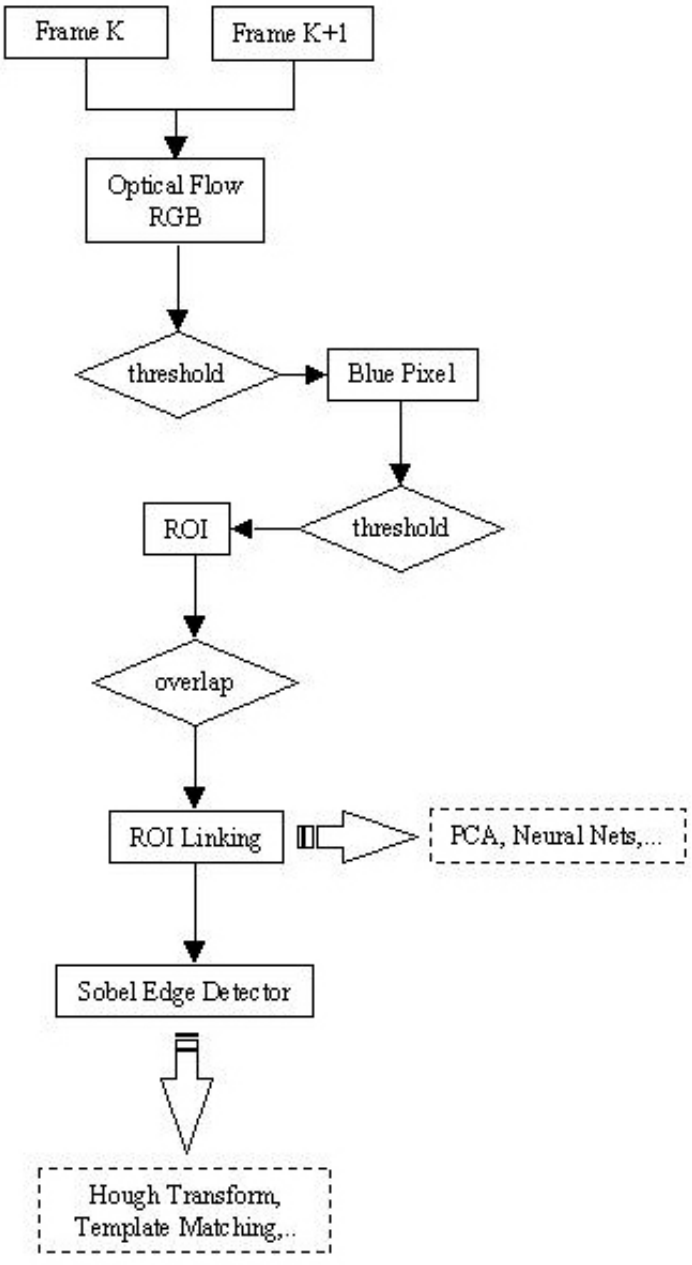


Figura1. Diagram of the algorithm.

Figura2. Original images in time K and K+1

Figura3. Optical Flow results and ROIs superposed in Frame K+1

Figura4. Extracted ROIs and Sobel Edge Detector on ROIs

Referencias

[1] [1] D.J. Dailey, L. Li, "An Algorithm to Estimate Vehicle Speed Using Un-Calibrated Cameras"

[2] [2] Stephen S. Intille, "Tracking Using a Local Closed-World Asumption: Tracking in The Football Domain", M.I.T. Media Lab Perceptual Computing Group Technical Report No. 296, August, 1994.

[3] [3] Dieter Koller, Joseph Weber, and Jitendra Malik, "Robust Multiple Car Tracking with Occlusion Reasoning", In Proc. European Conf. Comp. Vis., volume 1, pages 189-196, Stockholm, Sweden, May 1994.

[4] [4] Berthold K.P. Horn and Brian G. Schunck, "Determining Optical Flow", Artificial Intelligence, volume 17, pages 185-204, May 1981.

[5] [5] Josep Amat, Alicia Casals and Manel Frigola, "A Specialized Vision System for Control by means if Gestures", Advanced Control Workshop, Coimbra, pages 678-683, 1998.

[6] [6] T. Steinhertz, E. Rivlin, and N. Intrator, "Offine Cursive Script Word Recognition-A Survey", Int. J. Document Analysis Recognition 2, pages 90-110, 1999.

[7] [7] J. Badenas, F. Pla, "Segmentation Based on Region-Tracking in Image Sequences for Traffic Monitoring", In 14th International Conference on Pattern Recognition, ICPR, 1998.

[8] [8] R. García, J.Batlle, Ll. Magí, Ll. Pacheco, "Seguimiento de Múltiples Objetos: Un Enfoque Predictivo", 1999.

[9] [9] J.M. Sanchiz, R. Pérez, M. Aguilella, "Interpretación de Escenas de Tráfico para Asistencia a la Conducción".

[10] [10] Natan Peterfreund, "The Velocity Snake: Deformable Contour for Tracking in Spatio-Velocity Space", Computer Vision And Image Understanding, volume 73, pages 346-356, March 1999.

[11] [11] S. Watcher and H. H. Nagel, "Tracking Persons in Monocular Image Sequences", Computer Vision And Image Understanding, volume 74, pages 174-192, June 1999.

[12] [12] Wlodzimierz Kasprzak and Heinrich Niemann, "Adaptative Road Recognition and Ego-state Tracking in the Presence of Obstacles", International Journal of Computer Vision, volume 28, pages 5-26, 1998.

[13] [13] Stephen S. Intille, James W. Davis, and Aaron F. Bobick, "Real-Time Closed-World Tracking"IEEE Conference on Computer Vision and Pattern Recognition, November 1996.

[14] [14] T. N. Tan, G. D. Sullivan, and K. D. Baker, "Model-Based Localisation and Recognition of Road Vehicles", International Journal of Computer Vision, volume 27, pages 5-25, 1998.

[15] [15] Yuri Ivanov, Aaron Bobick, and John Liu, "Fast Lighting Independent Background Subtraction"M.I.T. Media Laboratory Perceptual Computing Section Technical Report No. 437.

[16] [16]Pratt, William K, "Digital Image Processing 2nd Ed.", John Wiley and Sons. Inc, New York, 1991.

[17] [17] Azriel Rosenfeld, "Classifying the Literature Related to Computer Vision and Image Analysis", Computer Vision and Image Understanding 79, pages 308-323, 2000.

[18] [18] Greg Welch and Gary Bishop, "An Introduction to the Kalman Filter", UNC-Chappel Hill, TR 95-041, November 2000.

Detección probabilística de puertas con visión monocular activa

José María Cañas Plaza[1], Reid Simmons[2], and María C. García-Alegre[3]

[1] Escuela Superior de Ciencias Experimentales e Ingeniería
Universidad Rey Juan Carlos
[2] Computer Science Department
Carnegie Mellon University
[3] Instituto de Automática Industrial
Centro Superior de Investigaciones Científicas (CSIC)

Resumen Este artículo describe un sistema de percepción activa que integra visión con sonars para detectar la existencia y la posición de puertas en el entorno de un agente móvil. El flujo visual se analiza buscando bordes verticales y esa información en imágenes tomadas desde distintas posiciones se mezcla con los datos sonar en un marco probabilístico. La dinámica de probabilidades que suben y bajan dependiendo de los datos sensoriales acaba confirmando la ubicación real de la puerta, desechando las ficticias, que son debidas a la ambigüedad de profundidad inherente a la visión monocular. El sistema se ha implantado con éxito en un robot de interiores. También se equipa al agente con un control autónomo que gobierna los movimientos del robot para facilitarle activamente la percepción de las puertas. Este control elige los puntos desde los cuales tomar sucesivas observaciones del entorno de manera que se maximice la discriminación entre las puertas candidatas que se tienen en cada momento.

1 Introducción

La incorporación de la visión como sensor a los agentes móviles se ha visto frenada por el alto precio del equipo (cámaras y tarjetas digitalizadoras) y sobre todo por la complejidad inherente en la extracción de información útil desde el flujo de imágenes. Así se han preferido otros sensores más baratos o con una interpretación directa cara a la navegación, como el sonar o el láser.

Si bien han aparecido recientemente enfoques que generan comportamientos desde las imágenes directamente (por ejemplo desde flujo óptico), el uso tradicional de la visión ha sido el de resconstrucción 3D del espacio (pares estéreo). En nuestro caso se utiliza la visión para enriquecer la representación del entorno con un estímulo concreto, las puertas. Este estímulo es muy frecuente en entornos de oficina, laboratorios, etc, por lo que la técnica descrita puede ser de utilidad para muchos investigadores de robótica móvil en interiores.

La motivación inicial para abordar este trabajo nació de la necesidad de que un robot móvil pasara por puertas relativamente estrechas. El robot móvil es cilíndrico y está equipado con diferentes sensores (sonars, encoders y tactiles). Las estrategias de navegación basadas en sonars resultan insuficientes para atravesar puertas estrechas (con las anchas no hay problema) debido a la alta incertidumbre angular asociada a las medidas sonar. Esa falta de resolución lleva a percibir los huecos de una anchura

menor a la real, lo cual resulta crítico con puertas que ofrecen poco margen a parte de la anchura del robot. Para solventar esta situación se pensó incorporar visión al sistema perceptivo de modo que se pudiera detectar la anchura real de las puertas y con ello implementar más fiablemente el control para atravesarlas.

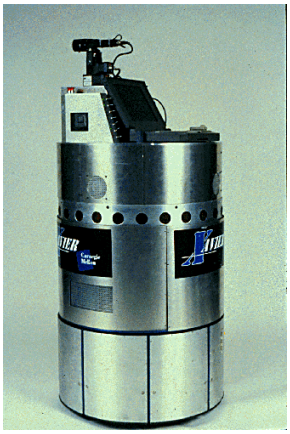


Figura1. Robot Xavier

Este sistema perceptivo detector de puertas también resulta ventajoso en otras dos situaciones. Primero, como ayuda a la autolocalización inicial del robot, las puertas detectadas en el entorno tras arrancar el robot le ayudan a anclar mejor cual es su posición en cierto mapa global que ya pudiera tener. Segundo, como ajuste fino para llegar a una habitación concreta después de que la navegación global ha llegado aproximadamente al destino y deja al robot en los alrededores de la puerta dando por concluída su tarea (véase [Simmons95]). Estos dos ejemplos son de utilidad directa en Xavier (figura 1), el robot para el cual se programó este sistema perceptivo.

El problema se ha descompuesto en varias partes, que coinciden con las siguientes secciones del artículo. En la primera se extraen las jambas de las puertas que hay en la imagen, siguiendo técnicas de filtrado de bordes y detección de líneas verticales. En una segunda etapa se describe el marco probabilístico que permite integrar información desde distintas imágenes y la fusión con otros sensores como los sonars y tactiles. Después se describe la estrategia que orienta el movimiento autónomo del robot y de la cámara en su búsqueda de las puertas, convirtiendo al sistema en un sistema de percepción activa. Finalmente se esbozan unas conclusiones y las líneas por las que puede avanzar el trabajo.

2 Detección de puertas en la imagen

Para percibir la puerta en profundidad primeramente hay que detectar sus jambas en cada imagen monocular tomada por la cámara. La cámara está montada paralela

al suelo y sólo se permiten movimientos horizontales. De este modo las jambas aparecen siempre como líneas verticales en las imágenes de la cámara. El procedimiento seguido para detectarlas sigue tres pasos :

1. Las imágenes de entrada vienen en niveles de gris, no se utiliza información de color. Para minimizar el ruido se les pasa inicialmente por un suavizado que elimina pixels espúreos que sólo perturbarían los cálculos.
2. Se aplica un filtro de bordes siguiendo la técnica y el código de Stephen Smith descrito en [Smith97]. Este filtro clasifica cada pixel de la imagen como punto borde o no.
3. Sobre la imagen de bordes se buscan líneas verticales largas, esto es, columnas que acumulen más de $umbral_1 = 80\%$ de puntos borde. Como la verticalidad estricta es difícil de conseguir en la práctica también se tienen en cuenta las $n = 4$ columnas adyacentes. Todas las columnas que superen ese umbral de puntos borde representan una línea vertical pronunciada, posiblemente la jamba de una puerta.

Este detector funciona en tiempo real sin suponer demasiada sobrecarga al procesador y con él los bordes de las puertas aparecen resaltados. También resalta los límites de armarios o los bordes verticales de pizarras que contrasten con el color de la pared. Sin embargo no hay problema con esos falsos positivos, está previsto desecharlos posteriormente. Lo importante en este primer paso es no quedarse ninguna jamba real sin detectar, puesto que lo que no se vea en esta etapa no existe de ningún modo para las restantes partes del sistema perceptivo, y por ello se pierde.

Para que esta técnica sea válida se necesita que haya contraste entre el marco y la pared, o entre el marco y la hoja de la puerta, que suele ser habitual. Como todos los sensores, tiene asociado un rango de alcance: sólo detecta jambas próximas puesto que las puertas lejanas ocupan pocos pixels y por ello son ignoradas. No obstante en entornos de oficina las distancias no suelen ser muy grandes.

La figura 2 muestra una imagen típica, de 320*240 pixels, obtenida con una cámara de 55° de apertura. En ella se pueden observar los puntos borde superpuestos a la imagen suavizada, y la salida final del detector de jambas, con tres columnas resaltadas, las dos jambas reales y un tercer borde vertical muy pronunciado.

3 Mezcla probabilística de información desde varias imágenes

El objetivo último del sistema descrito es percibir la existencia de puertas y ubicar perfectamente tanto su posición respecto del robot como su anchura real. Para extraer profundidad usando visión se necesita al menos un par de imágenes desde distintas posiciones para resolver la ambigüedad inherente a la visión monocular. Dicho de otro modo cada una de las imágenes sólo proporciona información parcial acerca de la puerta, información no concluyente. La idea de este trabajo es utilizar el marco probabilístico para integrar esa información de varias imágenes tomadas

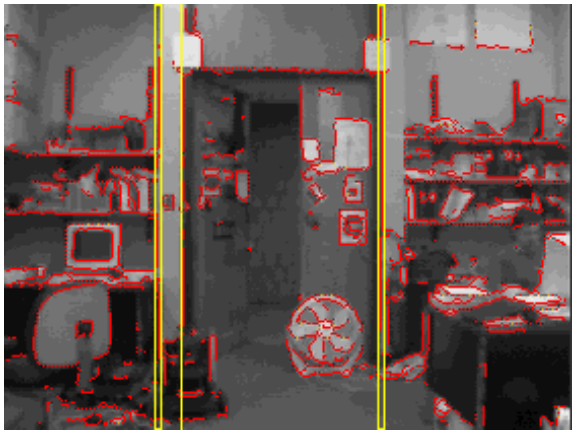


Figura2. Detección de la puerta en la imagen. Obsérvense los bordes resaltados y el falso positivo al lado de la jamba izquierda.

por el robot desde posiciones distintas, así como de los sensores ultrasónicos, con el objetivo de inferir la existencia y localización de las puertas (figura 3).

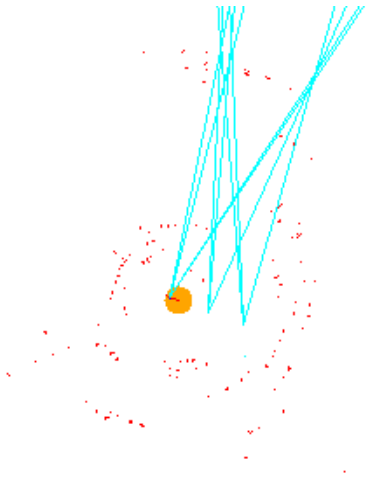


Figura3. Rayos visuales procedentes de imágenes obtenidas desde 3 posiciones distintas.

Para representar la creencia que tiene el robot sobre la existencia o no de puertas en su entorno, éste mantiene un **grid de probabilidad de jamba**. Este grid se va actualizando a medida que el robot recibe nueva información sensorial, bien desde la cámara, bien desde los sonars y es la base sobre la que se decide interpretar lo visto como puerta o no.

Es un grid bidimensional, basado en la representación propuesta por Elfes [Elfes90] y situado en un plano paralelo al suelo a la altura de los sonars ($\approx 1m$). Cada celda almacena la probabilidad de que en esa posición haya una jamba de puerta, y en los experimentos concretos se eligió un tamaño de $10cm * 10cm$. Esta representación horizontal resulta suficiente para representar las puertas, su orientación, posición y anchura. Las jambas reales son verticales y en este plano aparecen como puntos. Con un par de ellos se representa una puerta. Puesto que esta representación esta dirigida exclusivamente a la identificación del estímulo puerta no aparecen en ella objetos por debajo de la altura de los sonars como las papeleras.

Inicialmente todas las casillas del grid tienen valor 0.5, reflejando el desconocimiento total. La probabilidad de cada una de las casillas del grid se va actualizando siguiendo la *regla de Bayes* a medida que se incorpora información. En cada momento t la estimación que se tiene de la probabilidad de jamba en el punto (x, y), $p_{jamba}(x, y, t)$, se define como la probabilidad de jamba en ese punto condicionada a las observaciones que se han obtenido hasta ese momento. Así lo muestra la ecuación 1, donde $data(t-1)$ supone el conjunto de observaciones acumuladas hasta el instante $t-1$ e $imagen(t)$ la observación actual.

$$p_{jamba}(x, y, t) = p(jamba/imagen(t), data(t-1)...) \quad (1)$$

$$= \frac{p(jamba/data(t-1)) * p(imagen(t)/jamba, data(t-1))}{p(imagen(t)/data(t-1))} \quad (2)$$

Aplicando a (1) la regla de Bayes obtenemos (2). Si asumimos que las observaciones son *independientes en sentido markoviano* (3) y utilizamos nuevamente la regla de Bayes (4) llegamos a la expresión (5). La independencia markoviana supone que dado el hecho de la existencia de la jamba, la $imagen(t)$ no depende de las observaciones anteriores $data(t-1)$, según expresa (3). En general el mundo no es markoviano, pero en la práctica asumimos esa independencia entre observaciones incorporando al grid exclusivamente las imágenes tomadas desde posiciones relativamente separadas. Esta suposición facilita enormemente el tratamiento probabilístico de la información y posibilita el desarrollo teórico siguiente, paralelo al de Margaritis [Margaritis98].

$$p(imagen(t)/jamba, data(t-1)) = p(imagen(t)/jamba) \quad (3)$$

$$p(imagen(t)/jamba) = \frac{p(jamba/imagen(t)) * p(imagen(t))}{p(jamba)} \quad (4)$$

$$p_{jamba}(x,y,t) = \frac{p(jamba/data(t-1)) * p(jamba/imagen(t)) * p(imagen(t))}{p(jamba) * p(imagen(t)/data(t-1))} \quad (5)$$

Normalmente en lugar de manejar directamente la probabilidad de jamba p_{jamba} se maneja el *ratio de probabilidad*, definido como $\rho_{jamba} = p_{jamba}/p_{\overline{jamba}}$. Con esta simplificación se elimina la dependencia de $p(imagen(t))$ y de $p(imagen(t)/data(t-1))$ sin perder información, puesto que afectan por igual a p_{jamba} y a $p_{\overline{jamba}}$, según muestra (8).

$$\rho_{jamba} = p_{jamba}/(1-p_{jamba}) \quad (6)$$

$$p_{jamba} = \rho_{jamba}/(1+\rho_{jamba}) \quad (7)$$

$$p_{\overline{jamba}}(x,y,t) = \frac{p(\overline{jamba}/data(t-1)) * p(\overline{jamba}/imagen(t)) * p(imagen(t))}{p(\overline{jamba}) * p(imagen(t)/data(t-1))} \quad (8)$$

Sustituyendo (5) y (8) en (6) llegamos a (9), que posibilita un tratamiento incremental de la información. Con cada nueva observación la probabilidad acumulada se multiplica por un factor $\frac{p(jamba/imagen(t))}{1-p(jamba/imagen(t))}$ que es precisamente el valor dado por el modelo de sensor, y una constante $\frac{1-p_{jamba}}{p_{jamba}}$ determinada por la probabilidad a priori de jamba, sin ningún otro conocimiento. Trabajando con logaritmos los productos se convierten en rápidas sumas. Si en algún momento se quiere obtener la probabilidad basta con deshacer los logaritmos y aplicar (7).

$$\rho_{jamba}(x,y,t) = \frac{p(jamba/imagen(t))}{1-p(jamba/imagen(t))} * \frac{1-p_{jamba}}{p_{jamba}} * \rho_{jamba}(x,y,t-1) \quad (9)$$

Por lo tanto, si una imagen proporciona información sobre el estado de determinada celdilla (x,y), es el valor del modelo de sensor $p(jamba/imagen(t))$ en esa posición el que determina si allí la probabilidad de jamba sube o baja después de la nueva observación. La información proveniente de las lecturas sonar se trata del mismo modo, pero con otro modelo de sensor distinto. En el siguiente apartado se describen con detalle ambos modelos.

3.1 Modelo de visión

Cada vez que se recibe una imagen *independiente* se actualiza la probabilidad asociada a las celdillas del grid que caen dentro del campo visual. El campo visual viene determinado por un *alcance* y una *apertura* angular. Si en la imagen se observa un borde vertical pronunciado, es indicio de que hay realmente una jamba *en algún lugar* del plano perpendicular al suelo que proyecta en los pixels de ese borde. En el

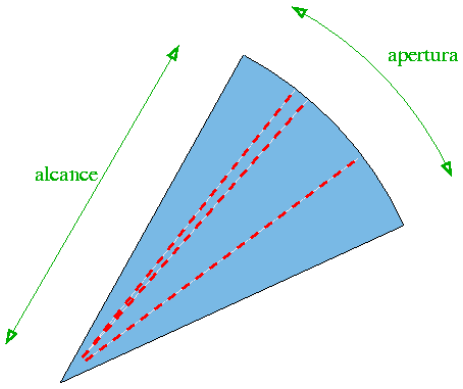


Figura4. Ejemplo de cómo actualiza una imagen el grid con probabilidad de jamba.

grid horizontal ese plano proyectante se convierte en una línea, y en cualquier punto de esa línea puede estar físicamente la jamba. La probabilidad de jamba asociada a esas celdillas debe aumentar.

Por el contrario si en la imagen no se aprecia ningún borde vertical pronunciado eso es síntoma de que en el campo de visión abarcado por la cámara no hay ninguna puerta y la estimación de probabilidad de jamba debe bajar.

La interpretación de las imágenes sigue el modelo de la figura 4, que corresponde a la proyección en profundidad de la figura 2. Siguiendo este modelo se aumenta la probabilidad de jamba en las celdillas sobre las líneas discontinuas porque todas ellas proyectan en pixels donde se aprecia una jamba en la imagen. También se baja la probabilidad en las celdillas oscuras que caen dentro del campo visual, porque en su proyección no se ha apreciado un borde vertical pronunciado. Empíricamente ajustamos los valores concretos a $p(jamba_{x,y}/imagen(t)) = 0.72$ si proyecta en una jamba de imagen o $p(jamba_{x,y}/imagen(t)) = 0.42$ si no.

En otras palabras, se aumenta la probabilidad de jamba en todas aquellas posiciones que podrían ser la causa de que se haya apreciado una jamba en la imagen y se disminuye en el resto. La idea subyacente es que después de unas cuantas observaciones sólo la posición real de la jamba recibirá la realimentación positiva de la mayoría de las imágenes. El resto de localizaciones se verán desmentidas con alguna observación y su probabilidad de jamba no podrá ser tan alta como en la posición auténtica. Los valores concretos del modelo suponen un compromiso entre el peso de cada observación individual y la tolerancia a algún error en la percepción.

3.2 Modelo de sonars

Se puede suponer que un sonar mide la distancia al obstáculo más cercano en la dirección perpendicular al sensor. Por ejemplo si un sonar detecta el primer obstáculo a 3 metros eso quiere decir que entre el sensor y esos 3 metros hay espacio vacío y desde el punto de vista de la puerta, difícilmente hay una jamba en ese área.

El modelo de sonar utilizado consiste en un cono que se abre desde la posición del sensor hasta el radio medido, con el eje en la perpendicular del sensor y siguiendo una apertura que refleja precisamente la de la onda ultrasónica. Todas las celdas del grid que caigan dentro de ese cono disminuyen su probabilidad de ser jamba, puesto que el sonar ha sensado espacio vacío. De este modo los sonars se utilizan para descartar hipotéticas localizaciones de las jambas.

3.3 Probabilidad de puerta

El grid de probabilidad de jamba tiene la información sensorial necesaria para determinar si el robot ha visto una jamba real o no. En nuestro sistema una puerta viene definida por un par sus extremos, $punto1$ y $punto2$, que son precisamente sus jambas. A cada posible puerta le asociamos una **probabilidad de puerta** dada por la fórmula:

$$p_{puerta}(punto1, punto2, t) = p_{jamba}(x1, y1, t) * p_{jamba}(x2, y2, t) * condiciones\_de\_puerta \tag{10}$$

Es decir el producto de la probabilidad de cada una de sus jambas por un factor que indica en qué medida esa posible puerta satisface las condiciones de puerta. La condición principal evalúa si su anchura casa con la esperada en las puertas del entorno. En la implementación realizada ese factor es binario: 1 si la distancia entre jambas caen entre un mínimo y un máximo parametrizables, y 0 fuera de ese intervalo. También se anula si las dos jambas están contenidas en el mismo rayo óptico, ya que en ese caso no puede tratarse de una puerta.

Para examinar la existencia o no de puertas primero se explora el grid de probabilidad de jamba, y sólamente teniendo en cuenta aquellas celdillas más probables, se elabora un **conjunto de puertas candidatas**, formado por todos los posibles pares. Seguidamente se evalúa la probabilidad de cada una de estas puertas candidatas y el robot asume que ha detectado realmente una puerta cuando su probabilidad está por encima de cierto *umbral de aceptación*. Es decir cuando sus jambas están confirmadas por las observaciones sensoriales y la distancia entre ellas es la adecuada.

3.4 Experimentación

El esquema probabilístico propuesto para integrar información desde varias imágenes y sensores sonar se ha implementado con éxito en el robot real Xavier (figura 1) de la Universidad Carnegie Mellon operando en una planta llena de despachos y laboratorios.

En la figura 5 se puede observar el estado del grid de probabilidad de jamba después de la observación del entorno desde cinco posiciones distintas, separadas entre sí unos 30cm. Alrededor del robot la probabilidad de jamba es muy baja (color oscuro), gracias a los sonars que han ayudado a descartar las hipótesis en ese área. En la parte superior se aprecian tres haces de alta probabilidad correspondientes

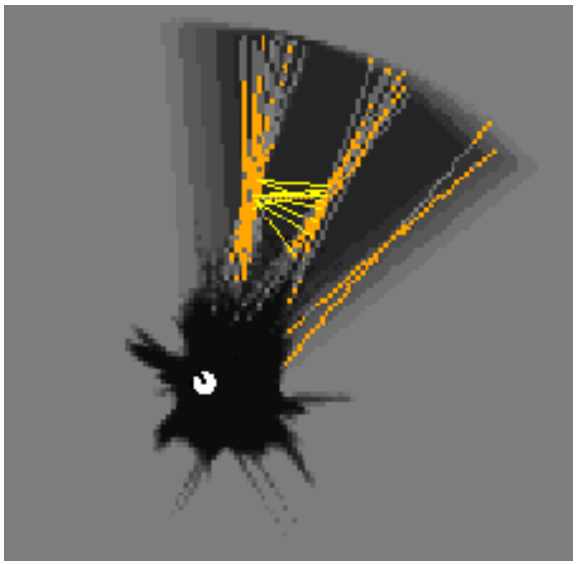


Figura5. Mezcla en un grid probabilístico de información obtenida desde varias posiciones. Obsérvese el espacio entre rayos con probabilidad menor que 0.5, si no se ha visto jamba se disminuye su probabilidad.

a los distintos rayos visuales de los bordes verticales detectados en las sucesivas imágenes. Las zonas oscuras intermedias corresponden a la proyección de la zonas de imagen con ausencia de bordes verticales.

Las partes intermedias de los dos haces centrales son las que reciben más probabilidad porque es donde intersectan los "rayos de los haces". En color claro, más o menos horizontales entre esos dos haces aparece superpuesto el conjunto de puertas candidatas obtenido hasta ese momento. Obsérvese que no hay ninguna puerta candidata con un extremo en el haz de rayos de la derecha, porque no han acumulado suficiente probabilidad. La longitud de todas las puertas candidatas generadas está dentro de los márgenes aceptables.

En estas pruebas el movimiento del robot y la orientación de la cámara venían teleoperados por el humano. Por lo tanto no se evaluaba la autonomía del agente percibiendo puertas, sino que la aproximación descrita fuera capaz de concluir la existencia y ubicación de la puerta desde unas observaciones razonables obtenidas desde posiciones distintas. En la siguiente sección describimos el enfoque seguido para dotar de autonomía agente y eliminar la participación humana en todo el proceso.

4 Percepción activa: movimiento para percibir la puerta

Debido al campo visual limitado, cobra importancia el lugar desde donde se observa el entorno. Dependiendo de ese lugar se percibirán unos estímulos y se ignorarán otros. En cierto modo se puede interpretar la propia posición del robot y la orien-

tación de su cámara como el foco sobre el cual tiene puesta su atención sensorial. Ese foco se deberá desplazar a lo largo del espacio para percibir todo el entorno y encontrar las puertas que en él pueda haber.

Las partes descritas hasta el momento son pasivas, en el sentido de que procesan lo que caiga delante de la cámara. Ahora describiremos la estrategia que controla los motores y guía su movimiento para facilitar al agente la percepción de la puerta. Con ella se completa el sistema de percepción *activa* que dota al robot de la capacidad de detectar las puertas de modo completamente autónomo, sin intervención humana alguna.

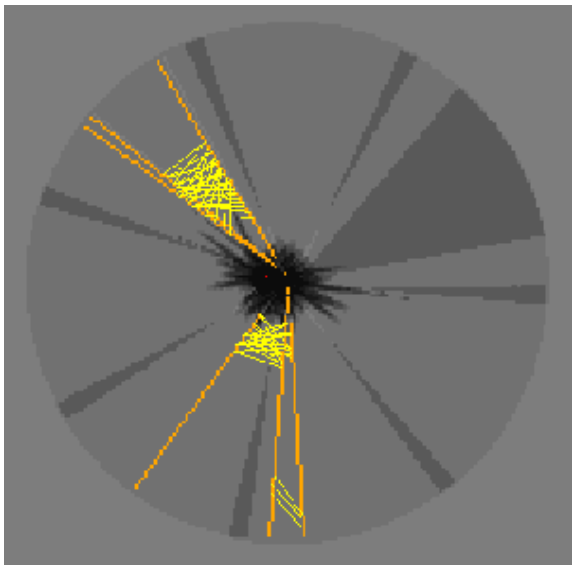


Figura6. Aspecto del grid después de la vuelta inicial, hay un conjunto de puertas candidatas. Entre ellas hay que discernir la puerta real con nuevas observaciones.

Inicialmente el robot da una vuelta completa sobre sí mismo mirando en todas direcciones, buscando todas las posibles jambas que pueda haber en sus cercanías. Después de ese primer vistazo el grid de probabilidad de jamba tiene un aspecto como el mostrado en la figura 6, con algunas jambas candidatas.

A partir de ese momento la estrategia de movimiento autónomo es iterativa: se calcula el punto óptimo para tomar la siguiente imagen y se desplaza el robot hasta él mediante navegación local. Una vez allí se vuelve a analizar la imagen de la cámara y se concluye si se ha detectado alguna puerta, si hace falta seguir explorando o si es mejor abandonar la búsqueda. En caso de necesitar más evidencia sensorial se calcula el nuevo punto óptimo y así sucesivamente.

4.1 Cálculo del punto óptimo

Después del primer vistazo, se tiene ya una cierta información sobre dónde pueden estar las puertas, en caso de existir alguna en el entorno. Las jambas que se hayan apreciado en la imagen se proyectan en el grid, elevando la probabilidad de todas las posiciones que explican esas observaciones. Debido a la ambigüedad de profundidad tanto la posición real como otras hipotéticas resultan resaltadas.

Desde el grid de probabilidad de jamba se extrae el *conjunto de puertas candidatas* formando pares con las celdillas de mayor probabilidad de jamba según se describió en la sección anterior. Para cada uno de esos pares se evalúa su probabilidad de puerta y el conjunto global queda ordenado de mayor a menor probabilidad. Tras la vuelta inicial ningún par habrá conseguido acumular evidencia suficiente como para sobrepasar el *umbral de aceptación* y dar la búsqueda por concluída, sin embargo ese conjunto representa las mejores hipótesis que el robot tiene para explicar las imágenes que ha visto. Previsiblemente la ubicación real estará contenida en ellas y será conveniente dirigir las nuevas observaciones para que confirmen o desmientan estas candidaturas.

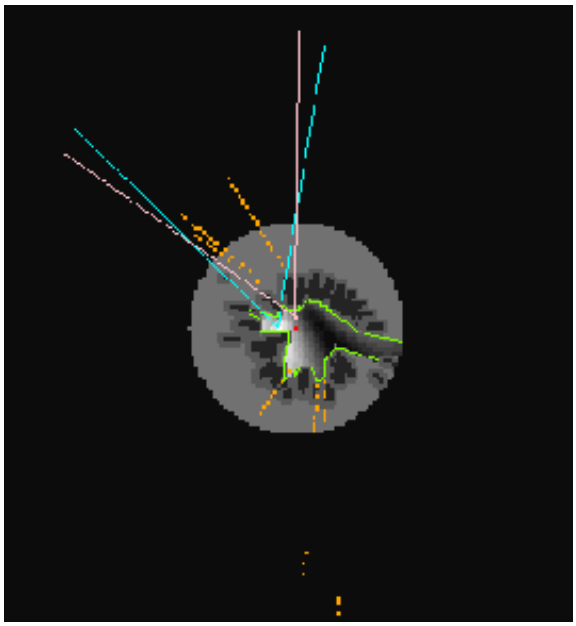


Figura7. Cálculo del punto desde el cual tomar la siguiente imagen. Para cada celdilla dentro de la zona aceptable se representa la utilidad de la mejor orientación de la cámara.

El lugar y orientación de la cámara desde el cual tomar la siguiente imagen del entorno se calculan maximizando una *función de utilidad*, definida en la ecuación (11). Por un lado cuantas más jambas candidatas se observen desde una posición su utilidad asociada es mayor, porque se pueden constrastar más candidaturas. En este

sentido el sumatorio de (11) se extiende a todas las jambas candidatas que caigan dentro del campo visual desde (x, y, θ). Por otro lado la utilidad es mayor cuanto menor es el solapamiento en la imagen entre las candidatas que se observen. Por ejemplo si desde una posición tres jambas candidatas aparecen en los mismos pixels entonces desde esa posición no se puede discriminar cual de ellas es la verdadera. Ese punto de observación es menos recomendable que otro desde el cual esas mismas candidatas aparecen en pixels distintos y por lo tanto se puede comprobar mejor la validez de cada una de ellas. En este sentido el sustraendo de (12) resta el solapamiento medio a la información que aporta esa jamba candidata observada desde ese punto. $solapamiento(jamba_i, jamba_j)$ es una función triangular que se hace 1 si ambas jambas candidatas proyectan en el mismo pixel y disminuye hasta 0 a medida que crece la distancia entre esos pixels de proyección.

$$utilidad(x, y, \theta) = \sum_{jamba_i \in campo\_visual} informacion(jamba_i) \tag{11}$$

$$informacion(jamba_i) = 1 - \frac{\sum_{i \neq j} solapamiento(jamba_i, jamba_j)}{\#jambas \in campo\_visual} \tag{12}$$

Al maximizar la utilidad (11) se encuentra tanto la mejor posición (x, y) como la orientación apropiada θ de la cámara para tomar la siguiente imagen. Para agilizar estos cálculos la función de utilidad sólo se calcula en posiciones aceptables: que no estén muy alejadas de la posición actual del robot (a menos de 3 m), que no estén cerca de ningún obstáculo y que sea directamente accesible con navegación local (un giro y una traslación). Estas condiciones reducen considerablemente el espacio de búsqueda.

En la figura 7 se puede apreciar la función de utilidad evaluada en las cercanías del robot después de la vuelta inicial reflejada en la figura 6. El punto central es el robot, y en él tiene el vértice el sector que representa la orientación actual de la cámara. Desde el punto central se abre también un contorno de celdillas directamente accesibles con navegación local sin chocar con los obstáculos, que son las manchas oscuras. Dentro de ese contorno se evalúa la función de utilidad, generando en ese caso un patrón que favorece a los puntos situados a la izquierda, haciéndose máxima en el vértice del segundo sector, que es el punto óptimo y tiene la orientación recomendada. Los puntos claros sueltos son las jambas candidatas que necesitan confirmación.

4.2 Navegación local

Una vez calculado el punto óptimo para situar la cámara se hace uso de arquitectura de control desarrollada para Xavier [Simmons94] encargando a la capa de navegación local la tarea de llevar al agente hasta aquella posición. No es necesario acudir a técnicas de navegación más compleja porque en el cálculo ya exigimos al punto destino que fuera alcanzable con un giro y una traslación.

Los navegadores reactivos toman control del robot hasta que finalizan su tarea. Entonces el sistema perceptivo vuelve a tomar y analizar una imagen, actualizando el grid de probabilidad de jamba y determinando si ha percibido alguna puerta o aún se necesita acumular más observaciones. Aunque la arquitectura lo permite se ha preferido no procesar imágenes mientras el agente se está moviendo.

5 Discusión

El sistema perceptivo presentado combina información visual, obtenida desde distintas imágenes, con datos ultrasónicos en un marco probabilístico para detectar la existencia y la posición de las puertas en el entorno. El enfoque permite integrar medidas que sólo aportan información parcial concluyendo la presencia o no de puertas en el entorno del agente móvil.

Cada vez que se obtienen nuevas imágenes y lecturas sónar se proyectan en un grid bidimensional siguiendo sendos modelos sensoriales. El grid contiene la probabilidad de que realmente exista una jamba en cada una de sus celdillas. A medida que se reciben nuevas observaciones debido a la dinámica de probabilidades se van descartando ubicaciones ficticias de las jambas y sólo las posiciones verdaderas acumulan suficiente evidencia sensorial. Emparejando distintas celdillas se obtiene el conjunto de puertas candidatas y su probabilidad asociada. El agente asume que ha visto una puerta cuando alguna de las candidatas sobrepasa el umbral de aceptación.

El sistema se ha materializado con éxito en un robot de interiores y se le ha dotado de un control autónomo que dirige los movimientos del agente y de su cámara para facilitar la percepción. El robot se mueve siempre hacia el punto que más discrimina entre la puertas candidatas existentes en cada momento, iterando hasta que finaliza la búsqueda. No obstante esta parte activa necesita aún más experimentación para conseguir más robustez en los resultados.

5.1 Trabajos relacionados

Además del trabajo realizado con pares estéreo la idea de combinar varias imágenes para inferir características 3D ha sido explorada profusamente. Por ejemplo Margaritis [Margaritis98] utiliza también la probabilidad como marco para localizar objetos en 3D a partir de visión. Su desarrollo teórico es más completo que el expuesto aquí: es realmente tridimensional e incluye modelos probabilísticos para reflejar el error inducido por el movimiento no determinista del robot. Sin embargo no aborda el problema de cómo dirigir ese movimiento para percibir mejor el estímulo buscado.

Un enfoque similar es el de Collins [Collins96][Collins97] que extrae estructura 3D del entorno, por ejemplo los edificios de un campus universitario desde varias imágenes aéreas. Como primitiva visual también utiliza bordes y los retroproyecta en el espacio 3D desde todas las imágenes disponibles. Las celdillas con los bordes reales quedan resaltadas porque recibirán más retroproyecciones que el resto. Una distribución estadística determina si el número de retroproyecciones que caen en una celdilla se debe al azar o a la existencia de un borde 3D real.

5.2 Ventajas e inconvenientes

El marco probabilístico utilizado ofrece muchas ventajas. La primera es que extrae la posición de la puerta sin necesidad de buscar homólogos. Para extraer mapas de profundidad, los pares estéreo emparejan puntos homólogos entre la imágenes derecha e izquierda y aplican triangulación. El problema de asignación de homólogos puede ser intratable si las dos imágenes han sido tomadas desde posiciones completamente distintas. Esa es precisamente nuestra situación con el robot, que puede observar la puerta desde lugares distantes más de un metro. En lugar de resolver explícitamente esta asignación de homólogos el grid de probabilidad permite representar la ambigüedad de cada imagen y solucionar el problema de modo implícito.

El enfoque probabilístico resulta válido para cualquier estímulo, no sólo puertas. Únicamente se necesitan los modelos sensoriales adecuados que traduzcan a probabilidad la información proporcionada por los sensores sobre el estímulo en cuestión.

Otra ventaja de este marco es que permite fusión sensorial de modo sencillo. Para ello se utilizan modelos distintos adecuados a los sensores diferentes pero que afectan a la misma probabilidad. También admite la incorporación de conocimiento a priori como valor inicial de la probabilidad de jamba. Por ejemplo si se sabe que la puerta estás más o menos por un área, esas celdillas tendrán un valor inicial mayor que el resto.

Esta aproximación facilita la tolerancia al ruido, a los falsos positivos y a fallos de observación. Por ejemplo si temporalmente deja de observarse una jamba por la presencia espúrea de un obstáculo interpuesto, el sistema funcionará bien si obtiene nuevas imágenes buenas con las que compensar ese fallo, la probabilidad acabará subiendo. Para materializar esa robustez los modelos de sensor deben diseñarse cuidadosamente.

Uno de los puntos débiles de este enfoque es el ruido en los encoders. Para poder mezclar distintas imágenes su información se ancla espacialmente a la posición del robot cuando tomó cada una de ellas. Si la estimación de posición es errónea la mezcla de probabilidades no es válida. Hemos utilizado encoders para estimar la posición y es de sobra conocido en la literatura el error acumulativo que padecen. No obstante las posiciones del robot solo necesitan ser relativamente válidas, es decir precisas respecto de la posición actual del agente, no en términos absolutos. Ese tipo de precisión se tiene con los encoders si sólo se tienen en cuenta medidas recientes, dentro de una ventana temporal en la que no da tiempo a acumular demasiado error relativo de posición. Por otro lado el propio grid asume bien errores de localización pequeños.

Otra desventaja es el actual tamaño de las celdillas, 10 cm, que pone en duda la precisión obtenible con este sistema. Sin embargo el enfoque se muestra eficiente para comprobar la propia existencia de puertas, y su posición dentro de la resolución del grid. Si se quiere más precisión se puede complementar con otros algoritmos, que resultan mucho más sencillos una vez que la existencia del estímulo puerta ha sido verificada.

5.3 Líneas futuras

Está previsto continuar el trabajo mejorando la parte activa, que no está suficientemente contrastada. En concreto la extracción en cada momento de las puertas candidatas es clave puesto que ellas determinan implícitamente el siguiente punto de observación. Umbrales fijos para esa extracción resultan inadecuados porque al principio la probabilidad acumulada es baja pero va creciendo con nuevas observaciones. El esquema de selección debe mantener en todo caso cierta discriminación sin excluir buenas candidatas fuera. Se está pensando en umbrales dinámicos que tengan en cuenta esa acumulación.

También se tiene pensado portar el sistema al robot B21-Hermes del Instituto de Automática Industrial e implementar el comportamiento *atraviesa-puerta* que motivó inicialmente este sistema y que debe sacar beneficio de todo el trabajo perceptivo realizado.

Agradecimientos

Este trabajo ha sido promocionado por la Comunidad Autónoma de Madrid a través de su programa de becas de Formación de Personal Investigador y realizado en los laboratorios del Computer Science Department de Carnegie Mellon University.

Referencias

[Elfes90] Alberto Elfes, Occupancy grids: a stochastic spatial representation for active robot perception. En *actas de la Sixth AAAI Conference on Uncertainty in AI*, Julio 1990.

[Simmons94] Reid Simmons, Structured Control for Autonomous Robots. En *IEEE Transactions on Robotics and Automation*, 10:1, páginas 34-43, Febrero 1994.

[Simmons95] R. Simmons y S. Koening, Probabilistic navigation in partially observable environments. En *actas de International Joint Conference on Artificial Intelligence*, Montreal, Julio 1995.

[Simmons97] R. Simmons, R. Goodwin, K. Zita Haigh, S. Koening y J. O'Sullivan, A layered architeture for office delivery robots. En *actas de la ACM International Conference on Autonomous Agents*, páginas 245-252, Marina del Rey (USA), Febrero 1997.

[Collins96] R. Collins, A space-sweep approach to true multi-image matching. En *actas de la IEEE Conference on Computer Vision and Pattern Recognition*, páginas 358-363, San Francisco, Junio 1996.

[Collins97] R. Collins, Multi-image focus of attention for rapid site model construction. En *actas de la IEEE Conference on Computer Vision and Pattern Recognition*, San Juan (Puerto Rico), Junio 1997.

[Margaritis98] D. Margaritis y S. Thrun, Learning to locate an object in 3D space from a sequence of images. En *actas de la International Conference on Machine Learning*, páginas 332-340, 1998.

[Smith97] S. Smith y J.M. Brady, SUSAN-A new approach to low level image processing. En *International Journal of Computer Vision*, 23(1):45-78, Mayo 1997.

Sesión 4: Agentes Físicos II

Un Sistema Multi-Agente basado en lógica difusa aplicado al ámbito de la RoboCup *

Eugenio Aguirre, Juan Carlos Gámez, and Antonio González

Departamento de Ciencias de la Computación e Inteligencia Artificial
E.T.S. de Ingeniería Informática. Universidad de Granada
18071-Granada phone: +34.958.246143 fax: +34.958.243317
e-mail: {eaguirre,A.Gonzalez}@decsai.ugr.es, jcgamezg@terra.es

Resumen La Inteligencia Artificial ha sido aplicada con éxito en numerosos y variados dominios. En particular son interesantes los recientes avances que se están produciendo en el desarrollo de entornos multi-agente dinámicos, colaborativos, de tiempo real y con adversario. En este trabajo desarrollamos una propuesta que utiliza un simulador de robots que juegan al fútbol, llamado TeamBots, que simula la modalidad de campeonato "RoboCup Small", encuadrado dentro del campeonato mundial de "RoboCup". El objetivo es poner en marcha un equipo con el que iniciar nuestra investigación en el tema. Así nos hemos propuesto el desarrollo de un equipo al completo, y posteriormente ir perfeccionando las capacidades del mismo. En este trabajo mostramos las líneas generales de diseño del equipo, y mostramos en detalle la estrategia de juego para un jugador de tipo portero, mostrando además parte de la amplia experimentación realizada para poner de manifiesto el correcto funcionamiento del equipo.

Introducción

La Inteligencia Artificial, y más concretamente los llamados Sistemas Inteligentes, ha tenido un notable progreso a partir de su aplicación en diferentes problemas reales. En este trabajo presentamos una aplicación de la *Inteligencia Artificial* y la *Robótica* en una faceta de la vida que hasta ahora ha sido meramente humana, como es el fútbol. La idea inicial surge a raíz de un congreso en Tokio, en 1.993, donde se gesta un proyecto llamado *Robot J-League* que más tarde daría paso a *Robot World Cup Initiative*, más conocido como *RoboCup*. Según se define en la página oficial, *"RoboCup es un proyecto internacional para promover la Inteligencia Artificial, la Robótica y los demás aspectos relacionados"* [2]. Para ello utilizan el fútbol como herramienta con diversos y variados fines. En el aspecto técnico, RoboCup es un sistema que aúna características que en otros dominios era impensable su integración, entre las que destacan el disponer de un entorno dinámico, de tiempo real y distribuido. En el aspecto social, RoboCup se puede ver como una forma amigable y llamativa de motivar a la investigación en estas áreas tanto a investigadores noveles, como a los más experimentados, teniendo como objetivo el desarrollo de un equipo que, de forma inteligente, sea capaz de jugar a fútbol.

* Este trabajo ha sido parcialmente financiado por la CICYT dentro del proyecto TAP1999-0535-C02-01

Robocup proporciona un dominio de investigación desafiante, el cual involucra múltiples agentes que necesitan colaborar en un entorno con adversarios para alcanzar un objetivo específico. Nos brinda la posibilidad de trabajar en el estudio de los Sistemas Multi-Agentes en dominios complejos de tiempo real, lo que requiere agentes para actuar eficientemente tanto autónomamente como por parte de un equipo. Como decimos el equipo está formado por un sistema de robots multi-agentes, con percepción global y, acciones y conocimiento distribuido.

El principal objetivo es conseguir que un equipo formado por cinco robots puedan desarrollar satisfactoriamente un partido de fútbol. Para conseguir este objetivo ha sido necesario realizar el diseño de las habilidades de cada tipo de agente y establecer la colaboración necesaria entre ellos para que puedan formar un equipo en conjunto. También se han llevado a cabo las tareas de implementación necesarias para obtener un equipo operativo y preparado para jugar contra otros equipos ya existentes. Además, nuestro diseño hace posible con cierta facilidad, la continua mejora del equipo y la incorporación de nuevas ideas y técnicas que se decidan poner en práctica a raíz de la experimentación realizada.

Los resultados y conclusiones de este estudio son de gran aplicación en multitud de dominios [5] como por ejemplo *Sistemas Industriales Complejos* (COSY, Complex Systems), que pueden ser integrados dentro de los Sistemas Distribuidos como por ejemplo redes de distribución de energía o material, grandes plantas de producción con varias estaciones y procesos dinámicos o discretos con un gran número de variables. Del mismo modo, otro campo de aplicación en el que se puede observar una extrapolación directa es la salvación de personas en escenarios extremos [2], donde se podría resolver la situación sin necesidad de poner en peligro la vida de otra persona.

Asimismo es de gran interés por sus propiedades de complejidad, dinamismo, conocimiento incierto y metas variables, siendo las características principales de este dominio tiempo real, con ruido, colaborativo y con adversarios.

Igualmente es calificado como uno de los mejores bancos de prueba, ya que puede ser usado para evaluar diferentes técnicas de sistemas multi-agentes de manera directa, según el desarrollo y los resultados del juego de equipos implementados con diferentes técnicas. Entre estas técnicas cabe destacar el intercambio dinámico de roles o intercambio de roles en plena simulación [10], predicción en tiempo real [12], la introducción de un análisis del equipo contrario, introducción de aprendizaje en el equipo [9, 11, 8], planes precompilados [6], memoria predictiva [1, 12], etc.

Un aspecto muy a tener en cuenta es que para que un equipo de robots pueda realmente desarrollar un partido de fútbol deben ser maduradas una amplia sistemática, incluyendo: principios de diseño de agentes autónomos, colaboración multi-agente, adquisición de estrategias, razonamiento en tiempo real, etc. En concreto, en este nuestro caso, hemos desarrollado un equipo formado por 5 robots en el que tenemos un portero, dos defensas y dos delanteros; donde cada robot es un agente autónomo, dándose también la colaboración con sus compañeros en aquellas circunstancias que así se precise. La autonomía de cada robot implica que cada jugador desarrolle sus

habilidades individuales de forma independiente. Estas habilidades se desarrollan por todos los tipos de jugadores con pequeños matices que ayudan a que cada jugador desarrolle la actividad más acorde con el tipo de jugador de que se trate. En el desarrollo de estas habilidades individuales se hace uso de técnicas básicas como la geometría para la obtención de ángulos, distancias y velocidades necesarias para el control del robot en el dominio. Las habilidades colaborativas se desarrollan entre iguales, en este equipo no hay jerarquía alguna, considerándose todos exactamente iguales, es decir, no hay capitán ni preferencias o privilegios por las acciones o decisiones de ningún jugador. Para llegar a un consenso en estas habilidades colaborativas se utiliza las posibilidades de comunicación que proporciona el simulador

Finalmente para el desarrollo del equipo, se hace uso de un Sistema Basado en Reglas Difusas mediante el cual se considera una estrategia conjunta de juego. Dicho Sistema de Reglas Difusas permite seleccionar las habilidades, las acciones o decisiones a tomar en cada momento, y se detallará a lo largo de este trabajo.

Una vez que se ha situado el problema y se han comentado las técnicas a utilizar, en la siguiente sección daremos una descripción general de las principales ideas que hemos considerado en el diseño de nuestro equipo, comentando algunas de las diferentes habilidades implementadas para los jugadores. En la sección 3 comentamos la estrategia que gobierna el equipo, centrándonos en el portero como ejemplo del desarrollo de dicha estrategia. Para finalizar analizamos la experimentación realizada con nuestro equipo y unas notas sobre posibles trabajos futuros.

Arquitectura

En esta sección describiremos a groso modo lo que consideramos más interesante del equipo realizado. En primer lugar para situarnos mostraremos algunas definiciones sobre agentes, sistema multi-agente, inteligencia artificial distribuida, y una breve taxonomía sobre sistemas multi-agentes. A continuación presentamos el simulador utilizado para el desarrollo de nuestro equipo y su experimentación, para finalmente mostrar el equipo en sí, destacando la escalabilidad del mismo así como dos habilidades interesantes de comentar y la técnica utilizada como ayuda en la implementación de las habilidades y la toma de decisión. Sin más pasamos a mostrar cada uno de estos apartados.

Algunas definiciones

Para el desarrollo de nuestra propuesta en primer lugar consideramos algunas definiciones y una taxonomía sobre sistemas multi-agentes.

En este trabajo vamos a considerar que *"un* agente *es una entidad con percepciones, metas, acciones y conocimiento del dominio, situado en un entorno"* [7]. El modo de actuar ante las percepciones o su planificación a la hora de actuar se llama "comportamiento". Un grupo de agentes en un Sistema Multi-Agente que comparte metas comunes se dice que forman un "equipo". Los miembros de un equipo, o

"compañeros", coordinan sus comportamientos adoptando procesos de conocimiento compatibles. Otros agentes del entorno tienen objetivos opuestos, son los "adversarios" o "miembros del equipo contrario".

La *Inteligencia Artificial Distribuida* es considerada como un subcampo de la *Inteligencia Artificial* desde al menos dos décadas. Tradicionalmente, la Inteligencia Artificial Distribuida estaba dividida en dos subdisciplinas: *Resolución de Problemas Distribuidos* y *Sistemas Multi-Agentes*.

Los Sistemas Multi-Agentes los podemos clasificar en:

- *Sistemas Multi-Agentes Homogéneos No Comunicativos*: son aquellos en los que todos los agentes tienen la misma estructura interna, incluyendo metas, conocimiento del dominio y posibles acciones. No tienen posibilidad de comunicación.
- *Sistemas Multi-Agentes Heterogéneos No Comunicativos*: hay diversas variedades, desde tener diferentes metas, hasta tener distintos modelos de dominios y acciones. No tienen posibilidad de comunicación
- *Sistemas Multi-Agentes Heterogéneos Comunicativos*: el comentario sería el mismo que el anterior con la salvación de que éstos sí tienen posibilidades de comunicación.

Éstos últimos son los utilizados en nuestra propuesta.

El simulador de RoboCup TeamBots

Desde el comienzo del proyecto internacional, RoboCup, ha habido desarrollos de simuladores en diferentes lenguajes: LISP, C++, Visual C++, Java, etc. y en diferentes plataformas: Linux, Windows, UNIX, Iris, etc. La opción elegida es el simulador TeamBots [3, 4], que como define su propio autor: *"TeamBots es un conjunto de aplicaciones, programas y paquetes Java orientados a la investigación de robots móviles Multi-Agentes. La distribución TeamBots es de código fuente completo libre, se trata de un software completo que soporta prototipos, simulación y ejecución de sistemas de control multirobot. El entorno de simulación es 100% Java"*. Además, al estar desarrollado en Java, se puede utilizar en todas las plataformas con el único requisito que posean el correspondiente compilador e intérprete Java

Dentro de los diferentes dominios para los que se puede utilizar la distribución TeamBots, destacar SoccerBots ya que es el que simula la dinámica y dimensiones de un juego de liga RoboCup Small, formado por dos equipos de 5 robots que compiten sobre un campo de dimensiones similares a una mesa de ping-pong conduciendo y golpeando una pelota naranja, parecida a una pelota de golf, a la meta del contrario. La figura 1 representa un escenario típico de la SoccerBots.

En lo referente a la utilización de la distribución como banco de pruebas, comentar que es un software desarrollado en Java por lo que utiliza el paradigma orientado a objetos. La distribución está formada por un conjunto de 9 paquetes, siendo "EDU.gatech.cc.is.abstractrobot" el paquete más interesante para el control del robot, integrando un extenso diagrama con alrededor de 38 elementos entre clases

Figura1. Ejemplo de SoccerBots dentro de la distribución TeamBots

e interfaces. Formando parte de este diagrama se encuentra la clase ControlSystemSS de la que debe heredar el objeto que vaya a ser el Sistema de Control o sistema que gobierne el desarrollo de nuestro equipo en la simulación. De ahí la necesidad de tener una clase que herede de ésta e implemente los métodos "configure" y "takeStep" encargados de la inicialización y seguimiento de la simulación. En la siguiente sección se muestra una breve explicación sobre el diagrama de clases implementado.

El equipo de agentes

Hay diversas formas de abordar este trabajo, que pueden ser consideradas como diferentes niveles de dificultad, por ejemplo, cada jugador puede ser considerado independiente y desarrollar una función u otra según la posición que ocupe; cada jugador es independiente pero existe una comunicación con los demás para desarrollar cualquier tipo de juego; hay una jerarquía con un capitán que es el que organiza el juego pudiendo cada uno de los restantes tener, o no, su propio razonamiento; o bien, hay un capitán que controla todo el juego y los demás realizan todas las acciones como indica el capitán.

Igualmente, hay que tener en cuenta que los robots deben ser capaces de ir hacia el balón, conducir el balón de un punto a otro, ir hacia los contrarios, chutar, regatear, defender, ..., es decir, deben ser capaces de desarrollar habilidades individuales como: movimiento del jugador en todo el campo, donde podemos destacar el acto de evitar colisiones con los demás; conducción del balón, en cuyo movimiento destaca el regate con el balón a un obstáculo; la defensa de la portería, tanto por el portero como por el resto de los jugadores; y la principal habilidad colaborativa, ésta es, el pase.

Para implementar todas estas acciones o habilidades, y como continuación de lo comentado en el apartado anterior, hacemos uso de las facilidades que proporciona un lenguaje orientado a objetos como Java. En primer lugar, el desarrollo del equipo se realiza tomando como base las habilidades que debe poseer todo jugador

de fútbol, con esto creamos una clase jugador que posee todos los comportamientos generales de un jugador de fútbol, creando a partir de éste los diferentes tipos de jugadores a través de los mecanismos de herencia y sobrecarga, y desarrollando nuevos comportamientos. Por ejemplo, tenemos una clase jugador que desarrolla todas las habilidades generales de un jugador. Como sabemos un portero es un jugador, por lo que creamos una clase portero que hereda todo el comportamiento del jugador general, pero refina por ejemplo la defensa que es algo diferente y crea nuevos comportamientos que son exclusivos del portero. A su vez dentro del tipo de jugador "portero" puede haber diferentes tipos de porteros, un portero más arriesgado, que salga más del área, un portero que no se aleje demasiado de la portería, etc., cada uno heredaría su comportamiento del portero general refinando o modificando algunas características concretas y así podríamos continuar hasta el nivel que deseemos llegar. Este razonamiento es igualmente extrapolable a los demás tipos de jugadores. Esta explicación puede ser mejor comprendida con el diagrama de la figura 2.

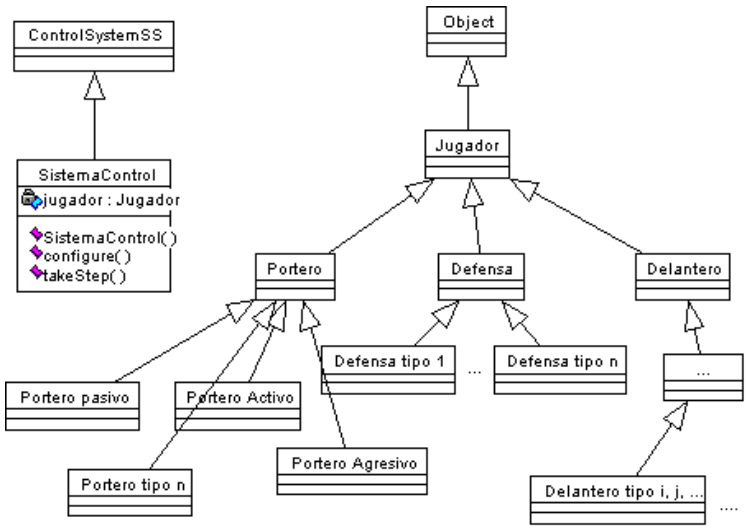


Figura2. Ejemplo de diagrama de clases

Para el desarrollo de la lógica se introduce una clase abstracta entre la clase jugador general y el tipo de jugador correspondiente considerando que la estrategia de juego es parte del comportamiento de un jugador pudiéndose realizar variaciones de la estrategia general en cada uno de los subtipos del tipo de jugador correspondiente.

El desarrollo del equipo completo ha supuesto la elaboración de un conjunto amplio de habilidades. A continuación y, como ejemplo de las ideas contempladas, mostramos esquemáticamente la solución desarrollada a algunas de las habilidades de los jugadores.

Conducción del balón (regate) Dentro de la interacción con el balón, nos encontramos con una de las habilidades más difíciles, la realización de un regate. En este

caso, destacar, como sabemos, que se trata de robots sin miembros, que se pueden considerar como esferas, con la complicación que conlleva tratar que una esfera de mayor tamaño controle los movimientos de otra esfera de un tamaño menor.

En breve la explicación de este movimiento sería: calculamos tres vectores, un vector del balón al destino, un vector del jugador al balón y el vector del destino al que vamos, en función de los cuales se realizan una serie de cómputos llevándose a cabo el regate. A groso modo lo que se realiza es lo siguiente: se comprueba si el contrario está lejos y si es así no se hace nada, en caso contrario, se lleva a cabo el regate, para lo cual se calcula una posición de regate hacia la que debe dirigirse el jugador: en primer lugar se dirige hacia el obstáculo, golpeando el balón en ese movimiento y alejándolo del obstáculo, para posteriormente dirigirse al balón y conducirlo a la posición deseada. Este tipo de movimiento puede entenderse de manera gráfica en la figura 3.

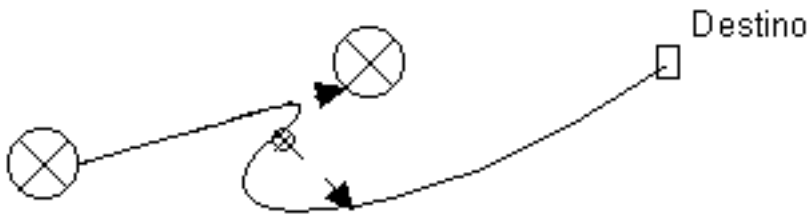


Figura3. Movimiento de Regate

Habilidad Colaborativa (pase) Pasamos ahora a comentar el pase desde el punto de vista de acción colaborativa entre robots. Para ello, aunque no es estrictamente necesario, en algunos sistemas se produce una comunicación previa para desarrollar esta habilidad. Lo que se pretende es realizar el intercambio de balón entre jugadores, lo que comúnmente se denomina pase. Para ello establecemos una comunicación entre dichos jugadores, facilitando de esta forma la transmisión de las intenciones, y así poder llegar a un consenso a la hora de realizar el pase. Por tanto, lo primero que debemos realizar es la implementación de las funciones que posibiliten dicha comunicación, permitiendo al mismo tiempo la comunicación no sólo con un jugador, sino con todos los jugadores; y finalmente realizar la implementación necesaria para el traspaso del balón de un jugador a otro.

Este proceso de pase se puede sintetizar en la realización de dos acciones complementarias, por una parte el jugador que realiza el pase debe comunicarse con el que lo recibe y acercar el balón al mismo, por uno u otro medio; por otra parte, el jugador que recibe el balón debe ser capaz de recibir la comunicación del compañero y ponerse de acuerdo con él a la hora de la recepción del balón. Esto nos

lleva a implementarlo en dos funciones una para la transmisión del balón y otra para la recepción del balón, ayudadas claro está, por las funciones necesarias para el desarrollo de la comunicación.

Representación 3-D del terreno Comentar, igualmente, el desarrollo de una representación 3-D del terreno para facilitar principalmente las decisiones de chutar, pasar, y otras habilidades relacionadas con el balón y la posición de los contrarios. Como decimos se trata de un tipo de representación de terreno distinto, en el cual se divide el campo en celdas o casillas y se coloca cada robot presente en el campo en la casilla correspondiente. A la posición que ocupa el robot se pondera con un valor determinado y conforme nos alejamos de dicha posición se va disminuyendo dicho valor, creando una especie de montaña donde la cima es la posición del robot. Además en el caso de que haya dos robots juntos estos valores se suman creándose una montaña de mayor tamaño y con dos cimas. Al finalizar este tipo de representación y extrapolándola a un tipo de representación en 3-D el campo de fútbol se convertiría en una zona de montañas y valles, donde las montañas significan lugares difíciles de atravesar y los valles lugares de fácil acceso, con lo que sería muy sencillo determinar si la posible trayectoria de un balón está obstruida o no en función de la altura de las montañas que atraviese.

Descripción de un tipo de agente

Una vez que hemos comentado brevemente el equipo, pasamos a explicar el desarrollo de la lógica difusa que gobierna el comportamiento del equipo en una simulación.

Para la definición de la estrategia de entre las múltiples opciones disponibles hemos elegido utilizar la lógica difusa [13], pero con la diferenciación necesaria según el tipo de jugador que deba realizar la acción, ya que, como es evidente, el juego del portero es diferente del de un defensa y distinto de un delantero. Esta diferencia se ha implementado finalmente con la utilización, como hemos comentado en el apartado 2.3, de clases que gestionan la lógica diferente para cada tipo de jugador.

Como es natural, esta separación o diferencia se puede realizar a distintos niveles debiendo llegar a un compromiso entre eficiencia, mantenimiento y funcionalidad, escogiendo en nuestro caso lo que mostramos a continuación. Aunque se han implementado la lógica difusa de todos los tipos de jugadores nos centraremos en uno únicamente por cuestiones de espacio.

Lógica Difusa utilizada en el Portero

Se ha realizado un portero con la peculiaridad de ser un portero "pasivo", con esto lo que queremos poner de manifiesto es que no se trata de un portero muy decidido a salir del área de la portería ante cualquier situación de peligro, sino que es un portero que defiende la portería prácticamente debajo de los palos que la forman. Sin embargo, esto no entraña ninguna limitación pudiéndose realizar una pequeña

modificación en la actitud del portero simplemente con la corrección de alguno de los parámetros o introducción de los pesos que forman la ponderación en la elección de la regla a disparar, u otros muy diversos métodos no difíciles de implementar con esta base.

La lógica que gobierna el comportamiento de un portero podría ser definida como sigue, para ello consideramos el punto de vista del portero:

- Si estoy fuera de mi área permitida, quiere decir que estoy lejos de la portería y la zona que debo defender, entonces debo ir a la posición por defecto que tenga el jugador, en este caso el portero, es decir, a mi portería.
- Si el balón está fuera de mi área, que podríamos llamar área de acción, y además no se ha dado el caso anterior ya que si no, saltaría la regla anteriormente detallada, debo posicionarme en posición de defensa de la portería. Esta posición la explicaremos con más detalle en la sección siguiente.
- Si estoy dentro de mi área, el balón también está dentro del área y está delante de mí, entonces hay una situación de peligro ya que el balón está cerca de mi portería, y debo ir hacia el balón para despejarlo.
- Si estoy dentro de mi área, el balón también está dentro del área y está a la misma altura que yo, debo ir hacia el balón y despejarlo, pero en este caso, se activa un comportamiento algo distinto, ya que entraña algo más de peligro que en el caso anterior, por estar el balón a mi misma altura.
- Si estoy dentro de mi área, el balón también está dentro del área y el balón lo tengo detrás y tengo muchas posibilidades de meter el balón en mi portería, entonces debo localizar al contrario más cercano, que casi con toda seguridad es el que ha conducido el balón a mi portería, y debo bloquearle porque con toda probabilidad intentará acercarse al balón para meter el gol; ésta es la única alternativa que tengo en esta situación, ya que si intento ir por el balón para recuperarlo seguramente sea yo el que meta el gol en mi propia portería.
- Si estoy dentro de mi área, el balón también, tengo el balón detrás y no hay peligro de meter el balón en la portería, entonces voy a por el balón para después poder aplicar otra regla de las anteriores e intentar despejarlo.

Las variables difusas que forman esta lógica son las siguientes:

- *Área*: en función de la posición del jugador a la portería nos dice si está dentro o fuera del área.
- *Balón*: en función de la posición del balón y el jugador nos muestra si el balón está delante, detrás o a la misma altura que el jugador.
- *MetoBalón*: en función de la posición del jugador y la portería nos indica la posibilidad de que el jugador meta el balón en su propia puerta.

Los conjuntos difusos correspondientes a estas variables difusas son mostrados por las figuras 4, 5 y 6.

En la variable difusa *Área*, se observa que hasta una distancia menor que 0.4 metros se considera que el jugador está dentro del área, a partir de ahí se empieza

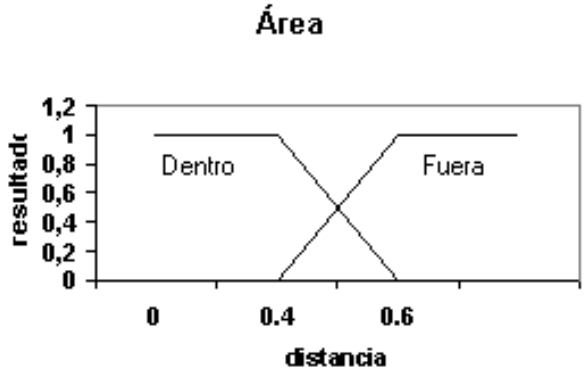


Figura4. Área

una zona un poco "difusa" en la que se va perdiendo la creencia de que está dentro y a partir de 0.6 metros se considera que está fuera. Se toman estos valores basados en las dimensiones del terreno de juego, por ejemplo, medio campo es aproximadamente 1.5m x 1.5m por lo que esto representa 1/3 del medio campo.

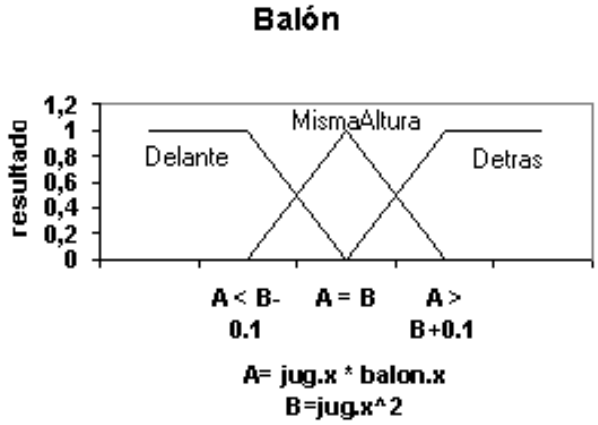


Figura5. Balón

En la variable difusa *Balón*, la elección de la posición del jugador respecto al balón, el balón delante, detrás o a la misma altura que el jugador, se realiza en función de los valores del jugador y el balón en el eje X. Por ejemplo, en el campo oeste, si el valor X del jugador es mayor que el valor X del balón, el balón está delante del jugador. Estas decisiones debemos hacerlas independientes del campo en el que juguemos, para lo cual eliminamos los signos utilizando la multiplicación. Para ello, elevamos el valor de X del jugador al cuadrado y multiplicamos los valores

de X del jugador y del balón, de esta manera si el valor de X del jugador al cuadrado (B) es mayor que el valor de X del jugador por el valor de X del balón (A), el balón está delante del jugador. Además hacer notar que el valor devuelto por este conjunto difuso sólo se tiene en cuenta en el caso en que el balón está dentro del área y por lo tanto cercano al portero. Además se toma 0.1m, ya que el diámetro del jugador es aprox. 0.12m y el del balón 0.05m considerando que el intervalo de confianza es algo menos del necesario para un jugador.

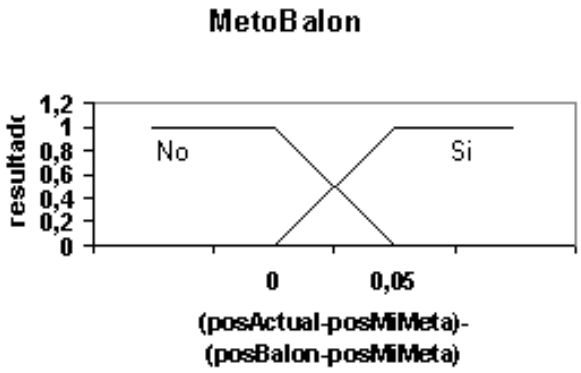


Figura6. MetoBalón

Mediante la variable difusa *MetoBalón*, se pretende comprobar si puede ser que se meta un gol en propia puerta. Para ello se crean dos vectores, uno de mi posición a la portería y otro de la posición del balón a la portería, y en función del ángulo que forman y un valor de confianza se dice la posibilidad o no de que se produzca un gol en propia meta por parte de dicho jugador. Se toma el valor 0.05 ya que los ángulos son medidos en radianes y este valor corresponde a un ángulo muy pequeño pero suficiente para que al ir hacia la portería golpee el balón y al mismo tiempo lo desvíe de su trayecto a la portería.

Igualmente, correspondientes a las variables y conjuntos difusos anteriores, las reglas difusas que gobiernan el comportamiento del portero son las siguientes:

1. if Area(posActual, fuera) –> goTo(posDefecto)
2. if Area(posBalon, fuera) –> defiende()
3. if Area(posActual, dentro) y Area(posBalon, dentro) y Balon(delante) –> despeja()
4. if Area(posActual, dentro) y Area(posBalon, dentro y Balon(mismaAltura) –> despeja()
5. if Area(posActual, dentro) y Area(posBalon, dentro) y Balon(detrás) y MetoBalon(sí)

–> goTo(posContrarioMasCercano)

6. if Area(posActual, dentro) y Area(posBalon, dentro) y Balon(detrás) y MetoBalon(no)
   –> goTo(posBalon)

Como nota de implementación comentar que en caso de empate entre reglas, se toma en el orden presentado.

Esta sería la descripción básica de la lógica, difusa en este caso, que haría de un robot, un portero con razonamiento. También hemos desarrollado lógicas para los otros jugadores considerados, pero con pequeñas modificaciones como por ejemplo, en el caso del delantero tenemos 6 variables difusas con los siguientes valores posibles de los conjuntos difusos: Area(dentro, fuera), Balon(esMio, cerca, lejos), BalonCampo(miCampo, suCampo), Porteria(libre, ocupada), CubroPorteria(si, no), CompaneroDelante(libre, cubierto), formando en este caso la lógica difusa un total de 8 reglas; para el caso del defensa definimos 5 variables difusas: BalonCampo(miCampo, suCampo), Balon(esMio, cerca, lejos), BalonContrario(esMio, cerca, lejos), BalonPos(delante, mismaAltura, detras), MetoBalon(si, no) siendo en este caso un total de 6 reglas las que integran esta lógica.

Una vez que hemos descrito algunas de las componentes más importantes para entender el diseño de nuestra propuesta, pasamos a mostrar la experimentación realizada con nuestro equipo.

Experimentación

En esta sección se pretende mostrar, de forma más tangible, mediante tablas de resultados de simulaciones, las conclusiones obtenidas del desarrollo del equipo.

El equipo que se ha desarrollado se ha realizado, como todo software, como fruto de una evolución a través de distintas versiones, mejorando cada versión a la anterior. Por motivos de espacio, no se muestran todos los resultados mostrando únicamente los de la última versión desarrollada, donde están integradas todas las habilidades de los jugadores, tanto individuales como colaborativas y competitivas, así como la representación 3-D del terreno, la utilización de la Lógica Difusa en la estrategia del equipo y demás características.

Para realizar estas experimentaciones se han utilizado otros equipos que acompañan a la distribución. Estos equipos han sido desarrollados por grupos que lo aportan junto con su código, e incluso añaden algunos comentarios, de forma desinteresada.

El primer ejemplo de experimentación que vamos a mostrar, nos permite comprobar el funcionamiento del equipo en una simulación en la que debe enfrentarse a otros equipos, los cuales poseen diferentes estrategias y comportamientos. Los resultados de estas simulaciones se muestran en la tabla 1, donde los valores de la parte izquierda corresponden al número de goles conseguido por nuestro equipo y el valor de la derecha corresponde al número de goles conseguido por el equipo contrario.

Tabla1. Tabla de resultados en competición con otros equipos

Resultados	Sistema Control-UGR
BasicTeam	23 0
GoToBall	12 3
CommTeam	16 1
AIKHomoG	1 1
BrianTeam	14 0
CDTeamHetero	10 0
DTeam	4 0
DaveHeteroG	0 0
DoogHomoG	1 2
DoogHeteroG	1 5
FemmeBotsHeteroG	2 0
JunTeamHeteroG	23 0
Kechze	3 1
LoneForwardTeamHomoG	0 1
MattiHetero	2 1
PermHomoG	0 0
SibHeteroG	1 2
SchemaDemo	2 0
SchemaNewHetero	4 0

En ellos se observa que por lo general se obtienen buenos resultados con una notable ventaja sobre la mayoría de los equipos, los cuales implementan diferentes estrategias, desde estrategias simples, otras más complejas, incluso estrategias de juego poco elegantes, dedicándose a ir hacia un contrario y bloquearle en un rincón para inutilizarlo.

El segundo ejemplo de experimentación que vamos a mostrar es la evolución de una especie de campeonato entre los equipos conseguidos. Para esto utilizamos los datos de otra experimentación realizada donde todos los equipos juegan contra todos y una vez en cada campo. Estos datos determinan un campeonato donde, por ejemplo, la simulación de un equipo en el campo Este es la ida y la simulación de ese equipo en el campo Oeste es la vuelta. Realizando todas estas simulaciones se obtiene una evolución del campeonato como se muestra en la figura 7, donde se observa que se obtiene la victoria final de nuestro equipo.

Conclusiones y Trabajos Futuros

En este trabajo hemos puesto en marcha un equipo con habilidades suficientes como para ser competitivo a nivel básico, y que pueda servir como plataforma de prueba sobre distintas investigaciones y desarrollos. Es importante poner de manifiesto la gran flexibilidad de la propuesta realizada y su capacidad de mejora tan solo refinando algunas componentes del modelo.

De esta forma este trabajo pretende ser una toma de contacto con el tema en cuestión, realizando el desarrollo de un equipo completo que lleva a cabo todas las ha-

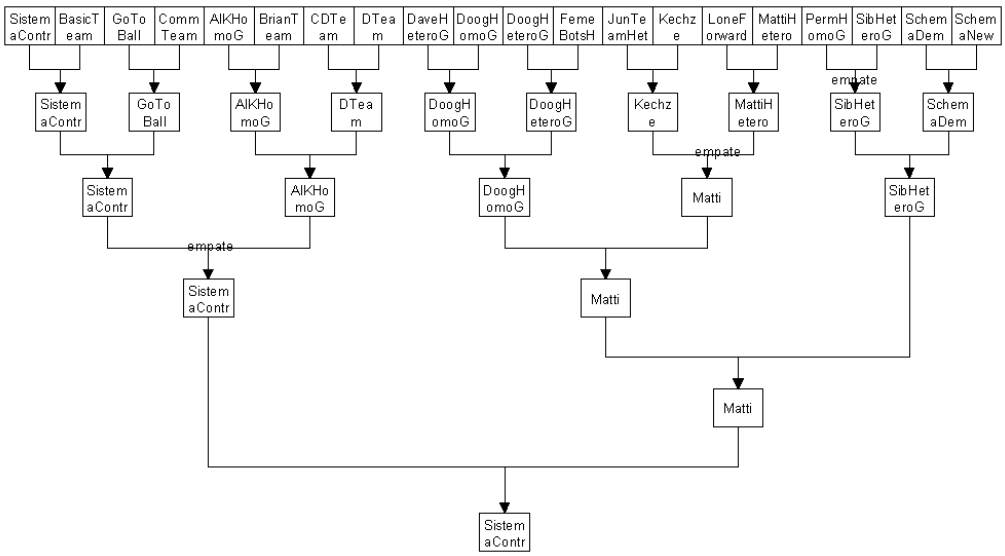


Figura7. Resultados de la ejecución del equipo SistemaControl-UGR contra los demás equipos

bilidades de un jugador de fútbol, dentro, claro está, de las limitaciones intrínsecas del entorno; cumpliendo con creces con el propósito inicial de desarrollo e implementación necesaria para que un equipo formado por 5 robots pueda desarrollar satisfactoriamente un partido de fútbol.

Finalmente, como trabajos futuros se puede pensar en el desarrollo y evaluación de diferentes alternativas de las técnicas utilizadas en este trabajo, como por ejemplo, el desarrollo de otra variación u otra lógica difusa que gobierne el equipo o el desarrollo de lo necesario para el intercambio dinámico de roles.

Referencias

[1] M. Bowling, P. Stone, and M. Veloso. Predictive memory for an inaccessible environment. In *Working Notes of the IROS-96 Workshop on RoboCup*, Osaka, Japan, 1996.

[2] Página Web Oficial de RoboCup. http://www.robocup.org.

[3] Página Web de TeamBots mantenido por Tucher Balch. http://www.teambots.org.

[4] Página Web del Simulador TeamBots. http://www.cs.cmu.edu/~trb/TeamBots.

[5] A. Oller, J. Ll. de la Rosa, R. García, J.A. Ramon, and A. Figueras. Micro-robots playing soccer games: A real implementation based on a multi-agent decision-making structure. Technical report, Electronics, Electrics and Automation, Engineering Departament, Universitat Rovira i Virgili; Intelligent Systems and Control Engineering Group, IIiA, University of Girona&LEA-SICA, 1998.

[6] P. Stone. *Layered Learning in Multi-Agent Systems: A Winning Approach to Robotic Soccer.* MIT Press, 2000.

[7] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning. Technical Report CMU-CS-97-193, School of Computer Science, Carnegie Mellon University, Pittsburgh, December 1997.

[8] P. Stone and M. Veloso. Towards collaborative and adversarial learning: A case study in robotic soccer. *International Journal of Human-Computer Studies*, 48(1):83–104, 1998.

[9] P. Stone and M. Veloso. Using decision tree confidence factors for multiagent control. Technical report, Computer Science Department, Carnegie Mellon University, Pittsburgh, 1998.

[10] P. Stone and M. Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241–273, 1999.

[11] P. Stone and M. Veloso. Team-partitioned, opaque-transition reinforcement learning. In M. Asada and H. Kitano, editors, *RoboCup-98: Robot Soccer World Cup II.* Springer Verlag. Berlin, 1999.

[12] M. Veloso, P. Stone, K. Han, and S. Achim. Prediction, behaviors, and collaboration in a team of robotic soccer agents. Technical report, Computer Science Department, Carnegie Mellon University, Pittsburgh, 1998.

[13] L.A. Zadeh. The concept of linguistic variable and its applications to approximate reasoning. *Parte I Information Sciences vol. 8, pag. 199-249, Parte II Information Sciences vol. 8, pag. 301-357, Parte III Information Sciences vol. 9, pag. 43-80*, 1975.

Agentes básicos de locomoción para un robot en exteriores

L. García-Pérez, M.C. García-Alegre, J.M. Cañas(2), A. Ribeiro, and D. Guinea

[1] Instituto de Automática Industrial (CSIC)
Arganda del Rey (Madrid)
e-mail: lia@iai.csic.es
[2] Escuela Superior de Ciencias Experimentales y Tecnología
Universidad Rey Juan Carlos
Móstoles (Madrid)

Resumen Este trabajo describe la etapa de automatización básica de un vehículo cortacésped para dotarle de la capacidad de percepción y actuación necesaria para la realización de movimientos elementales de forma semi-autónoma. La integración de sensores se realiza de forma gradual ante los requerimientos de percepción/acción necesarios para alcanzar una determinada funcionalidad. El Agente físico, posee una arquitectura de control diseñada según las pautas de la arquitectura AMARA, modelo jerárquico de múltiples Agentes de comportamiento.

1 Introducción

La Robótica se enfrenta actualmente al reto de diseñar e implementar prototipos de vehículos autónomos o semi-autónomos para realizar tareas que impliquen alto riesgo, repetitividad o un desarrollo ante condiciones climáticas adversas. Sin embargo, los avances en este campo no han sido hasta la fecha espectaculares debido a que la automatización de un robot para exteriores es una tarea compleja que requiere conocimientos de un equipo de expertos en disciplinas que van desde la Mecánica, Electrónica, Tratamiento de Señales, Inteligencia Artificial, hasta las Ciencias Cognitivas o Sociales. A pesar de ello, en los últimos años se vienen dedicando cada vez más recursos a la automatización progresiva de vehículos en exteriores, a fin de disponer en las próximas décadas de sistemas que naveguen tanto en modo teleoperado como semi-autónomo bajo control remoto. Las áreas que han experimentado un mayor auge son las relacionadas con el transporte de pasajeros o carga ya sea terrestre [1], marítimo o aéreo [2].

Sin embargo el transvase de estos conocimientos a otros campos, como es el caso de la Agricultura, se ha visto relegado fundamentalmente por razones de tipo económico relativas al valor añadido del producto a desarrollar. En la actualidad y debido a la drástica reducción experimentada por los sistemas sensoriales y de proceso, unido al creciente apogeo de las tecnologías de la información y las comunicaciones, asistimos al despegue de los sistemas semi-autónomos o autónomos para ayuda y sustitución de los operadores humanos. Algunas de las áreas que actualmente se encuentran más interesadas por estos avances son las de minería, ingeniería forestal, medioambiente, agricultura, horticultura o jardinería [3], [4], [5], y las relacionadas con las exploraciones en zonas heladas, desérticas o planetarias [6], [7].

Este trabajo se enmarca dentro del campo de la robótica móvil autónoma para la realización de tareas en exteriores y tiene como objetivo la automatización de un vehículo o tractor cortacésped, bajo un modelo de múltiples Agentes para su aplicación futura en tareas de fumigación. A continuación se presenta la etapa de automatización con la implementación de los Agentes básicos de locomoción Gira_Volante, Pisa_Pedal.

2 Robot-tractor *Rojo*

El vehículo automatizado es un cortacésped convencional, de dimensiones 2.0 (m)x 0.7 (m) con tracción a dos ruedas y motor de explosión alimentado con gasolina, de bajo coste y fácil acceso para la sustitución y manipulación de los elementos de control del giro y del embrague/marcha, Figura 1.

Figura1. Robot-tractor *Rojo*.

2.1 Sistema de actuación

Una vez analizados los posibles sistemas de actuación, se decidió la incorporación al robot de dos sistemas de actuación neumática [8], uno para el control del ángulo de giro de las ruedas delanteras y otro para el pedal del embrague.

El sistema de actuación neumática consta de los siguientes elementos: una válvula electroneumática todo/nada que permite el paso de un determinado volumen de aire comprimido, proporcional al tiempo de apertura de la misma y un cilindro neumático con un sistema de acoplamiento del pistón al actuador físico. Ambos sistemas comparten un compresor y un calderín para la generación y almacenamiento del aire comprimido, respectivamente.

Cuando la válvula se encuentra en un estado tal que permite el paso de corriente a través de su bobina, se desplaza una de las compuertas, permitiendo la salida del aire a presión procedente del calderín. Por cada cilindro de doble efecto se necesitan dos electroválvulas, una para cada cámara. Al ser la electroválvula un dispositivo todo/nada no es posible un control de la posición del pistón en función del voltaje, por ello la actuación se lleva a cabo mediante modulación en anchura de pulso: PWM (Pulse Width Modulation) [9].

2.2 Sistema de percepción básico

La sensorización del Agente físico es fundamental en el desarrollo de sistemas autónomos, ya que proporciona las claves o patrones que dirigen y modulan el sistema de control y permiten evaluar el grado de consecución del objetivo propuesto. Los sensores instalados hasta el momento en el robot-tractor *Rojo* van dirigidos a cubrir las siguientes etapas:

1. Cierre del ciclo de control de los actuadores de Giro y Marcha
2. Estimación de la posición del robot
3. Adquisición de información de estructuras del entorno

En este trabajo se presenta el desarrollo de la primera tarea. Para la automatización del robot tractor *Rojo* la mínima información requerida para realimentar el ciclo de control de la actuación es la posición del pistón dentro del cilindro. Para medir esta posición se han incorporado dos sensores potenciométricos lineales solidarios con el mismo. El pistón en su movimiento arrastra el vástago del sensor, permitiendo así medir la posición en la que éste se encuentra. Estos sensores son resistencias variables de gran calidad.

2.3 Sistema de proceso y control

Tanto la acción como la percepción tienen como soporte un procesador K6 II a 233 (Mhz.)sobre PC convencional. Esto permite disponer, de forma directa, de un conjunto de funcionalidades añadidas, tales como la comunicación, necesaria para el control remoto, y la visualización, de gran utilidad en la etapa de desarrollo del

primer prototipo de control. En la implementación de la arquitectura de control se han tenido en cuenta propiedades como modularidad, facilidad de crecimiento y ampliación del sistema perceptual y de conductas, prevención y tratamiento de posibles errores y posibilidad de modulación.

3 Control del movimiento básico de ROJO.

El movimiento del robot-tractor *Rojo* se controla mediante el giro del volante que determina el ángulo de las ruedas y el pedal embrague que actuando sobre la transmisión hace que el tractor se mueva/pare si el pedal está suelto/presionado. Aunque el vehículo disponía inicialmente de un pedal de aceleración, las restricciones de carga y las limitaciones en la automatización llevaron a la propuesta de movimiento a velocidad constante.

Por ello se han diseñado unicamente dos controladores para el movimiento del robot-tractor, denominados: Gira_Volante y Pisa_Pedal. Estos Agentes básicos de movimiento, Figura 2, se han diseñado siguiendo las premisas de la arquitectura de control AMARA [10] . Poseen una ventana espacio temporal limitada, con un ciclo de razonamiento y control de $T = 250(ms)$,y manejan un numero de variables muy reducido siendo siempre activados por Agentes de nivel superior, como pueden ser Ve_A_ Posición o Bordea_Zona_Verde. Constituyen la base de toda la futura arquitectura de planificación y control de navegación del robot.

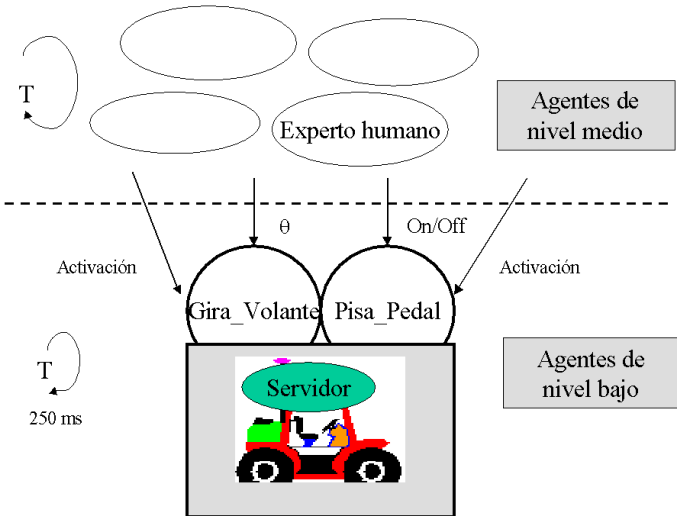


Figura2. Esquema de los módulos de bajo nivel.

3.1 Gira_Volante

El Agente Gira_Volante tiene como misión el control del actuador de la dirección del tractor. Los sistemas electroneumáticos están recomendados en aplicaciones de mo-

derada carga y moderada precisión. Son sistemas limpios, pues las posibles pérdidas son de aire y proporcionan una buena relación potencia/coste. Sin embargo poseen la desventaja de ser difíciles de controlar debido a las fugas de aire y al limitado conocimiento de las interacciones entre los distintos parámetros del complejo sistema de actuación [8]. A esta fuente de incertidumbre se une aquella derivada del propio sistema de dirección del tractor por las holguras en la unión del actuador con el eje. La gran dificultad en el desarrollo de un modelo analítico clásico ha llevado a la propuesta de un modelado fuzzy en el cuál mediante un conjunto de heurísticos intuitivos, se engloba el conocimiento de control de un experto sobre todo el sistema de dirección. Se ha diseñado un controlador PD borroso con un número reducido de variables y de reglas [9], que presenta un funcionamiento suave en todo el espacio de posibles estados del sistema.

3.2 Pisa_Pedal

El Agente Pisa_Pedal implementa un controlador de la parada/marcha del tractor. Posee una salida binaria pues sólo existen dos posibles posiciones: pedal_suelto que permite el movimiento del tractor a una determinada velocidad, en este caso 0.4m/s, y pedal_presionado que detiene al tractor. La salida del controlador a la válvula electroneumática es una señal de apertura de una u otra cámara, que presiona y mantiene el pedal a fondo.

4 Control remoto del robot *ROJO*.

El desarrollo de un robot autónomo requiere un gran esfuerzo, incluso en sus etapas básicas. La investigación en telerobótica pretende disminuir esfuerzos y costes, haciendo que el operador humano actue únicamente cuando sea necesario. Se trata de extender las capacidades de manipulación y percepción humanas a escenarios inaccesibles y/o incómodos, dejando cierta autonomía local al robot para reducir al máximo la intervención humana [11]. El control remoto puede verse como un fin en sí mismo o como un paso intermedio en la consecución de la autonomía. Todo los procesos se han desarrollado en C++ Borland Builder y el interfaz de comunicación está basado en WinSockets. En el estado actual de la arquitectura, existe un único Agente Cliente/Humano que dirige el movimiento elemental de Avance (dirección, distancia) y de Marcha/Parada (0,1) del robot. Su comunicación se realiza mediante paso de mensajes predefinidos.

El diseño del inicio de la arquitectura de control se ha guiado por los principios de: -Modularidad, de modo que sea sencillo incorporar nuevas funcionalidades - Accesibilidad, al tratarse de un robot móvil es fundamental el acceso remoto a la información suministrada por el dispositivo sensorial - Reusabilidad de los recursos, que permita reutilizar Agentes básicos en el diseño de futuros Agentes, para el desarrollo de una arquitectura de varios niveles [12]. Un Agente se define como un proceso dirigido a alcanzar o mantener un objetivo encapsulado en su propio diseño.

4.1 Programa Servidor Rojo

El programa Servidor:Rojo cumple dos objetivos primordiales:

1. Adquirir todas las señales sensoriales de los dispositivos físicos para enviarlas a los clientes que las soliciten.
2. Cerrar el ciclo de control de bajo nivel del movimiento del robot.

Para llevar a cabo los dos objetivos anteriores ha de implementarlas siguientes tareas:

- Lectura de las señales sensoriales adquiridas (PCLab) y procesamiento de las mismas para extraer la información sensorial deseada. Los sensores que acceden a PCLab son: potenciómetros, odómetros e inclinómetros.
- Lectura de la señal del láser, enviada por puerto serie. Esta señal no requiere ningún tipo de proceso.
- Atención a los mensajes de los clientes.
- Interpretación de los mensajes recibidos, i.e. los mensajes de suscripción y de-suscripción de los clientes.
- Envío a cada cliente de los datos a los que está suscrito.
- Activación de los módulos Gira_Volante y Pisa_Pedal.
- Control de errores.

Cliente y Servidor se comunican mediante una serie de mensajes predefinidos, con una estructura muy similar para facilitar su proceso y interpretación, Figuras 3 y 4.

Mensajes del cliente				
Mensajes de suscripción.			**Mensajes de de-suscripción**	
Válvulas	0002		Válvulas	0102
Odometría	0003		Odometría	0103
GPS	0004		GPS	0104
Giróscopo	0005		Giróscopo	0105
Láser	0006		Láser	0106
Voltajes	0011		Voltajes	0111
Datos giro	0012		Datos giro	0112
Freno	0099			
En marcha	0098			
Mensajes especiales.				
GiraVolante	12\tGiroObjetivo\r\n			
Reset odometría	98			

Figura3. Mensajes del cliente al servidor.

4.2 Programa Cliente: Control Remoto

Bajo la denominación de programa Cliente agrupamos a todos aquellos Agentes que se comunican vía Socket con el Servidor ROJO para hacer uso de alguno de sus

Mensajes del servidor	
Control	12 \t Tpo \t GiroVolante \t Error \t ErrorAnt \t PWM \t \r\n
Válvulas	02 \t Tpo \t VálvulaDirección \t VálvulaVelocidad \t \r\n
Odometría	03 \t Tpo \t x \t y \t Theta \t Desplazamiento \t \r\n
GPS	04 \t Tpo \t Lon \t Lat \t Alt \t NumSat \t Modo \t \r\n
Giróscopo	05 \t Tpo \t Theta \t \r\n
Láser	06 \t Tpo \t M0 \t M1 \t ... M360 \t \r\n
Voltajes	11 \t Tpo \t v0 \t v1 \t v2 \t v3 \t v4 \t v5 \t \r\n

Figura4. Mensajes del servidor al cliente.

servicios [13]. Estos Agentes o programas Cliente están especialmente diseñados para la realización eficaz de una determinada tarea, por ejemplo Evitar_Obstáculos. Sin embargo existen determinados procesos que van a ser comunes a todos los programas Cliente:

- Establecimiento de comunicación
- Envío de las solicitudes de suscripción y de-suscripción
- Lectura de los datos que envía el servidor.

Control remoto El Agente Control remoto simula un joystick y es capaz de guiar al tractor como si fuese un coche teledirigido. Es manejado sobre la pantalla por un operador humano con la finalidad de desplazar el tractor de una nave a otra en el IAI-CSIC. Este Cliente consta de dos partes bien diferenciadas: 1.-La de visualización de la posición del robot mediante la información de posición proporcionada por los odómetros y 2.- El joystick para control remoto del volante del vehículo, Figura 5.

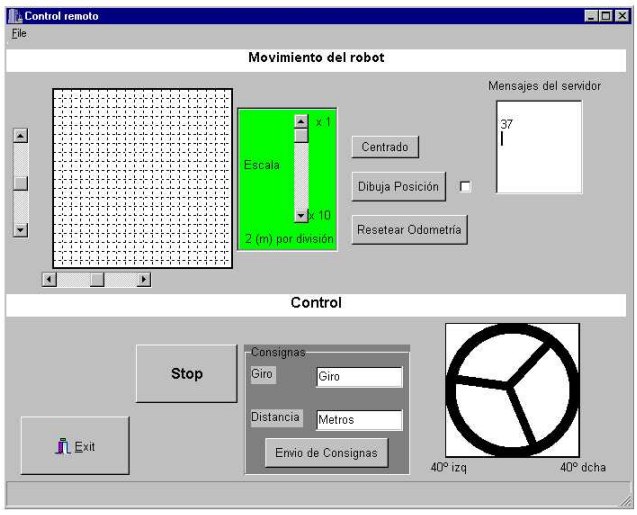


Figura5. Programa Cliente: Control Remoto.

La parte de visualización se encarga de representar la trayectoria seguida por el robot a través de la posición relativa proporcionada por los odómetros. La informa-

ción se envía via Radio-Ethernet desde el robot hasta cualquier nodo de proceso de la red interna del IAI, previa solicitud de la misma.

La parte de control consta de un joystick y de un botón de Parada/Marcha del tractor. Cuando el botón se encuentra pulsado/levantado se envía la señal de activación al Agente Pisa_Pedal con la consigna de Parada/Marcha. A través de la pantalla del joystick se envian las consignas de giro del volante, seleccionadas por el operario humano, al Agente Gira_Volante del tractor.

5 Conclusiones

Este trabajo describe la automatización de un robot móvil para la navegación en exteriores. Su automatización permitirá, en un futuro próximo, la realización autónoma o semi-autónoma de tareas rutinarias o peligrosas para el operario humano, como el corte del césped o la fumigación de frutales u olivos.

Se ha diseñado y demostrado una arquitectura con dos Agentes de bajo nivel para el movimiento del robot, que servirán como bloques básicos en una arquitectura MultiAgente para la generación de comportamiento cada vez más complejo. Con esta finalidad se ha implementado una arquitectura Cliente-Servidor que gestiona la comunicación entre los dispositivos físicos del tractor y los diferentes Agentes.

Se han realizado pruebas de movimiento teledirigido del tractor en el recinto del IAI-CSIC para su traslado de una nave a otra, que muestran un control suave y una adaptación rápida y sin oscilaciones a los cambios de dirección.

Agradecimientos

Este trabajo ha sido financiado en su totalidad por los proyectos de investigación: CICYT-TAP98-0781 "Arquitectura multiagente: Generación de comportamiento complejo para un robot de pulverización en exteriores", CICYT-TIC99-1321-CE "Supervisión inteligente del transporte multimodal de mercancias"y el Ministerio de Ciencia y Tecnología mediante becas predoctorales.

Referencias

[1] Ernst D. Dickmanns. Vehicles capable of dynamic vision: a new breed of technical beings? *Artifical Intelligence*, 103:49–76, 1998.

[2] Michio Sugeno and Hung T. Nguyen. *Fuzzy modeling and control: selected works of M. Sugeno.* CRC Press, 1 edition, March 1999.

[3] A. Mandow, J. M. Gómez-de Gabriel, J. L. Martínez, V.F. Muñoz, A. Ollero, and A. García-Cerezo. The autonomous mobile robot aurora for greenhouse operation. *IEEE Robotics and Automation Magazine*, pages 19–28, 1996.

[4] N.Noguchi and H. Terano. Path planning of an agricultural mobile robot by neural network and genetic algorithm. *Computers and Electronics in Agriculture*, 18:187–204, 1997.

[5] N. D. Tillet, John A. Marchant, and Tony Hague. Autonomous plant scale crop protection. In *Proceedings of the AGENG 96*, Madrid, Spain, 1996.

[6] C. Stocker. The search for life in mars: The role of rovers. *Journal of geophysical research*, 103:28557–28575, 1998.

[7] W. Whittaker, D. Bapna, M. W. Maimone, and E. Rollins. Atacama desert treck: Aplanetary analog field experiment. Technical report, Canergie Mellow University, 1998.

[8] Britt Rorabaugh. *Mechanical devices for the electronics experimenter.* Library of Congress Cataloging-in-Publication Data. TAB Books, 1995.

[9] Lia Garcia, Jose Maria Cañas, Maria C. Garcia-Alegre, Pablo Yañez, and Domingo Guinea. Fuzzy control of an electropneumatic actuator. In *Proceedings of the X Congreso Español sobre tecnologías y lógica fuzzy STYLF2000*, pages 133–138, Sevilla, Spain, 2000.

[10] Maria C. Garcia-Alegre and Domingo Guinea. Building and arquitecture for a farming robot. In *Bio-Robotics 97. International Workshop on Robotics and Automated Machinery for bio-Productions*, pages 255–260, Gandia, Spain, 1997.

[11] James Trevelyan. Simplifying robotics- a challenge for research. *Robotics and Autonomous Systems*, 21:207–220, 1997.

[12] Geroge A. Bekey Editor. *Autonomous Agents.* Kluwer Academic Publishers, Boston, 1998.

[13] Jose María Cañas. Desarrollo de un sistema cliente/servidor para el robot móvil hermes. Technical Report TR-06/99, Dpto. Sistemas. IAI-CSIC, June 1999.

Inclusión de capacidades específicas para la evaluación de la eficiencia de acciones físicas

Albert Oller[1] and Josep Lluis de la Rosa[2]

[1] Robòtica i Visió Intel·ligents (RIVI)
Universitat Rovira i Virgili (Tarragona).
[2] Institut d'Informàtica i Aplicacions (IIiA)
Universitat de Girona.

Resumen En este trabajo se analiza el problema de la influencia de la dinámica de los robots en la decisión de sus acciones. Se parte de la idea que la tecnología agente no tiene cuenta que las decisones de los agentes deben tomarse teniendo en cuenta que el robot tiene un cuerpo físico, es decir, una dinámica que en muchos casos limita la buena ejecución de la acción correspondiente. A partir de un modelo BDI del agente, se propone la inclusión de capacidades que tengan que cuantifiquen la dinámica del robot a partir de una caracterización del comportamiento de los controladores que contiene el robot. Los resultados que se muestran se basan en la plataforma RoboCup abordándose el problema de la situación de juego de 'el pase de la pelota'.

1 Introducción

La robótica móvil hace referencia al campo de aplicación de la robótica donde la característica esencial de los robots es su capacidad de motricidad autónoma. Dicha motricidad permite al robot la realización de desplazamientos en entornos más o menos estructurados y le obliga a equiparse de una sensorización esencial para captar el estado del entorno. El estudio de sistemas multi-robot es una extensión natural de los sistemas mono-robot, y es también es un área de investigación por sí misma cuando el trabajo en equipo de diversos robots es la única via de solución de un problema. En el área de la robótica móvil hay tres problemáticas generales que son: el control del movimiento del robot móvil como entidad individual, el control de un sistema compuesto por diversos robots (la cooperación), y la planificación de las acciones a hacer en función de restricciones temporales y espaciales.

1.1 El control del movimiento del robot móvil

Supongamos que queremos que un robot móvil siga una determinada trayectoria. A partir de la posición inicial del robot y de su modelo podemos calcular las velocidades lineal y angular que necesitamos para hacerlo. Ahora bien, una implementación real siempre presenta problemas de ruido y de incertidumbres lo que obliga a cerrar el lazo, es decir, a recalcular las velocidades mediante una ley de control. Dicha ley de control debe garantizar el buen funcionamiento del sistema a partir de un conjunto de especificaciones (características transitorias y estacionarias). La geometria de los robots móviles suele introducir limitaciones no-holónomas, lo cual introduce un

elemento añadido de complicación. En el caso de un sistema holónomo siempre es posible calcular el conjunto de coordenadas generalizadas independientes y, con ellas, cualquier movimiento arbitrario siempre es realizable. En el caso de un sistema no holónomo dicho conjunto de coordenadas generalizadas independientes no existe y, por tanto, no siempre será posible ejecutar cualquier movimiento. En definitiva, los algoritmos de planificación de trayectorias para robots móviles son complejos puesto que deben tener en cuenta muchas restricciones.

1.2 Del paradigma GOFAIR a los agentes situados

Uno de los problemas más importantes que se ha encontrado la IA trabajando con sistemas robóticos ha sido el paradigma GOFAI (*Good Old Fashion Artificial Intelligence*). Éste término fue introducido por Haugeland [Haugeland, 85] para referirse a la metodología de la IA basada en la manipulación de símbolos: "la inteligencia se corresponde con el razonamiento y el razonamiento está basado en la manipulación de reglas y con estructuras simbólicas". La forma como un sistema GOFAI percibe y actúa sobre el mundo tiene una importancia secundaria y queda relegada a módulos específicos. Aplicado a la robótica, GOFAIR (*Good Old Fashion Artificial Intelligence and Robotics*) caracteriza la idea de construir sistemas robóticos haciendo una descomposición funcional en tres módulos: percepción, razonamiento y actuación. En líneas generales, los supuestos con que se trabaja en sistemas GOFAIR son [Mackworth, 93]:

- Cualquier cosa que sea útil para un agente es descriptible.
- Las creencias que tiene un agente sobre el mundo son ciertas y justificadas.
- El conocimiento que tiene el agente del mundo está definido y es positivo.
- El conocimiento que tiene el agente del mundo es completo. El agente adquiere el conocimiento de cualquier cosa relevante que pasa en el mundo. Esto permite al agente inferir que algo es falso si no puede deducir que es cierto.
- El entorno es estático a no ser que el agente lo modifique.
- Sólo hay un agente en el mundo.
- El agente puede predecir todos los efectos de sus acciones en el mundo.
- Las acciones son discretas y se ejecutan de forma secuenciada.

Estos supuestos vienen de postular que todos los efectos de una acción son perfectamente conocidos antes que la acción se lleve a cabo. En estas condiciones, un plan consiste en una lista de acciones que si se ejecutan llevarán al mundo al estado deseado porqué existe un modelo del mundo que se corresponde con la realidad y porqué no hay más factores que intervengan en la evolución del mundo. Evidentmente un plan de este tipo en un entorno real no funcionaría: uno de los elementos de los entornos reales más contrario a los supuestos de los sistemea GOFAIR es la presencia de incertidumbres.

La metodología de programar la percepción, el razonamiento y la actuación en módulos diferentes no és escalable [Brooks, 88] y cabe buscar estrategias reduccionistas en forma de estructuras jerárquicas a fin de obtener una integración cognitiva.

Dicha búsqueda ha motivado la aparición de la llamada *nouvelle IA* en la que el diseño de los agentes se hace de manera *ad hoc*, es decir, el agente consiste en un sistema físico real desarrollado para trabajar en un mundo real y que actua y reacciona en tiempo real. Son los llamados agentes situados (*situated agents*): estos agentes pueden trabajar en entornos multi-agente y en mundos dinámicos gracias a una percepción dinámica [Mackworth, 93]. En este sentido, se franquean todos los supuestos de los sistemas GOFAIR.

La integración cognitiva obliga a diseñar y a implementar los sistemas robóticos en un mismo entorno de trabajo que disponga de herramientas para trabajar con metodologías de la IA y con metodologías de la teoría de control. El problema de enlazar la IA con la teoría de control está en que no hay interfícies entre el alto nivel que trabaja la IA y el bajo nivel de la teoría de control. La coordinación entre estos dos niveles no se comprende completamente y ello provoca que el comportamiento de un sistema no se pueda analizar *a priori*. Para solventar este problema se ha propuesto la creación de teorías unificadas con fundamentos matemáticos y con formalismos computacionales adecuados como la propuesta en [Zhang, 94]. Otra opción es intentar mezclar el conocimiento numérico (procesado de señal, teoría de control, dinámica de sistemas, etc.) con el conocimiento simbólico (razonamiento dialéctico, autómatas, estructuras de argumentos, planificación) como la arquitectura denominada InteRRaP [Müller, 96].

La filosofía del presente trabajo se enmarca en esta segunda opción: el diseño interno del agente consiste en una estructura modular jerárquica con diferentes niveles de abstracción y el funcionamiento interno se basa en el flujo de información entre los diferentes módulos [Oller, 00a]. La información a bajo nivel relativa a las características dinámicas queda descrita mediante un conjunto específico de capacidades que se obtienen a partir de las propiedades de los controladores, y dicha información puede fluir hacia niveles más altos e influir en las etapas de decisón a alto nivel.

1.3 El dominio del fútbol robótico

Los resultados que se expondrán en este trabajo se basan en el dominio del fútbol robótico. El motivo de dicha elección se basa en que dicho dominio rompe muchas de las suposiciones de la GOFAIR y, en cambio, cumple las especificaciones para trabajar con agentes situados. El dominio del fútbol se caracteriza por las siguientes propiedades [MacWorth, 93]:

- Hay agentes neutrales, amigos y hostiles, y hay cooperación entre agentes.
- Hay interacción en tiempo real, el entorno es dinámico, real e impredictible.
- Hay criterios objetivos de *performance*.

De todas las posibles "situaciones de juego" que pueden aparecer en dicho dominio se ha optado por realizar un análisis de la situación del "pase de la pelota" puesto que presenta un grado de complejidad suficientemente elevado, más si se tienen en cuenta las dinámicas de los robots y de la pelota.

2 Los agentes físicos con dinámica propia

Los agentes que llevan a cabo tareas en entornos dinámicos multi-agente tienen que cumplir los requisitos que impone el funcionamiento en tiempo real y el mundo real. Algunos de dichos requisitos son:

- Saberse comportar de acuerdo con las circunstancias (*situated behavior*), y saber reaccionar ante sucesos imprevistos.
- Tener un comportamiento dirigido hacia un objetivo (*goal-directed behavior*) y saber seleccionar las acciones en función de dicho objetivo.
- Las acciones deben realizarse con eficiencia aún teniendo en cuenta las imposiciones restrictivas del tiempo real.
- Deben tener presente la presencia de otros agentes y comportarse de forma cooperativa.

Aunque ya hay diversos trabajos orientados a diseñar arquitecturas que integren dichos requerimientos, aún hay un problema básico no planteado: teniendo en cuenta que el término 'dinámica' se refiere tanto a rapidez de cambio del entorno como a la dinámica del propio cuerpo del agente físico, la cuestión más importante que plantea este trabajo es como puede especificarse la dinámica del propio agente físico.

2.1 El cuerpo físico del agente

Los agentes que componen un sistema multi-agente recorren a los demás para obtener su ayuda, y las preguntas que deben responderse entre ellos son del tipo: ¿a qué otro agente puedo ayudar? ¿hasta donde llegará mi compromiso hacia él? ¿bajo qué condiciones puedo hacerlo? En definitiva, cuando el agente se compromete debe saber qué implican sus compromisos y saber si puede hacerlo o no.

Para sistemas físicos reales es necesaria la introducción del concepto de "cuerpo físico del agente" puesto que para saber qué puede hacer y qué no, deberá tenerse en cuenta algún conocimiento físico del sistema (en nuestro caso del robot y del entorno). Es el denominado agente físico. Esto significa que las entradas y las salidas físicas hacia y desde el entorno tienen que estar integradas en la base de conocimiento de cada agente, y esto es así porque cada agente está contenido en un cuerpo físico que debe controlar y mover según sus decisiones.

En los trabajos actuales con robots reals no se evalua la problematica asociada a la dinámica del propio robot; lo único que hacen es interpretar los problemas que aparecen desde el punto de vista de la supervisión o de la detección de errores. Ahora bien, si tenemos en cuenta que los movimientos de los robots se pueden describir con ecuacions diferenciales, podemos concluir que la teoría de control es imprescindible para el análisis de la dinámica de los agentes físicos. Aunque la decisión del agente físico sea la correcta, no tiene en consideración el hecho que el agente debe ser capaz de controlar su propio cuerpo físico: tiene que realizar cualquier tipo de tarea a pesar de los continuos cambios (discretos o continuos) que se produzcan en el entorno o a causa de sus propios movimientos o acciones.

En definitiva, el conocimiento físico se obtiene a partir de la dinámica del cuerpo físico y que se representa de forma declarativa a través de las capacidades [Oller, 00b].

2.2 Las capacidades del agente

¿Cómo pueden definirse las capacidades de un agente? Todo depende del nivel de abstracción en que se trabaje. La definición se basa en un lenguaje descriptivo numérico empezando por los niveles de abstracción baja e ir aumentando dicho grado para ir pasando a descripciones más simbólicas. Una razón de peso para comenzar a bajo nivel está en el hecho que todo agente físico dispone de unos controladores que lo limitan y/o capacitan para ejecutar una cierta cantidad de acciones de una forma más o menos correcta. De esta manera se pretende que el mundo lógico del agente tenga conocimiento sobre si está capacitado para controlar su cuerpo físico y así asumir un rol concreto para la comunidad.

Los capacidades deben definirse en los siguientes niveles:

Capacidad a nivel de control: consiste en ser la percepción de las cualidades de un controlador. Por tanto, una capacidad integra tanto la identificación del cuerpo físico como la percepción del entorno a través del cuerpo. Las capacidades a nivel de control se denominan capacidades atómicas, y son aquellas que la acción asociada puede ejecutarse directamente con la percepción y la actuación del nivel más bajo, y sin necesidad de interpretar los datos adquiridos. La acción se llevará a cabo a partir de la medida de las coordenadas generalizadas del propio cuerpo y de los eventos y las magnitudes externas al cuerpos medidas con los sensores del propio agente. Así, la definición de las capacidades atómicas se hace observando la cualidad de la ejecución de estas acciones con los diferentes controladores implementados en el cuerpo fisico, es decir, midiendo los errores en estado estacionario, el tiempo de respuesta, la amplitud de oscilación, etc. Debe notarse que el grado de sensorización es indicativo del grado de autonomía del agente, es decir, tendrá más capacidades y, por tanto, será capaz de realizar más acciones o de realizarlas de maneras diferentes.

Capacidad a nivel de supervisor: se denominan capacidades básicas, y son aquellas que se obtienen combinando las atómicas. El método de combinación consiste en utilizar las capacidades atómicas como estados en una máquina de estados finitos, y en utilizar las condiciones que obliguen a abandonar un controlador y escoger otro como transiciones. El métode de combinación consiste en construir un sistema de control híbrido asociado a cada acción. Puesto que el lenguaje descriptivo no es tan numérico como en el nivel de control, la evaluación de la cualidad de ejecución de las acciones (precisión y/o dificultad) puede hacerse con métodos heurísticos.

Capacidad a nivel de agente: se denominan capacidades simbólicas, y se definen de forma más simbólica ya que el modelo del mundo es mucho más abstracto y es completamente dependiente de la aplicación concreta. El método de definición es completamente escalable al que se utiliza en el supervisor, es decir, las capacidades en el nivel agente se construyen a partir de las capacidades básicas. El objetivo del

método de combinación no es construir un sistema de control híbrido, sino construir una máquina de estados finitos en que los estados son acciones básicas y las transiciones son las condiciones que favorecen o no el mantenimiento de la ejecución de una acción básica concreta.

3 El fútbol robótico: el pase de la pelota

En esta sección se analizarà el problema del 'pase de la pelota' teniendo en cuenta las siguientes consideraciones:

- Es una acción coordinada y cooperativa entre dos robots: el pasador y el receptor
- El pasador realiza un chut con una dirección y velocidad completamente controladas
- El objetivo final del pase es que el receptor empuje la pelota hacia la portería contraria
- El receptor deberá chutar correctamente y con restricciones temporales
- La pelota tiene una dinámica de primer orden
- La dinámica del robot receptor contiene un modelo realista de los motores

La resolución del problema pasa por la evaluación de la efectividad del chut del receptor a partir de sus capacidades básicas. En concreto, se considerará que la maniobra del receptor consta de dos movimientos: un primer movimiento de aproximación a una distancia de 35cm de la pelota, y un segundo movimiento de chut de la pelota hacia a la portería. Dichos movimientos consisten en ser acciones que se analizarán previamente: el primer movimiento se realiza con la acción atómica 'Moverse hacia unas coordenadas relativas', mientras que el segundo se realiza con la acción básica 'Chutar la pelota hacia la portería'.

3.1 Características físicas de un equipo de fútbol robótico

Las dimensiones de los robots son equivalentes a las de un cilindro de 7.5cm. de diámetro y se desplazan en un entorno cerrado de 120x90cm. En este entorno hay definidas dos zonas de 30cm. de ancho que están situadas en ambos extremos del campo de juego (porterías). Los robots tienen completa autonomía motriz (dos motores c.c y tracción de tipo diferencial) y disponen de controladores que les permiten desplazarse a través del campo de juego mediante trayectorias curvilíneas a una velocidad lineal máxima de 70cm/s. Puesto que el choque entre el robot y la pelota es completamente inelástico, el chut se producirá empujando la pelota en lugar de golpearla y, por tanto, la dirección del movimiento de la pelota posterior al chut será la del robot prévia al choque.

3.2 Especificaciones de los controladores: capacidades atómicas

El estado de un móvil se define a partir del siguiente conjunto de variables:

$$S(t)=\{\ x,\ y,\ \alpha,\ V_{LIN},\ V_{ANG}\ \}$$

Teniendo en cuenta que la tracción del robot es diferencial, las velocidades de los respectivos motores pueden obtenerse a partir de las siguientes ecuaciones:

$V_X = dx/dt = \cos\alpha\cdot(V_E+V_D)/2$; $V_Y = dy/dt = \sin\alpha\cdot(V_I+V_D)/2$

$V_{ANG} = d\alpha/dt = (V_I-V_D)/2R$; $V_{LIN} = \sqrt{(V_X^{\ 2} + V_Y^{\ 2})}$

El objetivo del control del movimiento es calcular las consignas de velocidad para cada motor para posibilitar que el móvil pueda ir de un estado inicial S_0 a otro final S_1. Se trata de un sistema de control MIMO (*Multi-Input Multi-Output*) cuyas especificaciones de diseño deben tener en cuenta las propiedades de la trayectoria deseada así como las limitaciones dinámicas del robot como, por ejemplo, no superar los valores máximos de velocidad para evitar que las ruedas patinen. Teniendo en cuenta además el carácter no holónomo del móvil, el control del movimiento se plantea como un problema con un alto grado de complicación.

Para ilustrar como se obtienen las capacidades atómicas se ha optado por escoger un controlador relativamente sencillo y eficaz, y que suele aplicarse en robots reales:

$$V_{I,D} = K\cdot(K_1\cdot d \pm K_2\cdot\Delta\theta),$$

siendo d la distancia relativa al estado final, $\Delta\theta$ la orientación relativa, y K, K_1, K_2 los parámetros del controlador.

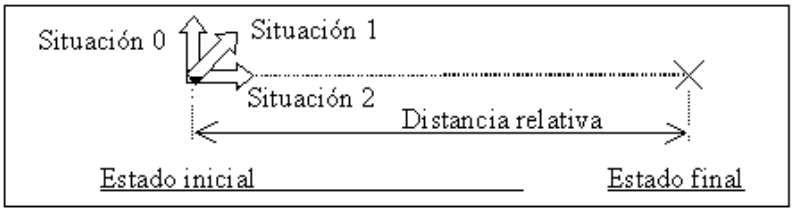


Figura1. Conjunto de posibles combinaciones de estados inicial y final

De las posibles acciones realizables por el controlador (véase tabla 3.2), sólo se analizarán sus cualidades para la acción 1 puesto que es una acción necesaria en el pase de la pelota. Para analizar dichas cualidades se han realizado ensayos para diferentes combinaciones de estados iniciales y finales según muestra la fig. 1. En la misma tabla también se muestran las capacidades que indican los tiempos de establecimiento, es decir, el tiempo que tarda el móvil en llegar al estado final.

1.Ir a una posición	2. Mantener una velocidad	3. Mantener una aceleración
4.Perseguir un objeto detectado	5. Golpear un objeto detectado	6. Evitar el choque con un objeto detectado

Tabla1. Acciones realizables por el controlador

Distancia relativa / Situación	0	1	2
35	1.25	1.15	1.1
40	1.4	1.3	1.3
55	1.95	1.6	1.55
70	2.2	-	1.85
85	2.35	-	2.1
105	2.6	5.1	2.45

Tabla2. Capacidades para la acción 'Ir a una posición' (tiempo en segundos, K=1

3.3 Especificaciones de las tareas: capacidades básicas

El cálculo de las capacidades básicas se realiza a partir de las atómicas introduciendo información simbólica. Los símbolos son 'pelota', 'contrario', 'pared', 'portería contraria', 'porteria local', etc., y algunas de las tareas (acciones a nivel supervisor) son:

1. Ir hacia la pelota
2. Ir hacia la portería local
3. Ir hacia la portería contraria
4. Evitar choques
5. Chutar pelota hacia la porteria
6. Chutar pelota hacia un punto
7. Interceptar la pelota

Algunas de estas tareas necesitan una estructura de control específica mientras que otras funcionan con la misma. Por ejemplo, las tareas 1, 2, 3 y 4 pueden funcionar con la estructura de control equivalente a 'Ir de A a B siguiendo una trayectoria libre de colisión', y las tareas 5, 6 y 7 funcionan con la equivalente a 'Ir al punto B con dirección final específica': todo depende del símbolo del punto final B. A continuación se comentará en detalle la estructura de control 'Ir al punto B con dirección final específica' puesto que es una tarea muy importante en el problema del pase de la pelota. En concreto, se considerará que el punto B corresponde al símbolo 'pelota'.

Ir al punto B con dirección final específica Las dos condiciones para obtener un chut adecuado son: estar detrás de la pelota y estar orientado correctamente para poder empujarla hacia la portería. Dado que la posición relativa inicial entre el móvil y la pelota puede variar, se ha optado por construir una máquina de estados finitos que asigna consignas de posición al móvil a medida que la posición relativa a la pelota varía: las transiciones se sucederán a medida que el móvil vaya desplazándose a través de las zonas que se muestran en la parte izquierda de la fig. 2.

Esta máquina sólo posiciona el móvil por detrás de la pelota. Para cumplir con la condición de la orientación, la zona circular cercana a la pelota consta de cuatro zonas tal como muestra la parte derecha de la fig. 2.

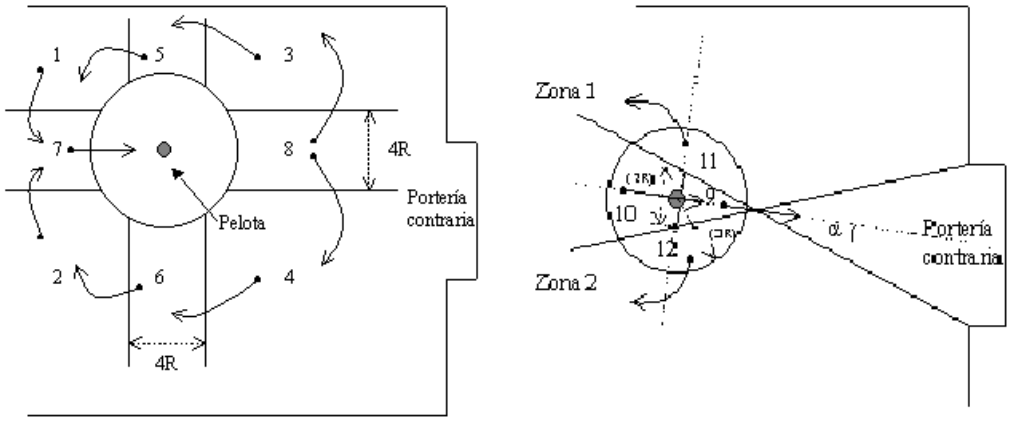


Figura2. Esquema para la asignación de consignas de posición según la posición actual del robot

Se han definido cinco posiciones relativas de test (**A**, **B**, **C**, **D** y **E**), y para cada posición se han definido diferentes estados iniciales para la orientación del robot (ver fig. 4, izda.).

Para la obtención de la tabla de capacidades se ha tenido en cuenta el cálculo de los siguientes parámetros de la respuesta del sistema: el tiempo de llegada a la pelota (T_ch en segundos), el error de orientación del robot en el momento del chut ($\Delta\alpha$ en grados) y el error frontal del choque entre el robot y la pelota (ΔW en centímetros). A continuación se muestra la tabla de capacidades para el estado '10' correspondiente a la situación **D**:

K=1				K=2			K=3		
T_ch	$\Delta\alpha$	Δ	W	T_ch	$\Delta\alpha$	ΔW	T_ch	$\Delta\alpha$	ΔW
2.3	2.5	4.7		1.7	0.4	5.2	1.7	7.2	4.9

Tabla3. Tabla de Capacidades

Puede deducirse que la localización en el estado final no es exacta a causa de las restricciones no holónomas. Aparece un error frontal ΔW del orden de 5cm, es decir, que la maniobra del robot no es adecuada para poder empujar la pelota ya que la zona frontal del robot es de 7.5cm, valor inferior a los 10cm necesarios para poder chutar. Por tanto, el estado '10' no es apropiado para ejecutar la tarea de 'Chutar la pelota hacia la portería': el robot no tendrá capacidad para hacerlo.

Las capacidades para al estado inicial '7' (fig. 4, dcha.) indican que los valores de ΔW son inferiores a 3cm, es decir, en todos los casos es posible realizar el chut. Sin embargo, aparece un valor de $\Delta\alpha$ cuya importancia dependerá de la distancia de la pelota a la portería. Es decir, para evaluar la capacidad de la ejecución de la tarea 'Chutar la pelota hacia la portería' deberán tenerse en cuenta diversos parámetros de la tabla de capacidades.també depèn del valor de la distància de la pilota a la

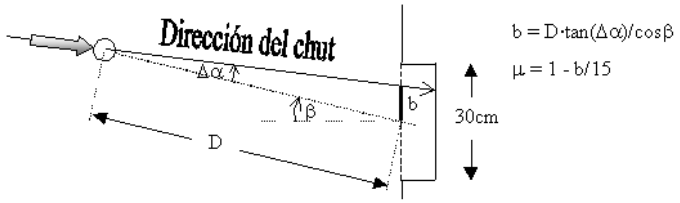


Figura3. Ilustración de los diversos parámetros que influyen en el cálculo del índice μ

porteria. Como ejemplo, la figura 3 muestra una situación de chut a una distancia D de la portería y con un valor $\Delta\alpha$¿0. En este caso, la pelota entra en la portería a **b** cm. del centro: dicha desviación respecto del centro permite hacer un cálculo de un índice de la precisión (μ) del chut.

3.4 Especificaciones de los roles: capacidades agente

Las acciones a nivel agente son roles que pueden incluir movimientos coordinados o cooperativos con otros agentes (escenas) cuya eficiencia debe calcularse a partir de las capacidades simbólicas. Para la escena 'Pase de la pelota' hay dos agentes con dos roles asignados: un agente pasador y otro receptor. Puesto que se ha supuesto que el agente pasador realiza el chut en una dirección y velocidad completamente controladas, el análisis de la efectividad del chut consistirá en evaluar la efectividad de la ejecución del chut por parte del agente receptor.

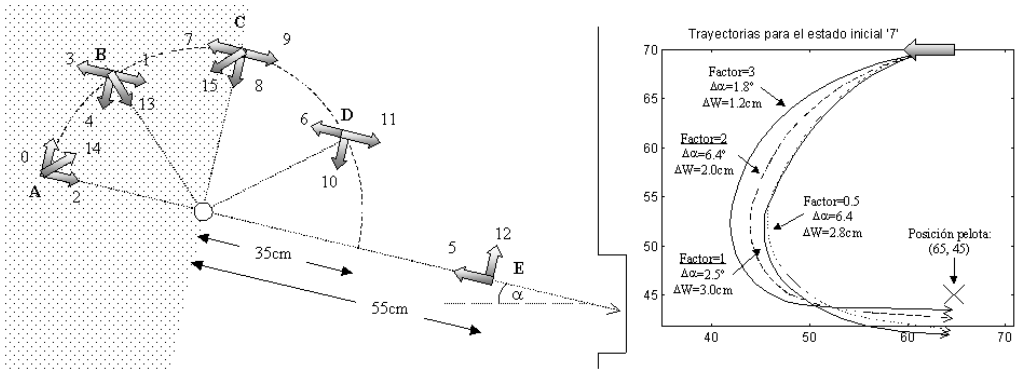


Figura4. Esquema de los estados iniciales para 'Ir al punto B con dirección final específica

Se ha considerado que la maniobra de recepción de la pelota consta de dos partes (ver fig. 5): una primera de acercamiento a 35cm de la pelota, y una segunda de chut hacia la portería.

Analizando los valores de μ a partir de las tablas de capacidades para la maniobra del chut (figura 3) se deduce que sólo en los casos **A**, **B, B', C** y **C'** su ejecución

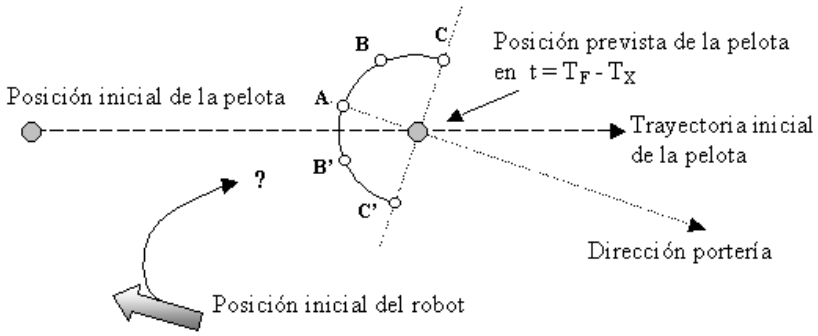


Figura5. Maniobras para la recepción de la pelota

es satisfactoria. La forma de calcular cual esos puntos será el escogido dependrá de que los tiempos de maniobra cumplan la siguiente restricción temporal:

$$T_{recepcion} \geq T_{acercamiento} + T_{chut}$$

A fin de optimizar la efectividad del chut (mínimo valor de $\Delta\alpha$), el mejor punto siempre será el **A**. Así, antes de escoger otro punto se calcula si la maniobra hacia **A** cumple la restricció temporal, en caso contrario, se intenta el punto **B** (**B'**) y posteriormente el **C** (**C'**). Independientemente del punto final **P** que se escoja, con un razonamiento por casos se minimizará la suma de los tiempos de ejecución de ambas maniobras.

En estado de cosas, el robot pasadordeberá especificar cual es la trayectoria de la pelota, su velocidad inicial y el instante de tiempo en que debe realizarse el chut ($T_{recepcion}$). El receptor evaluará los valores de $T_{acercamiento}$ y T_{chut} consultando sus capacidades. En caso que no pueda cumplir la restricción temporal, los dos robots deberán negociar algun parámetro del movimiento de la pelota: velocidad, trayectoria o $T_{recepcion}$.

Ejemplos A continuación de mostrarán dos pares de ejemplos de maniobras de recepción de la pelota, y se analizan los efectos de la restricción temporal. La fig. 6 muestra las trayectorias del primer par.

Sin restricción temporal se obtiene $T_{recepcion}$=2.6s, $\Delta\alpha$=0.37 y μ=0.6, mientras que en el caso con restricción $T_{recepcion}$=3.9s, $\Delta\alpha$=0.8 y μ=0.93. En este segundo caso, el razonamiento por casos escoge el punto **B'** y obtiene a partir de las tablas que T_{chut}=1.8s y $T_{acercamiento}$=2s, es decir, calcula que $T_{recepcion}$=3.8s. Es decir, a partir de las tablas de capacidades básiques es posible realizar cálculos aproximados del $T_{recepcion}$ y , por tanto, se dispone se suficiente conocimiento para saber si el robot receptor tiene tiempo para realizar su rol. Aunque el valor de μ presenta más imprecisiones (a partir de la tablas μ=0.36) puede utilizarse como índice de la efectividad de la ejecución de la acción.

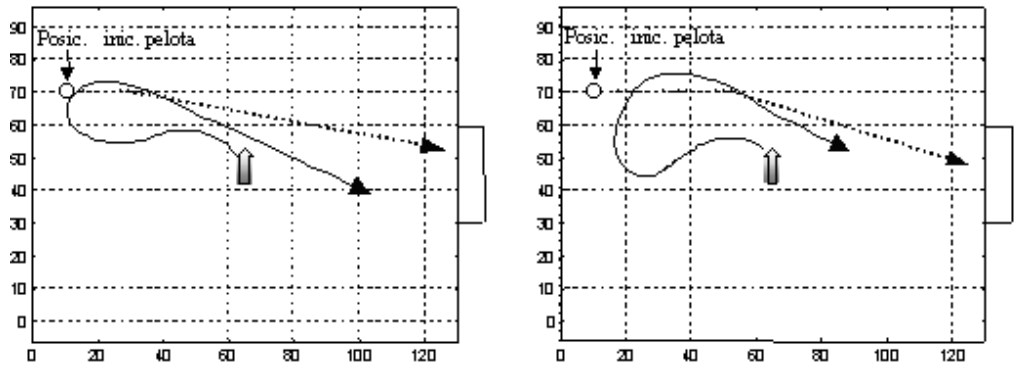


Figura6. Sin restricción temporal (izquierda), con la restricción $T_{recepcion}$=4s (derecha). Ambos con una velocidad inicial de la pelota de 10cm/s.

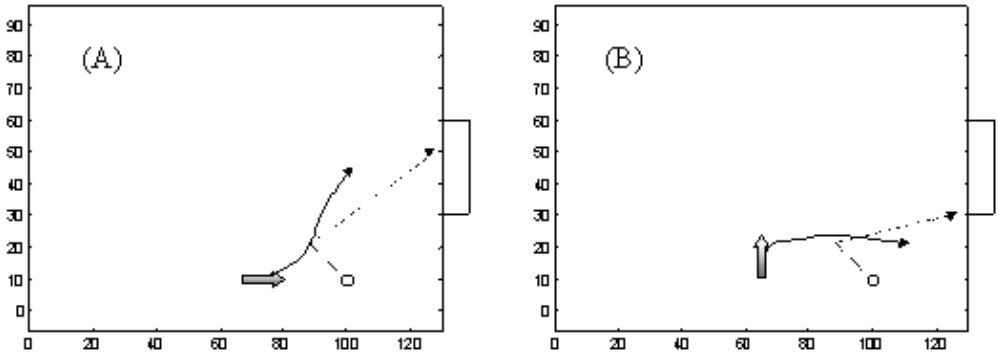


Figura7. Maniobras de recepción de la pelota en posiciones forzadas

En el segundo par (ver fig. 7)se muestran dos ejemplos más y puede apreciarse en el caso (A) la maniobra de chut se realiza completamente y se consigue un chut correcto. En cambio, en el (B) el chut no es correcto ya que la maniobra no puede finalizarse. Los valores de μ que se obtienen son los siguientes:

	Caso (A)	Caso (B)
Valores exactos	0.57	0.25
A partir de las tablas	0.73 $\Delta\alpha$=11.7 ; b=4/15	0.45 $\Delta\alpha$=71 ; b=8/15

Los valores son relativamente altos ya que se han escogido unos estados en que el chut se hace en una zona muy cercana a la portería contraria. Cabe comentar que, al contrario del ejemplo anterior, estos valores son superiores a los reales pero

ateniéndonos al uso de μ, puede concluirse que su utilidad para la evaluación de la eficiencia de la acción del chut es aceptable.

Referencias

[Brooks, 88] R.A.Brooks. A robot that walks: emergent behaviour from a carefully evolved network. Cambridge , MA: MIT. (1988)
[Haugeland, 85] J.Haugeland, Artificial Intelligence: The Very Idea. Cambridge, MA: MIT Press. (1985)
[Mackworth, 93] A.K.MackWorth. On Seeing Robots. Computer Vision: Systems, Theory, and Application. pp 1-13, World Scientific Press, Singapore. (1993)
[Müller, 96] J.P.Müller. The Design of Intelligent Agents. A Layered Approach. Lecture Notes in Artificial Intelligence, Vol. 1177. Ed. Springer-Verlag. (1996)
[Oller, 00a] A.Oller, J.Ll. de la Rosa, R. García, J.A. Ramon, A. Figueras. Micro-Robots Playing Soccer Games: a Real Implementation Based on a Multi-Agent Decision-Making Structure. International Journal on Intelligent Control and Soft Computing. Special Issue on Soccer Robotics: Micro-Robot World Cup Soccer Tournament '97, Vol. 6, No 1, 2000. Autosoft Press. (2000)
[Oller, 00b] A.Oller, J.Ll. de la Rosa. DPA2: Una Arquitectura para agentes físicos dinámicos. Actas del primer Workshop Hispano-Luso en Agentes Físicos (WAF'00). pp. 25-43. Tarragona, 12-13 de Septiembre. (2000)
[Zang, 94] Y.Zang and A.K.Mackworth. Will the robot do the right thing? Proc. Artificial Intelligence'94. Pp. 255-262,Banff, Alberta. (1994)

Sesión 5: Docencia

Robótica móvil: Recursos para la docencia

O. Déniz and A. Falcón

Universidad de Las Palmas de Gran Canaria
Departamento de Informática y Sistemas
Edificio de Informática y Matemáticas
Campus Universitario de Tafira
35017 Las Palmas - SPAIN
e-mail: {odeniz,afalcon}@dis.ulpgc.es

Resumen Cada vez son más las posibilidades y recursos con los que cuenta el profesorado universitario para impartir conocimientos en temas de contenido disciplinar en continua fase de cambio y adaptación como es el caso de la Robótica Móvil. Son ya muy pocos los centros que no disponen de alguna plataforma móvil más o menos elaborada. En este trabajo presentamos las experiencias derivadas de la impartición de dos cursos de robótica móvil, uno de ellos desarrollado en un ámbito no universitario. En particular, se describen y analizan las posibilidades y limitaciones de dos productos hardware, el kit Lego MindStorms y la plataforma Pioneer 2-DX de ActivMedia Robotics, así como uno software, el entorno Saphira, utilizados en el contexto de las actividades prácticas de los mencionados cursos.

1 Introducción

En los últimos tiempos ha aumentado significativamente el número de recursos a la disposición del profesorado universitario para el desempeño de las actividades prácticas asociadas a los contenidos teóricos. En el caso de la robótica móvil, muchos centros cuentan ya con plataformas de diseño relativamente elaborado. Estos recursos, antes dedicados casi exclusivamente a tareas de investigación, de forma natural pueden ahora utilizarse en la propia docencia, la cual mejora en calidad.

En este trabajo se exponen las experiencias e ideas extraídas a raíz de la impartición por los autores de dos cursos de Robótica Móvil. Uno de los cursos constituye la asignatura "Sistemas Robóticos Móviles" de la titulación de Ingeniería Informática, en la Universidad de Las Palmas de Gran Canaria, la cual tiene carácter optativo en cuarto curso de carrera [16]. El otro, que contó con la colaboración de la Consejería de Educación, Cultura y Deportes del Gobierno de Canarias y el Museo Elder de la Ciencia y la Tecnología, se impartió a profesores de enseñanza secundaria con el fin de iniciar la introducción de la robótica móvil en la Enseñanza Secundaria Obligatoria [1].

En la sección 2 se describen y analizan las posibilidades del producto Lego(R) Mindstorms^{TM} [2] en el contexto reseñado. En la sección 2.1 indicamos diversas posibi-

[1] La concepción de este curso seminal tiene como origen trasladar conceptos, principios e ideas a los niveles inferiores del sistema educativo de forma que se generen aptitudes y vocaciones en el ámbito de la Robótica Móvil.

[2] Lego(R) es una marca registrada del grupo LEGO.

lidades de ampliación del producto estándar, tanto a nivel hardware como software. Las actividades prácticas propuestas en los cursos se describen someramente en la sección 2.2. En la sección 3 se analiza el entorno Saphira de desarrollo y simulación de aplicaciones de robótica móvil, así como la plataforma móvil Pioneer de ActivMedia Robotics, a la hora de estudiar experimentalmente sistemas basados en comportamientos. Por último, las conclusiones más importantes se resumen en la sección 4.

2 Lego(R) Mindstorms^{TM}

El producto comercial Lego(R) Mindstorms^{TM} [6, 18, 19, 23] se presenta como un juguete de coste relativamente medio-alto. El producto responde a las expectativas creadas en este mercado en los últimos años. En un contexto menos lúdico, las características del mismo permiten aplicar hasta cierto punto aspectos prácticos de la robótica móvil, lo cual lo convierte en una herramienta atractiva para la docencia. Prueba de ello son los diversos centros que ya hacen uso de él [25, 5, 4, 2, 3].

El elemento central de Lego(R) Mindstorms^{TM} es el llamado RCX, que contiene el microcontrolador. El RCX se alimenta con pilas, y alrededor de él se construye el robot con piezas Lego(R) tradicionales. El RCX dispone de tres salidas PWM y de 3 entradas a las cuales podemos conectar diversos sensores, incluyendo el kit dos sensores de contacto y un sensor de infrarrojos. El RCX dispone también de un elemento de comunicación por infrarrojos. Lo abierto de las posibilidades de diseño hacen este kit particularmente atractivo (figura 1).

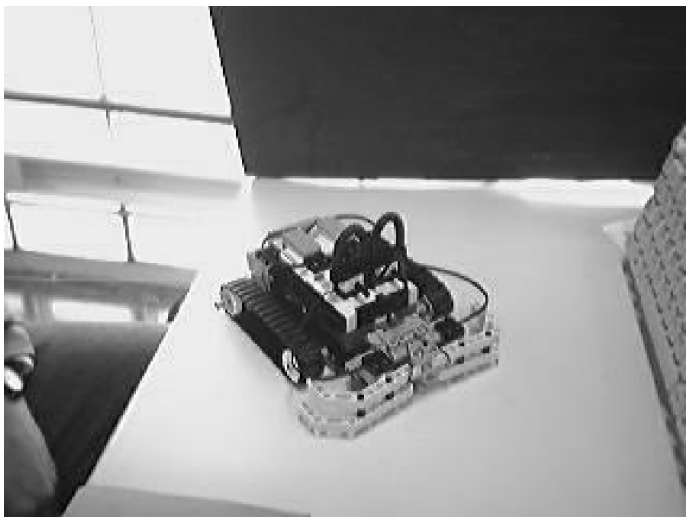


Figura1. Ejemplo de plataforma móvil simple con los sensores del kit.

Lego(R) Mindstorms^{TM} es un producto versátil basado en el microcontrolador H8/3292. Aunque con ciertas limitaciones en la programación y el hardware acce-

sible externamente, representa una evolución de conceptos similares utilizados con anterioridad, como es el caso del Brick desarrollado en el MIT [11].

La programación del RCX se realiza en un PC. El programa binario se transfiere al RCX mediante una torre emisora/receptora de infrarrojos, que se conecta a un puerto serie. El producto se entrega con un CDROM que contiene dos entornos de programación distintos. Uno de ellos es enormemente sencillo, pues permite diseñar el programa en base a bloques gráficos. Sin embargo, para ciertas aplicaciones resulta más flexible programar en el lenguaje NQC [13]. NQC es un lenguaje de sintaxis sencilla, similar al C, que cuenta con un excelente editor, el RCX Command Center [7]. La mayoría de las limitaciones del lenguaje vienen impuestas por el propio hardware del RCX. Entre ellas hay que destacar la falta de números en coma flotante [3]. Por otro lado, la sencillez del lenguaje permite implementar fácilmente comportamientos reactivos básicos. También se puede hacer uso de multitarea, lo cual facilita enormemente la escritura de ciertas aplicaciones de naturaleza muy reactiva [22]. Al utilizar pilas, el robot es autónomo, y puede comunicarse por infrarrojos con el PC o con otro robot. Esto último permite la posibilidad de implementar estrategias básicas de comportamientos de grupo [17, 26], o de realizar la computación en el PC. Existe un control OCX que puede utilizarse con los lenguajes de programación visual más comunes para comunicarse directamente con el RCX [23].

El sensor de infrarrojos permite implementar aplicaciones sencillas e ilustrativas como seguimiento de líneas, búsqueda de zonas iluminadas o, utilizado conjuntamente con la ventana de comunicación por infrarrojos del RCX, construir un detector de obstáculos.

A pesar de las limitaciones indicadas, creemos que el producto es apropiado para el desarrollo de una parte de las actividades prácticas asociadas a los contenidos teóricos de una asignatura de introducción a la robótica móvil. Una de las mayores ventajas del producto la representan sus posibilidades de ampliación.

2.1 Posibilidades de ampliación

Las posibilidades de ampliación del Lego(R) Mindstorms^{TM} son sustantivas, lo cual ha sido uno de los factores determinantes a la hora de considerarlo positivamente.

Hardware Las posibilidades de ampliación hardware podemos dividirlas en dos: productos comerciales de Lego(R) y ampliaciones de tipo "hágalo usted mismo". De un lado se dispone del *Ultimate Accesory Kit* de Lego(R). Este producto contiene un sensor de rotación de 16 pasos por vuelta (que pueden convertirse en más según la relación de engranajes que se utilicen), un mando a distancia, un sensor de contacto y piezas adicionales. Con sensores de rotación, que también pueden adquirirse por separado, es posible realizar aplicaciones con realimentación del movimiento ("cerrar

[3] Las únicas variables que existen son los enteros de 16 bits con signo. Además, solo se permite usar un máximo de 32 variables.

el bucle"), así como poner en práctica nociones básicas de odometría para una plataforma diferencial. En particular, una de las prácticas asignadas en el curso consiste en realizar el experimento del cuadrado bidireccional [20] y comprobar la reducción de los errores de odometría. Otro producto Lego(R) de expansión lo constituye el *Vision Command*. Este producto incluye una cámara USB que puede captar secuencias a 30 frames por segundo. Con *Vision Command* pueden ponerse en práctica aplicaciones básicas de visión artificial, como detección de zonas de color, etc. No obstante, su uso parece no estar muy extendido aún, y la calidad del proceso es en ciertos aspectos deficiente, en comparación con otros productos de bajo coste en el mercado.

Existen numerosas ampliaciones al hardware del RCX que puede construirse uno mismo, entre las que se incluyen diseños para sensores de rotación, temperatura, etc [10]. Una de las limitaciones del RCX, el tener solo 3 puertos de entrada, puede evitarse fácilmente de varias formas. Una posibilidad es conectar más de un sensor de contacto al mismo puerto de entrada, e incluso conectar un sensor de luz y uno de contacto a un puerto. Para más sensores, existen varias alternativas (ver [10]), como la de construir un multiplexor en el tiempo, como el que mostramos en la figura 2. Su funcionamiento se basa en la utilización de uno de los puertos de salida del RCX, y en la capacidad de éstos de adoptar tres estados distintos: polaridad directa, inversa, y desconectado. Con este simple diseño se pueden conectar hasta tres sensores distintos a una entrada del RCX. Cambiando periódicamente el estado del puerto de salida se consigue conectar los distintos sensores al puerto de entrada. En [14] se describe un programa para LegOS (ver sección 2.1) que controla el multiplexor. El multiplexor expuesto destaca por su simplicidad de montaje y bajo coste.

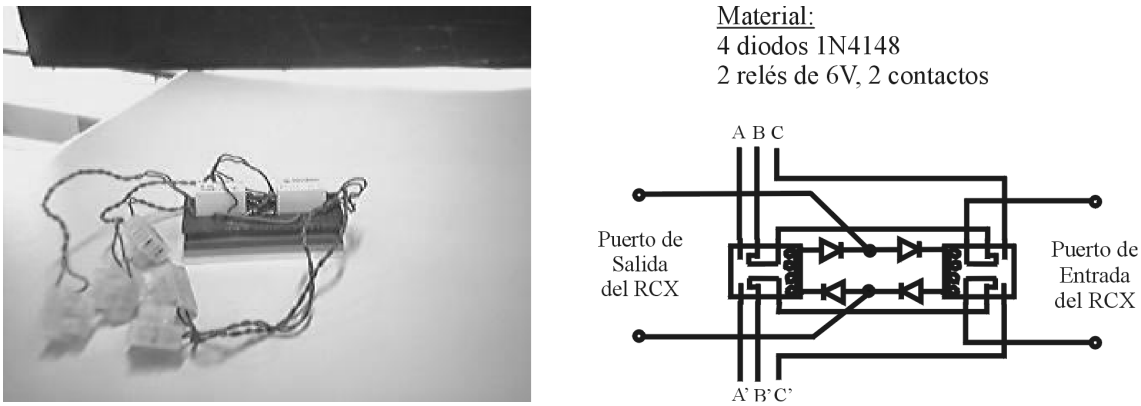


Figura2. Multiplexor de entradas. El que se muestra en la figura utiliza como elementos básicos dos relés y cuatro diodos. El circuito se cubre con piezas de Lego para un mejor montaje.

Software Al igual que en el caso de ampliaciones hardware, existe una gran variedad de herramientas software adicionales, la mayoría de ellas de libre distribución.

Es destacable la gran variedad de lenguajes de programación que pueden utilizarse: FORTH, Smalltalk, Java, etc. También hay que destacar la gran cantidad de programas ejemplo y aplicaciones disponibles en código fuente.

Probablemente el esfuerzo más importante de ampliación software del RCX es LegOS [8]. LegOS es un sistema operativo para el RCX, diferente del original de Lego(R). La programación para LegOS se realiza en C estándar. Muchas de las limitaciones anteriormente mencionadas desaparecen en LegOS. Entre sus características más atractivas cabe destacar: números en coma flotante, número no limitado de variables, semáforos POSIX y un mayor control sobre el hardware del RCX. Con LegOS se ha podido realizar aplicaciones de control con redes neuronales sencillas [12]. La desventaja fundamental de LegOS estriba en que no es aún un producto muy elaborado, lo cual hace que puedan encontrarse "bugs".

2.2 Prácticas diseñadas

En el contexto de los cursos reseñados, el diseño de las prácticas con el kit Lego(R) Mindstorms^{TM} se orientó a alumnos cuyos conocimientos en cinemática y dinámica son escasos. Las actividades prácticas diseñadas fueron las siguientes:

- Construcción de un vehículo tipo tanque con dos sensores de contacto y uno de luz y desarrollo de un programa NQC para seguimiento de líneas y evitación de obstáculos. Se construye un circuito con cinta adhesiva negra.
- Desarrollo de un programa NQC que ante la detección de un obstáculo asuma que se ha colisionado con otro robot e inicie un intercambio de mensajes a través del puerto IR para establecer un protocolo de continuación de movimientos.
- Desarrollo de un programa en NQC o para LegOS para hacer que el robot se acerque a una fuente luminosa, a la vez que evite obstáculos.
- Desarrollo de un vehículo de Braitenberg [21] que cumpla los requisitos de la práctica anterior. La computación necesaria se realiza mediante un perceptrón.
- Realización del test del cuadrado bidireccional [20]. El test permite corregir errores de odometría sistemáticos. En este caso se hace uso de los sensores de rotación ya mencionados, uno por cada rueda.

Con el fin de ilustrar la diferencia entre robots móviles con ruedas y robots con patas, en el curso impartido a profesores de enseñanza secundaria se construyó un robot móvil con 6 patas no independientes. El diseño de este robot puede consultarse en [9].

Con respecto a las prácticas en la asignatura "Sistemas Robóticos Móviles", se formaron 5 grupos, cada uno compuesto de 4 alumnos. Cada grupo se hizo cargo de un kit, responsabilizándose (a principios de curso) por escrito de cuidar todas su piezas y las pilas entregadas. Es necesario señalar que, si bien son comparativamente más caras, las pilas recargables resultan imprescindibles. La actividad práctica de la asignatura correspondía a un total de 30 horas, a razón de dos horas semanales. Las únicas características requeridas por el laboratorio son ordenadores PC con un puerto serie libre, mesas amplias y cargadores de pilas.

Los criterios de evaluación utilizados en la asignatura fueron:

- Un cuaderno de experimentos, donde se recojan los diseños, planteamientos, problemas y estrategias adoptadas, con un recorrido cronológico de la actividad realizada.
- Vídeo explicativo que muestre el funcionamiento correcto del robot móvil según las prácticas propuestas. Con el fin de grabar las secuencias de vídeo se instaló una *Web Cam* en el laboratorio de prácticas.
- Memoria sintética del trabajo (Descripción del hardware final, tareas que lleva a cabo el robot, software documentado...)

3 Saphira

Saphira [15] es un entorno de desarrollo y simulación de aplicaciones de robótica móvil. Saphira ya ha sido utilizado con éxito como herramienta educativa en el ámbito universitario [1] y en el investigador. Existen versiones de libre distribución, para diversos sistemas operativos, con la única limitación de no poder usarlas con un robot físico. Para poder utilizar Saphira para desarrollar aplicaciones es necesario tener instalado algún compilador de C. Creemos que las posibilidades del simulación de Saphira, así como la posibilidad de utilizar su librería C para crear aplicaciones complejas lo hacen un recurso de gran valor en el contexto que nos atañe. Esta librería permite utilizar funciones de lectura de sonars, sensores de contacto y un sistema de visión opcional. Saphira permite simular y ejecutar las aplicaciones desarrolladas de forma transparente.

Existe una versión multiagente de Saphira, capaz de controlar varios robots. Utiliza la arquitectura *Open Agent Architecture* (de libre distribución para propósitos no comerciales) que, entre otras posibilidades, permite el control de los agentes a través de Internet.

En el ámbito de los cursos impartidos por los autores, se utilizó Saphira para desarrollar actividades prácticas relacionadas con el control borroso de robots móviles y los sistemas basados en conductas. Entre otras posibilidades, Saphira permite crear comportamientos y combinarlos mediante reglas borrosas para dar lugar a actividades más complejas [28, 24]. Además, Saphira incluye determinados comportamientos básicos, como detección de obstáculos o movimiento a velocidad constante. Los comportamientos se escriben en una mezcla de C y un lenguaje de especificación de comportamientos borrosos. Saphira guarda las lecturas de los sensores en un modelo de los datos llamado LPS (*Local Perceptual Space*), a partir del cual se computan determinadas variables difusas a utilizar en los comportamientos. Las reglas difusas hacen uso de estas variables para dar como resultado un conjunto difuso de control de los motores del robot. Por último, este conjunto se *defuzzifica* para obtener un valor concreto a enviar a los motores [27].

Al tratarse de un curso introductorio se propuso una actividad práctica muy sencilla que consistía en hacer que el robot merodease y cuando encontrase un obstáculo

(utilizando los sonars) empezara a *orbitarlo*, esto es, a girar a su alrededor a una distancia relativamente fija. Una vez diseñados los programas, se procede a su simulación en Saphira y, una vez comprobada su validez, se ejecutó con una plataforma móvil física Pioneer (figura 3). En este nivel es importante que los alumnos tengan conocimientos básicos de lógica difusa.

La plataforma diferencial Pioneer, producto comercial de ActivMedia Robotics, está preparada para funcionar con Saphira y, además de un anillo de sonars y sensores de contacto, dispone de una amplia variedad de accesorios como módem radio, cámaras, láser, etc. Es de destacar la precisión de los codificadores de las ruedas, así como su capacidad de carga.

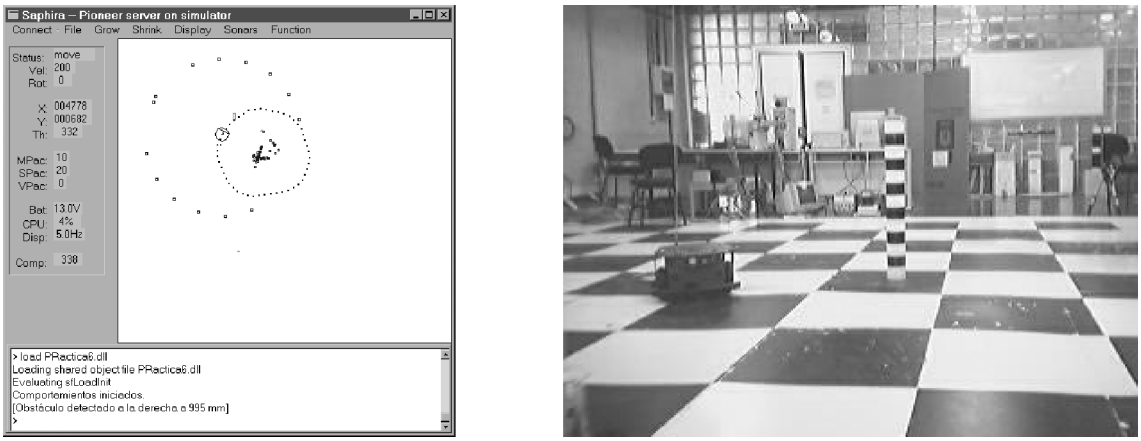


Figura3. Aplicación práctica usando control borroso desarrollada con Saphira, simulación (izquierda) y ejecución con una plataforma física (derecha).

Además de la posibilidad de escribir aplicaciones de control borroso, Saphira incluye un lenguaje llamado Colbert. Colbert es un lenguaje de control reactivo, de sintaxis parecida a la de C, que permite emplear concurrencia y el paradigma de los autómatas de estados finitos en la programación. La utilización conjunta de Saphira y una plataforma móvil permite poner en práctica conceptos más avanzados como actividades complejas, realimentación visual, construcción de mapas, etc.

4 Conclusiones

En virtud de lo expuesto en los apartados anteriores, donde se analizan las posibilidades docentes de una solución hardware y una software, se plantea la validez de dichas soluciones para el desempeño de las actividades prácticas asociadas a los contenidos teóricos de un curso universitario de introducción a la robótica móvil.

La solución hardware expuesta en la sección 2 no supondría una inversión económica excesiva. Por otro lado, como se ha explicado las limitaciones del producto pueden superarse de diversas formas. En general la respuesta de los alumnos resultó positiva, en tanto en cuanto la actividad práctica se hizo más amena. Es de destacar el

entusiasmo que el kit consiguió imprimir a los alumnos. Actualmente se está estudiando la posibilidad de diseñar actividades prácticas más elaboradas para cursos posteriores, especialmente relacionadas con comportamientos de grupo.

La solución software descrita destaca por ser de libre distribución para fines educacionales, así como por la potencia de sus capacidades de simulación. Debido a la falta de tiempo en la asignatura, no se trató con la profundidad deseada. No obstante, actualmente se utiliza como herramienta fundamental en dos Proyectos de Fin de Carrera relacionados con navegación, lo cual es a su vez un indicador de sus posibilidades.

Referencias

[1] CS224 - Real World Autonomous Systems at Stanford University. http://www.ai.sri.com/~konolige/CS224/index.html.

[2] Curso de robots LEGO en SUNY/Utica. http://gozer.sunyit.edu/classes/CSC490/csc490.html.

[3] Curso de robótica basado en LEGO en la Rice University. http://www-brazos.rice.edu/elec201/.

[4] Curso del departamento de informática de la Universidad de Utrech (Holanda). http://www.cs.uu.nl/people/markov/lego/index.html.

[5] Lego Lab en la Universidad de Aarhus (Dinamarca). http://legolab.daimi.au.dk/.

[6] The LEGO MindStorms official site. http://www.legomindstorms.com.

[7] Lego Robots: RCX Command Center. http://www.cs.uu.nl/people/markov/lego/rcxcc/.

[8] Legos Official Web Page. http://www.noga.de/legOS/.

[9] Mindstorms Info Center. http://www.mi-ra-i.com/JinSato/MindStorms/index-e.html.

[10] MindStorms RCX Sensor Input Page. http://www.plazaearth.com/usr/gasperi/lego.htm.

[11] The MIT Programmable Brick. http://lcs.www.media.mit.edu/groups/el/projects/programmable-brick/.

[12] Neural Net control in LegOS and Mindstorms. http://www-cse.ucsd.edu/~msemanko/cse151proj1/dland.html.

[13] Not Quite C. http://www.enteract.com/~dbaum/nqc/index.html.

[14] Programa de control de multplexor para LegOS. http://serdis.dis.ulpgc.es/~ii-srm/MatDocen/notas_practicas/legOS_Ejemp%los/sensor_mux.c.

[15] Saphira Robot Control System. http://www.ai.sri.com/~konolige/saphira/index.html.

[16] Sistemas Robóticos Móviles, Ingeniería Informática, Universidad de Las Palmas de Gran Canaria. http://serdis.dis.ulpgc.es/~ii-srm.

[17] R. Arkin. *Behavior-Based Robotics.* MIT Press, 1998.

[18] Dave Baum. *Dave Baum's Definitive Guide to LEGO MINDSTORMS.* APress, 1999.

[19] Dave Baum, Michael Gasperi, Ralph Hempel, and Luis Villa. *Extreme Mindstorms: an Advanced Guide to Lego Mindstorms.* APress, 2000.

[20] J. Borenstein, H.R Everett, and L. Feng. *Navigating Mobile Robots: Sensors and Techniques.* A.K. Peters, Ltd., Wellesley, MA, 1996.

[21] V. Braitenberg. *Vehicles. Experiments in Synthetic Psychology.* MIT Press, 1984.

[22] R. Brooks. *Cambrian Intelligence: The Early History of the New AI.* MIT Press, 1999.

[23] Jonathan B. Knudsen. *The Unofficial Guide to LEGO MINDSTORMS Robots.* O'Reilly, 1999.

[24] D. Kortenkamp, P. Bonasso, and R. Murphy (Eds.). *Artificial Inteligence and Mobile Robots. Case Studies of Successful Robot Systems.* MIT Press, 1998.

[25] Página de la asignatura de robótica, Ingeniería Técnica en Informática de Sistemas, Universidad Rey Juan Carlos. http://gsyc.escet.urjc.es/docencia/asignaturas/robotica/.

[26] R. Murphy. *Introduction to AI Robotics.* MIT Press, 2000.

[27] ActivMedia Robotics. *Saphira Software Manual - Saphira version 6.1.* ActivMedia Robotics, 1997.

[28] A. Saffioti, E.H. Ruspini, and K. Konolige. Blending reactivity and goal-directedness in a fuzzy controller. In *Procs. IEEE Intl. Conference on Fuzzy Systems*, pages 134–139, San Francisco, CA, 1998.

Opciones para la Programación de los Robots LEGO MindStorms(TM)

Vicente Matellán Olivera, Jesús M. González Barahona,
Pedro de las Heras Quirós, José Centeno González

e-mail: {vmo,jgb,pheras,jcenteno}@gsyc.escet.urjc.es
Departamento de Ciencias Experimentales e Ingeniería
Universidad Rey Juan Carlos

Resumen GNU/Linux sobre un ordenador personal es la opción libre preferida por muchos desarrolladores de aplicaciones, pero también es una plataforma de desarrollo muy popular para otros sistemas, incluida la programación de robots, en particular es muy adecuada para *jugar* con los LEGO Mindstorms. En este artículo se presentaremos las dos opciones más extendidas a la hora de programar estos *juguetes*: NQC y LegOS. NQC es una versión reducida de C que permite el desarrollo rápido de programas mientras que LegOS es un sistema operativo completo que permite la programación en lenguajes tradicionales como C o C++. Además, presentaremos algunas herramientas para GNU/Linux relacionadas con LegOS como simuladores o compiladores web.

1 Introducción

Antes de comenzar a analizar las herramientas de programación para los LEGO Mindstorms[1] conviene hacer una pequeña recapitulación sobre el estado de la robótica a nivel comercial de forma que se puedan comprender las razones del éxito que ha supuesto este "juguete". Se puede afirmar que el LEGO Mindstorms constituye uno de los primeros intentos de difundir los robots a gran escala. De hecho se pueden considerar equivalentes en cuanto a difusión, a lo que fue la familia de los ZX de Sinclair en su día en el mundo de los ordenadores. Al igual que ocurrió entonces, donde habían existido computadores que se vendían en forma de *kit*, como por ejemplo el MIT Altair 8800 (basado en el intel 8080) [9], han existido otros robots que han tenido difusión comercial, como por ejemplo el RugWarrior[6], quizá el más conocido. Este robot, que todavía se comercializa, está basado en el chip 68HC11 de Motorola, uno de los más extendidos en el mundo de la electrónica aplicada al control y de la robótica. Otro ejemplo también basada en el chip 68HC11 es la tarjeta HandyBoard [2], diseñada específicamente para la construcción de robots, o la tarjeta española de Microbótica (`http://www.microbotica.es`) que usa el mismo chip. De igual forma, han existido y siguen apareciendo, muchos otros robots "en *kit*" basados en este y en otros chips.

Todos estos robots comparten con los primeros ordenadores el "espíritu de garaje", es decir, el programador tiene que montar su hardware y luego hacer sus programas. Este espíritu es atractivo para muchos de nosotros, pero a la vez hace que muchos potenciales usuarios queden fuera del mundo de robótica por la complejidad y dedicación que implica. En general tener que usar un soldador, un polímetro, etc. son condiciones inaceptables para el gran público.

Por supuesto, existen robots que se venden como productos comerciales finales. Estos robots generalmente están destinados a la investigación y su precio es tan elevado (casi siempre por encima de los 6.000 Euros) que los hacen inaceptables para el gran público. Ejemplos típicos de estos robots son los de Real World Interface `http://www.irobot.com/` con robots como el B21, Magellan o la familia ATRV; o los de ActiveMedia Robotics `http://www.activrobots.com/` con productos como el Pioneer, Peoplebot, o Amigobot. En esta categoría de robots para centros de investigación también hay algunos más pequeños y asequibles como los suizos de K-Team `http://www.k-team.com` (Khepera y Koala) o el australiano Eye-Bot `http://www.ee.uwa.edu.au/~braunl/eyebot/`, pero con el coste de ser productos menos profesionales.

El siguiente estadio tras los garajes en la evolución del mercado de los robots dirigido al gran público es el paso del *kit* de auto-montaje, al producto final, el equivalente al del ZX en el mundo de los ordenadores. En este paso el caso de los LEGO Mindstorms puede resultar engañoso, de hecho es un producto comercial que precisamente está pensado para que los usuarios "construyan" sus modelos. Sin embargo, el aspecto electrónico del producto es el de producto final, es decir, no hay que soldar ningún elemento, simplemente comprar unas pilas y encenderlo. De la misma forma la parte de programación del robot es un producto final, se entrega un entorno de programación completamente visual muy orientado a los niños, que hace que cualquier usuario, sin ninguna formación informática pueda programar fácilmente el robot.

A la vez que los LEGO Mindstorms han aparecido otros robots destinados al consumo. Por ejemplo, el RL-500 `http://friendlyrobotics.com/` diseñado para cortar el césped, Cye `http://www.personalrobots.com` capaz de pasar la aspiradora o de llevar una bandeja de una habitación a otra, la mascota de Sony, el famoso perrito AIBO `http://www.sony.com/aibo`, etc. Sin embargo, ninguno de ellos ha alcanzado los miles de unidades vendidos por LEGO, ni ha conseguido la enorme atención de la comunidad del software libre que ha conseguido el MindStorms.

Una vez analizada brevemente la situación de la robótica en el aspecto comercial, la siguiente sección analizará las alternativas de programación de los Lego Mindstorms bajo GNU/Linux, que es el objetivo de este artículo.

2 La programación de los LEGO Mindstorms

LEGO Mindstorms es un producto diseñado como producto de consumo, lo que le dota de una gran calidad y además de una enorme flexibilidad. Permite construir una enorme variedad de robots, desde un brazo hasta un robot recolector, sin necesitar ninguna soldadura. Además, permite una gran libertad a la hora de colocar los sensores y los motores de los robots. El cerebro de estos robots está basado en un microprocesador Hitachi H8/300 con 32 Kbytes de RAM. Dispone de tres puertos de salida para conectar motores y tres puertos de entrada para conectar sensores, además de un puerto de infra-rojos para comunicarse con el ordenador que sirve

de plataforma de desarrollo. Los sensores disponibles son el de luz ambiente, de luz reflejada, de colisión, de temperatura y de rotación.

Los LEGO Mindstorms se venden como *kits* completos formados por más de 700 piezas tradicionales de LEGO. Cada *kit* además incluye dos motores, dos sensores de contacto, uno de luz y el bloque del micro-procesador que se denomina habitualmente "ladrillo" o RCX (con la memoria y los puertos de entrada/salida). Además, el *kit* incluye un entorno gráfico de desarrollo y el equipo necesario para descargar el software desde un ordenador personal al ladrillo.

El entorno para el desarrollo de programas incluido en el *kit* Mindstorms 1.5 es un lenguaje gráfico de programación pensado fundamentalmente para niños o adultos sin ninguna preparación informática. Se trata de un entorno muy limitado, y que además no es requisito de ser software libre. Afortunadamente, los LEGO Mindstorms han recibido mucha atención por parte de la comunidad de software libre, con lo cual han aparecido varias opciones diferentes para su programación muchas de las cuales se analizan en el libro de Jonathan B. Knudsen [7]. Entre ellas destacan dos, NQC y LegOS, que se analizan en las siguientes secciones.

NQC El acrónimo NQC [5] significa *Not Quite C*. Se trata de un simple lenguaje con una sintaxis muy similar a C que se puede usar para programar el ladrillo. Es, desde luego, la alternativa más sencilla al lenguaje de programación basado en iconos arrastrables que proporciona LEGO.

El aspecto de un programa en NQC es el siguiente:

```
/* Ejemplo en NQC */
int tiempo, giro;

task main() {
  SetSensor(SENSOR_1,SENSOR_TOUCH);
  SetSensor(SENSOR_2,SENSOR_TOUCH);
  start avanzar;

  start evitar;
}

task avanzar(){
  while(true){
    OnFwd(OUT_A+OUT_C);
  }
}

task evitar(){
  while(true) {
    if (SENSOR_1 ==1) {
```

```
        stop avanzar;
        OnRev(OUT_C); Wait(50);
        start avanzar;
    }
    if (SENSOR_2 ==1) {
        stop avanzar;
        OnRev(OUT_A); Wait(50);
        start avanzar;
    }
  }
}
```

Este programa realiza el control del robot de la Figura 1. Este robot tiene dos "palpos" que al tropezar con algún escalón presionan un sensor, detectando el peligro. El objetivo es conseguir que el robot pasee aleatoriamente por una zona, por ejemplo sobre una mesa, sin caerse por ningún posible escalón.

Figura1. Robot usado para los ejemplos

El programa consta de tres tareas, una principal que configura el sistema (indica en que puertos están conectados los sensores) y arranca las otras dos (`main`). La segunda tarea hace avanzar en línea recta al robot (`avanzar`) y la tercera detecta los choques, y según el sensor que se haya activado hace girar al robot hacia atrás durante medio segundo.

NQC es software libre, distribuido bajo la Licencia MPL (*Mozilla Public License*). Sin embargo, usa el sistema operativo original de LEGO, lo que hace que no se cumpla el requisito previamente impuesto de emplear sólo software libre en las prácticas. Sin embargo esa no es la principal razón por la que se escogió legOS. NQC se diseñó para ser muy simple y para ser utilizable por personas con limitados conocimientos de programación. Todo ello se ha conseguido, pero a costa de una serie de limitaciones, de entre las cuales las más importantes son:

- Las subrutinas no admiten parámetros, por lo que no es posible emplear realmente los principios de la programación estructurada.
- Sólo se pueden usar variables globales, es decir, el espacio de nombres es único, lo cual complica enormemente el desarrollo y mantenimiento de programas complejos.
- Las subrutinas no pueden devolver valores, lo cual las limita a ser subconjuntos de código, sin poder aportar funcionalidades nuevas ocultando su implementación.
- El número de variables está enormemente limitado, de hecho sólo se dispone de 32.
- No existen las estructuras de datos, ni las estáticas ni las dinámicas, lo que imposibilita casi cualquier tipo de aproximación que necesite almacenar cualquier tipo de estado.

A pesar de estas limitaciones, NQC es un lenguaje muy popular entre los usuarios de baja formación informática, debido sobre todo a su simplicidad, pero también a la abundante documentación y a la existencia de herramientas como el *RCX Command Center* para MS-Windows que facilitan el desarrollo y la descarga de programas. En el caso de GNU/Linux, se dispone de un compilador que produce código directamente descargable en el robot, pudiendo utilizarse como editor cualquiera que edite texto ASCII, como por ejemplo Emacs.

Sin embargo, si se quieren realizar programas que requieren cierto almacenamiento de datos, por ejemplo crear un mapa, o simplemente que sean más grandes, NQC no es opción y hay que utilizar un lenguaje de programación real (C, C++, Forth, Ada, etc.) lo cual obliga a cambiar el sistema operativo del ladrillo y en consecuencia a usar LegOS.

LegOS LegOS es un sistema operativo libre diseñado para el LEGO Mindstorms, diseñado e implementado fundamentalmente por Markus Noga [8]. Comparado con el sistema operativo original de LEGO, ofrece muchas ventajas además de mejores prestaciones y mayor flexibilidad. Entre las características más importantes de la versión legOS 0.2 se pueden destacar:

- La carga dinámica de programas y módulos.
- El protocolo de comunicación basado en el transmisor infra-rojo.
- La posibilidad de realizar programas multitarea.
- La gestión de memoria dinámica.

- La existencia de *drivers* para todos los subsistemas del ladrillo.
- El uso de la velocidad nativa del micro-procesador, esto es 16 MHz.
- El acceso a los 32K de memoria RAM.
- Permitir el uso completo del lenguaje de programación elegido, como por ejemplo C. Lo cual implica que se pueden usar punteros, estructuras de datos, etc.

LegOS es solo el sistema operativo, lo que quiere decir que se necesita el soporte de un compilador capaz de generar código para el H8 a partir del lenguaje de programación que deseemos y para el que existan las librerías adecuadas de LegOS. Para ilustrar el aspecto de dichas librerías se incluye a continuación un ejemplo del mismo programa anterior realizado en esta ocasión en C:

```
#include <conio.h>
#include <unistd.h>
#include <dsensor.h>
#include <dmotor.h>

/*Declaro una función que se ejecutará al detectar la colisión*/
wakeup_t colision(wakeup_t dato);

int main(int argc, char *argv[]) {
  int dir=0;

  while(1) {
    /* arranco
       - Fijo velocidad*/
    motor_a_speed(MAX_SPEED);
    motor_c_speed(MAX_SPEED);
    /* - Avanzo*/
    motor_a_dir(fwd);
    motor_c_dir(fwd);

    /* Espero */
    wait_event(&colision,0);
    /* Me apunto en que lado fue la colisión*/
    if(SENSOR_1<0xf000)
      dir=0;
    else
      dir=1;

    /* reculo */
    motor_a_dir(rev);
    motor_c_dir(rev);
```

```
    /* - el ratito que reculo*/
    msleep(500);

    motor_a_speed(MAX_SPEED);
    motor_c_speed(MAX_SPEED);

    /* Una vez reculado, ahora giro un pelín*/
    if(dir==1) {
      motor_c_dir(fwd);
    } else {
      motor_a_dir(fwd);
    }
    /* - ratito para girar */
    msleep(500);
  }
}

wakeup_t colision(wakeup_t dato) {
        lcd_refresh();
        /* Los sensores están conectados a los puertos 1 y 2 */
        return SENSOR_1<0xf000 || SENSOR_2<0xf000;
}
```

La estructura del programa es equivalente a la del anterior en NQC, pero en esta ocasión sólo se utiliza una tarea que espera en un bucle infinito a que se produzca un evento, la pulsación del sensor, para realizar el giro. Se han añadido suficientes comentarios al código como para que no merezca la pena entrar en más detalles de su descripción.

El entorno de programación bajo GNU/Linux incluye el compilador de C de GNU (gcc) compilado como cruzado para el Hitachi H8, para lo que hace falta usar las `binutils`. La distribución para GNU/Linux de LegOS incluye varias herramientas que permiten descargar el código de forma dinámica, descargar el *firmaware* o sistema operativo, así como varios ejemplos.

Además, alrededor del sistema operativo LegOS se han desarrollado múltiples herramientas auxiliares, como por ejemplo simuladores que hacen más fácil la depuración al permitir ejecutar programas en la propia plataforma de desarrollo usando un depurador tradicional de GNU/Linux como por ejemplo `gdb`. Algunas de estas herramientas se analizan en la próxima sección.

3 Herramientas relacionadas con legOS

Se eligió legOS como base para la programación del RCX, lo que obliga a utilizar otras herramientas, algunas evidentes como el entorno de compilación cruzado. En

este caso se ha utilizado el entorno de la FSF[1] basado en las `binutils` y el compilador `gcc`. También se han utilizado otras herramientas como los simuladores. Algunas de estas herramientas son:

3.1 LegoSim

LegoSim[3] es un simulador para legOS cuya principal virtud, que le diferencia de Emulegos (descrito en la siguiente sección), es que el interfaz gráfico de usuario (GUI) se puede separar del simulador propiamente dicho. Esta separación permite utilizar el GUI como unidad de control no sólo como simulador, permitiendo por ejemplo, conectarlo a otros RCX vía infra-rojos. Por supuesto, también permite ejecutar el GUI en una máquina distinta de la del simulador.

El GUI es un *applet* de Java que se parece realmente al RCX. El simulador es sólo una biblioteca (librería). Ambos componentes se relacionan mediante un conjunto de *scripts* en Perl. El simulador es una biblioteca que reemplaza la parte de legOS que se enlaza con cualquier aplicación en el proceso de compilación, generando una aplicación completa y ejecutable en la máquina de desarrollo. En la simulación, las tareas (*tasks*) de legOS se traducen en *threads* POSIX. Las entradas y salidas, que resultan vitales, se simulan mediante cadenas de texto sobre `stdin` y `stdout` siguiendo una sintaxis particular.

LegoSim se distribuye bajo licencia MPL, es decir, es software libre. Sus diseñadores e implementadores principales han sido Frank Mueller, Thomas Röblitz, y Oliver Bühn.

3.2 EmuLegOS

EmuLegos [4] es otro simulador de legOS. Su objetivo de diseño fue proporcionar un entorno más confortable para probar y depurar programas. EmuLegOS es, en esencia, un conjunto de código escrito en C++ que se puede compilar y enlazar junto con cualquier aplicación para legOS, generando como resultado de ese proceso una aplicación que emula el comportamiento de ese código al que tuviese si estuviese ejecutándose en un RCX real. El nivel de programación o API (*Application Program Interface*) emula las rutinas de legOS. La mayoría de legOS está implementado en EmuLegOS, incluyendo por ejemplo el soporte de tareas, o la comunicación por infra-rojos.

El aspecto externo de una aplicación para legOS ejecutando en EmuLegOS es el de la Figura 2. EmuLegOS permite al usuario configurar los sensores e interaccionar con ellos mientras el programa está en ejecución, por ejemplo simulando eventos externos. El interfaz también muestra el estado de los hasta tres motores que se pueden "pinchar virtualmente" en los puertos A, B y C del RCX.

EmuLegOS también permite que se emule el mundo real, proporcionando un lugar donde insertar código que imite algunas de las características físicas del robot.

[1] http://www.fsf.org

Figura2. Simulador Emulegos

Por ejemplo, se puede incorporar un sensor de rotación que gire mientras un determinado motor virtual está en marcha, o hacer que un sensor de colisión se active pasada una cantidad de tiempo desde que se arrancó el motor.

La mayor utilidad de los simuladores es la posibilidad de depurar. En ambos casos (EmulegOS y LegoSim), se pueden utilizar todas las herramientas disponibles en el entorno de desarrollo, ya que el programa para legOS se ejecuta dentro del simulador, y éste dentro de la plataforma.

Otra categoría de herramientas relacionadas con legOS son las que se pueden catalogar como herramientas de terceras partes. Así por ejemplo, existen compiladores para legOS accesibles vía Web, como el que se analiza en la siguiente sección.

3.3 WebLegos

Web-LegOS es un interfaz escrito en HTML para compilar programas escritos para el sistema operativo legOS del LEGO Mindstorms RCX.

El uso de Web-LegOS es sencillo bastando con cortar y pegar un fichero fuente en una caja de una página web, elegir el lenguaje de programación y seleccionar la forma en que se quiere recibir el fichero que se genera. A continuación, con pulsar el botón de "compilar" se produce el envío del fichero fuente y su compilación cruzada remota a legOS. El fichero obtenido está en formato `S-record` (un formato diseñado para para permitir la descarga de datos desde un ordenador a otro diferente). Una vez que el fichero `S-record` se ha obtenido a través del compilador web es posible descargarlo en el RCX utilizando el canal de infra-rojos habitual.

4 Conclusiones

Este artículo se centra en presentar la programación mediante software libre de los robots LEGO Mindstorms. Para ello comienza dando un repaso a la situación emergente de la robótica comercial, centrándose especialmente en el papel de los LEGO Mindstorms. A continuación se aborda su programación bajo GNU/Linux, presentando el lenguaje NQC y el sistema operativo LegOS haciendo especial énfasis

en las herramientas relacionadas utilizables bajo GNU/Linux. Como conclusión, se puede afirmar que LegOS es la opción más recomendable, en primer lugar por ser una opción completamente libre, en el caso de NQC el sistema operativo sigue siendo el propietario de LEGo. Pero además porque aporta la flexibilidad suficiente y necesaria en cualquier desarrollo. Por último, el soporte bajo GNU/Linux es el más adecuado.

Referencias

[1] LEGO Mindstorsms `http://www.mindstorms.com`
[2] Handyboard `http://el.www.media.mit.edu/projects/handy-board/`
[3] `http://www.informatik.hu-berlin.de/$\sim$mueller/legosim/`
[4] `http://www.geocities.com/$\sim$marioferrari/emulegos.html`
[5] **Dave Baum**. *Dave Baum's Definitive Guide to LEGO Mindstorms.* Apress, USA, 1999.
[6] **J. L. Jones, A. M. Flynn y B. A. Seiger**. *Mobile Robots: Inspiration to Implementation (2nd Edition).* A. K. Peters, Wellesley, Massachusetts (USA), 1998.
[7] **J. B. Knudsen**. *The Unofficial Guide to LEGO MINDSTORMS Robots.* O'Reilly & Associates, 1999.
[8] **Markus L. Noga**. *LegOS: Open-source embedded operating system for the LEGO Mindstorms.* `http://www.noga.de/legOS/`.
[9] **M. V. Wilkes**. *Computing Perspectives.* Morgan Kaufmann Publishers Inc., 1995.

Anexo: Presentaciones de Proyectos

Hardware Abierto: Microbot Tritt

Juan José San Martín et al.

Microbótica S.L.

Resumen El objetivo que persigue esta ponencia es el de plasmar de una manera objetiva los beneficios del Hardware Abierto y cómo ello ha impulsado la tecnología de los robots móviles. Con el fin de darle un eminente enfoque práctico nos basaremos en un modelo concreto, el microbot TRITT. El carácter abierto de Tritt lo ha impulsado a convertirse en uno de los microbots más ampliamente utilizados en las universidades y centros de investigación españoles.

1 Introducción

Antes de empezar, introduciremos el hardware abierto como un movimiento que trata de llevar las teorías de la Free Software Foundation, la licencia GPL y los modelos de negocio al campo del hardware. Se intenta con ello obtener en el desarrollo del hardware las mismas ventajas que en el software libre, sobre todo en lo referente a aspectos como la reutilización de código, acceso al código fuente, colaboración y cooperación y un largo y conocido etcétera.

Naturalmente, no nos embarcamos en este ambicioso objetivo en solitario, ya que el HA no es una idea nueva, sino que como otras, se viene gestando desde la aparición del modelo GPL en el campo del software. En los últimos meses se ha mantenido estrecha comunicación entre los diferentes grupos interesados en hacer realidad el HA (Open Hardware, Open Collector, GNU Projects,...) con el fin de aunar esfuerzos y centralizar el trabajo hasta ahora realizado.

Por otro lado, Microbótica S.L. es una joven empresa de ingeniería. Su actividad principal son los proyectos electrónicos y el desarrollo de software para GNU/Linux, aunque no por ello ha descuidado la actividad que la vió nacer: La Microbótica. Es

decir, el desarrollo de pequeños microbots que permiten a la gente introducirse en el terreno de la robótica profesional. La forma de hacerlo es ofreciendo una serie de kits y cursos de formación, siendo el más popular de todos ellos el microbot Tritt, el cual se puede considerar como el primer producto de HA ofrecido por una empresa española.

2 Lo importante: La información

El microbot Tritt nació en el año 1997 con un carácter pedagógico, enfocado a introducir a estudiantes, profesionales e investigadores en el universo de los microbots, y con ello al de la Microbótica. La palabra microbot proviene de la abreviatura microrobot y surge como una alternativa a los robots clásicos. Mientras que los últimos son complejos, los primeros son fáciles y simples de construir y programar. Al hablar de microbótica se está aludiendo implícitamente a un conjunto de microbots que cooperan y que interactúan los unos con los otros.

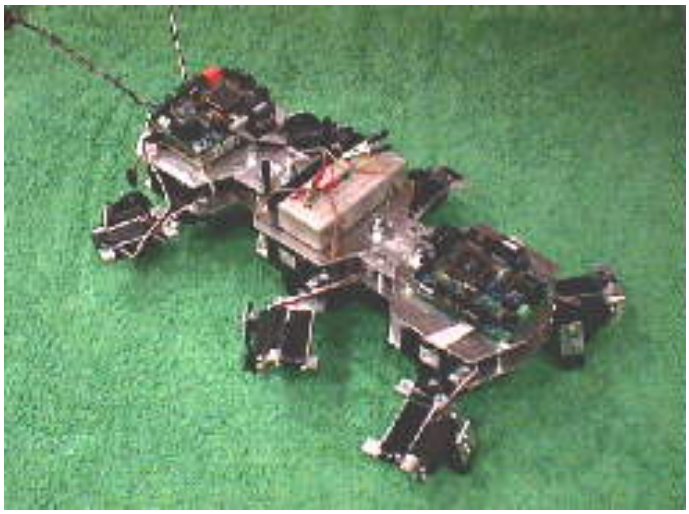


Figura1. Microbot Hexápodo controlado por dos CT6811

2.1 El inicio: Las velas

Un equipo de investigación europeo colocó en una habitación un conjunto de robots autónomos de reducidas dimensiones, los cuales disponían de una programación rígida y muy rudimentaria (Microbots reactivos). Su consigna era la de deambular por el recinto de forma aleatoria recolectando a su paso pequeños trozos de velas que habían sido esparcidos previamente. Al cabo de cierto tiempo, los robots se verían sobrecargados disparándose otra conducta reactiva: *dejar las velas recolectadas en el suelo*. Para sorpresa de muchos, al finalizar el ensayo, en la habitación quedaba

un único conjunto de velas. Este experimento fue repetido obteniéndose siempre el mismo resultado. En Microbótica nos enganchamos a esta nueva forma de pensar, la cual ve que un conjunto de robots pequeños (microbots) pueden realizar tareas complejas fruto de una colaboración en grupo, y ser más eficaces que un sólo robot complejo que centralice en él todas las funciones.

2.2 Tritt al descubierto

Es importante notar que Tritt, debido a su carácter pedagógico persigue dos objetivos principales.

- Servir de plataforma de lanzamiento e introducción a los robots móviles
- Ser flexible, expandible y modular con el fin de poder proyectar diseños más complejos sobre él.

De acuerdo con esto, el microbot Tritt puede entenderse no solamente como el primer paso para introducirse en esta tecnología, sino que también permite crecer y profundizar sobre la misma, haciendo hincapié sobre la disciplina que más interese (mecánica, sensores, control, electrónica, comunicaciones, software en general, etc...).

Una descripción breve de sus partes es la siguiente:

- La tarjeta microcontroladora CT6811, basada en el MC68HC11 y que actúa como cerebro del robot.
- Una extensión para poder mover motores y leer diversos sensores. Es la CT293+.
- Una estructura mecánica, que en este caso es de LEGO, de forma que es muy facil ampliarlo.

Como se ha dicho, Tritt ha sido diseñado para cubrir criterios de modularidad y ampliación con lo que a modo de ejemplo podemos citar:

- Tarjeta CT256 de ampliación de memorias. Capaz de contener 64Kb en dos bancos compatibles tanto con memorias RAM, EPROM, EEPROM o RAM con PILA.
- Sensores de contacto, de luz, de temperatura, de infrarrojos, etc...

Naturalmente el sistema Tritt comprende diferentes piezas de software como son ensambladores, compiladores de C, ejemplos, programas de depuración y comunicaciones, etc. También diferentes documentos técnicos como son artículos de montaje, de programación y de divulgación.

Todos los planos (de circuitos y mecánica), los programas (con licencia GPL), ejemplos así como gran cantidad de información está disponible en http://www.microbotica.es donde continuamente se reciben y publican modificaciones en el diseño, nuevos ejemplos e ideas así como proyectos completos basados en Tritt.

Microbótica comercializa Tritt con el fin de ayudar a superar el escalón que genera cualquier proceso de fabricación artesanal. De todas maneras es completamente realizable la implementación de Tritt en base a la información disponible.

La información pública es uno de los pilares sobre los que se sustenta el microbot Tritt

Figura2. Microbot TRITT

3 Libertad de movimiento

En su nacimiento, Tritt ya contaba con software de carácter libre (para plataformas LINUX y MS-DOS) y la información sobre el montaje y su estructura se podía obtener de las fotos situadas en la WEB. La liberación del hardware tardó un poco más en llegar, debido a que era necesario contar con un soporte e infraestructura que hiciera posible el compartir dicha información. El proyecto Hardware Abierto (http://www.microbotica.es/ha.htm) hizo esto posible.

La libertad y publicación de la documentación ha conseguido que el microbot sea completamente libre y esta cualidad resultó ser una ventaja competitiva. A la gente le agradó esta decisión y la han avalado aportando, a su vez, diferentes ampliaciones y trabajos realizados.

Actualmente esta forma de trabajo es nueva, por lo tanto todavía queda mucho por hacer, por ejemplo, acordar formatos, vías de comunicación, la localización de herramientas de desarrollo libres, etc... Pero a pesar de todo ya es posible observar las ventajas que aporta esta nueva forma de trabajo donde la ética, la colaboración y la cooperación forman una estructura tan firme como para levantar proyectos a la altura de GNU/Linux.

4 Conclusiones

Gracias al Hardware Abierto, Tritt está capacitado para afrontar el reto de servir tanto de plataforma de lanzamiento como base hardware para trabajar en el mundo

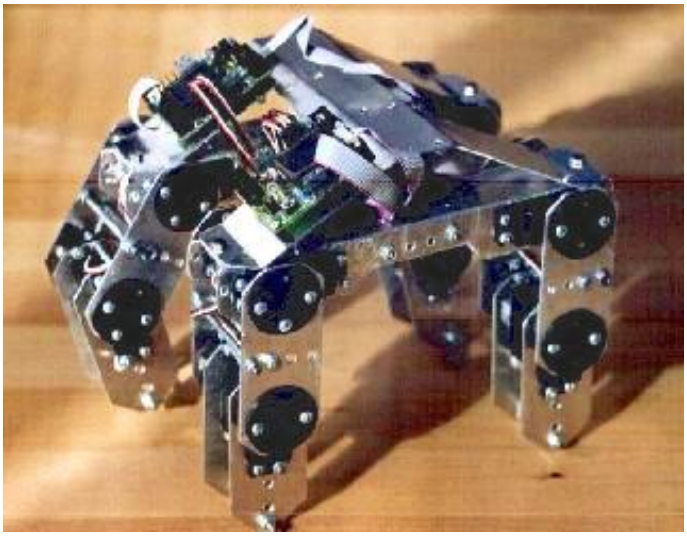


Figura3. Microbot Perro

Figura4. Arquitectura ápoda controlada con la CT6811

de los robots móviles, los microbots. Tritt recibe de la comunidad de desarrolladores de hardware abierto un gran apoyo, consiguiendo las siguientes ventajas.

1. Miles de circuitos similares con la misma funcionalidad que pueden degenerar en unos pocos, óptimos, probados y por supuesto, libres. La comunidad de desarrolladores trabaja concurrentemente sobre ideas similares que poco a poco se van fundiendo en una sola en la que se aunan las mayores virtudes.
2. La reutilización de diseños, es decir evitar "reinventar la rueda". El poder basarse en diseños abiertos para mejorarlos o adaptarlos a las necesidades, no solamente consigue optimización en cuanto a coste sino que produce una batería de diseños cada vez mayor.
3. La extensibilidad de los diseños con la aparición de extensiones y periféricos. La compatibilidad y estandarización es uno de los pilares sobre los que se sujeta el Hardware Abierto y el que ha logrado su expansión.
4. El desarrollo comunitario como metodología de cuasi-perfección.
5. La potenciación de nuevos talentos que antes podrían pasar desapercibidos.

5 Referencias

```
Microbótica : \texttt{http://www.microbotica.es}
Debian : \texttt{http://www.openhardware.org}
Sistemas empotrados : \texttt{http://www.openhardware.net}
Un procesador Libre : \texttt{http://www.techweb.com/wire/story/TWB20000228S000/}
Compartiendo Core : \texttt{http://www.openip.org/oc/index.htm}
Herramientas CAD libres : \texttt{http://www.geda.seul.org}
```

A Video Sensor Based on CORBA Architecture

Antonio Guzmán, José Pelegrin, and Enrique Cabello

Universidad Rey Juan Carlos
Escuela Superior de Ciencias Experimentales y Tecnología
C/Tulipán s/n. 28933 Móstoles (Madrid)

Abstract The demo presents a distributed computer vision application based on CORBA communication. This system supplies flexibility in image processing for several tasks with the advantage of being a distributed system. These requirements are needed, for example in face verifier. The demo will be focused in the tasks involved in an accurate analysis of face features to extract an identity.

The demo presents a distributed computer vision application based on CORBA communication. This system supplies flexibility in image processing for several tasks with the advantage of being a distributed system. These requirements are needed, for example in face verifier. The demo will be focused in the tasks involved in an accurate analysis of face features to extract an identity. This demo is related with the VISOR-BASE project. The VISOR BASE project is aimed at creating VISOR, a CORBA adaptation to the requirements of digital video monitoring systems applied to the development of observation applications with artificial vision functionality.

The VISOR architecture will include:

- A set of CORBA-compliant classes to define the VISOR interface
- A digital video object-based broker implementation

As examples of VISOR-compliant video sensors, a motion detector, a people counter, a number plate reader, a face recogniser, and a people tracker will be developed.

To validate the architecture three applications will be developed. The first application will be a basic digital video monitoring system. The second application will be a video sensor-based access control system. Finally, the third application will be an Intelligent Alarm Detection and Video indexing system for surveillance purposes.

The VISOR Architecture will include:

- A set of CORBA components, to enable the communication between video sensors in the video capturing field units and presentation and exploitation programs at remote observation centres. These components will be based on OMG micro CORBA and the Audio/Video specifications, to provide maximum interoperability at minimum footprint.
- The "VISOR Guidelines for Observation Applications development,"to help developers structure their VISOR-compliant applications.
- A digital-video optimised ORB, meeting the stringent performance requirements of digital video objects communication.

To validate the VISOR architecture and to contribute to the roll-out of a high-volume video sensor industry, the VISOR BASE project will also produce a set of VISOR-compliant video sensors oriented to the implementation of observation and surveillance services. These video sensors will provide the following functionality:

- Motion detection
- People counting
- Number plate recognition
- Face recognition
- Multi-purpose video detection and tracking of individuals

The video sensors development will focus on adaptation to the VISOR architecture and the ability to work in real world conditions, like sub-optimal lighting or camera adjustments, avoidance of algorithm training and limitations on processor performance originated by cost constraints. This approach is expected to develop products sufficiently easy to install as to be deployed in high volume in actual CCTV installations.

To complete the VISOR architecture validation and to help testing the different video sensors, the following VISOR-compliant digital video based applications will be implemented:

- A core observation system, regarded as a basic building block for any service to be built using the VISOR Architecture, including live and stored video management functionalities. In addition to being a building block for more advanced applications, this observation system will constitute a simple stand-alone application solving the basic remote surveillance scenario.
- A video sensor-based access control system, regarded as paradigm of a typical application that could benefit more from the integration of video sensing capabilities, such as face recognition, numberplate recognition and people counting.
- An intelligent Alarm Detection system to integrate the technologies developed within VISOR-BASE project and using VISOR architecture in Surveillance application of public sites (like Service stations along the Highways and Integrated control of Urban Areas). This will contribute to the realisation of a new generation of surveillance systems for a clever use of the large amount of video information usually collected and stored by conventional systems. The detection of human faces, license plate reading, people counting and behaviour classification are just a few examples of indexing features which will be used to simplify video archive and retrieval functions.

In order to disseminate the VISOR results into the market a user's group, called Videosafe User's Group, has been created. The Videosafe User's Group will benefit from:

- The strong links with the leading worldwide companies in the control and security sectors of VisualTools.

- The knowledge of the video surveillance and access control markets of Elsag and TechTalk.
- The relations with the research institutions related to artificial vision of Universidad Politecnica de Valencia and Universidad Rey Juan Carlos, and, in particular, of INRIA, as coordinator of one of the key European networks in computer vision.

Acknowlegde: This project is supported by European Union (Project: IST - 1999 - 10808 VISOR BASE). URL: `http://www.vtools.es/visorbase/`

Índice de Autores