XII Workshop of Physical Agents (WAF'2011)

Ismael García-Varea and Luis Rodríguez-Ruiz

***Abstract*—The Workshop of Physical Agents intends to be a forum for information and experience exchange in different areas regarding the concept of agent in physical environments, especially applied to the control and coordination of autonomous systems: robots, mobile robots, industrial processes or complex systems. This special issue is devoted to the selected papers presented in the WAF11 that took place in Albacete (SPAIN) from 5th to 6th of September.**

***Index Terms*—WAF, Physical Agents.**

I. The XII edition of the Workshop of Physical Agents

SINCE its first edition in 2000, the Workshop of Physical Agents has served as a meeting point for different research groups from different areas of knowledge and with a common interest: physical agents. Most of the groups that actively participate in RedAF (the Spanish Network of Physical Agents) took part in this event. RedAF is the main local sponsor organizing this Workshop. This annual event has three main objectives:

- Providing a forum for communication on research, technological innovation and technology transfer in the field of physical agents, encompassing areas such as embedded systems, artificial intelligence, mobile robotics,
- Promoting communication among researchers from different research groups related to physical agents.
- Providing a mean for active and efficient transfer of research and technological upgrading.

The XII edition was a complete success, since these goals were achieved once again. We started on September 5th, enjoying the plenary talk "Social Robots" given by Professor Carlos Balaguer from the University of Carlos III, Spain. After that, we had two working sessions on "Robots and Agents" and "Localization and Planning", consisting of four talks per session. Finally, on September 6th we attended the plenary talk "Visual Place Classification" given by Dr. Barbara Caputo from the IDIAP Research Center, Switzerland. Next, we had a working session on "Vision and Sensors", a working session on "Applications and Interaction" and, after lunch we had the demo session.

As a summary, a total of twenty papers were presented in the four working sessions. As in previous editions of the workshop, the live demos and exhibitions attracted the attention of significant media press, such as TV news and different articles in the local, regional and national newspapers, were published. We had, as well, the opportunity to interact with the mobile social robot Loky, designed by the SIMD group of the University of Castilla-La Mancha in conjuntion with the RoboLab group of the University of Caceres, and equiped with different elements developed by: Robotnik Automation, S.L., and IADex, S.L. These two companies also showed their last products and developments in their exhibition stands during the two days of the workshop. Also, we enjoyed the Nao Robots soccer team from the University of Rey Juan Carlos, as well as with some localization and mapping experiencies from the Robotnik patrol robots. Finally, in the closing ceremony the prize for the Best Paper of the Workshop was awarded for the first time. This prize was based on the criteria and judgement of all from the participants to the event.

In this special issue we include five papers about different topics in vision, HRI, localization and planning.

The first one, *Engaging human-to-robot attention using conversational gestures and lip-synchronization*, presents an algorithm for synchronizing Text-To-Speech systems with robotics mouths. The proposed approach estimates the appropriate aperture of the mouth based on the entropy of the synthetic audio stream provided by the TTS system. The authors also present the results of the opinion poll that has been conducted in order to evaluate the interaction experience.

The second one, *Obstacle Avoidance in Underwater Glider Path Planning*, presents a path planning scheme with low computational cost for underwater glider vehicles that allows static or dynamic obstacle avoidance, frequently demanded in coastal environments, with land areas, strong currents, shipping routes, etc. The method combines an initialization phase, inspired by a variant of the A* search technique and ND algorithm, with an optimization process that embraces the physical vehicle motion pattern. Consequently, the proposed method simulates a glider affected by the ocean currents, while looking for the path that optimized a given objective. According to experiments the authors carried out, this planner shows the promising results in realistic simulations, including ocean currents that vary considerably in time, and provides a superior performance over other previous approaches.

The third one, *Local robot navigation based on an active visual short-term memory*, proposes a dynamic visual memory to store the information gathered from a continuously moving camera onboard the robot and an attention system to decide where to look at with this mobile camera. The visual memory is a collection of relevant task-oriented objects and 3D segments, and its scope is wider than instantaneous field of view of the camera. The attention system takes into account the need reobserving objects in the visual memory, exploring new areas and testing cognitive hypotheses about object presency in the robot surroundings. The system has been programmed and validated on a real Pioneer robot that relies on the information from the visual memory for navigation tasks.

The fourth one, *Multi-agent system for fast deployment of*

Ismael García-Varea is with the University of Castilla-La Mancha, Spain. E-mail: Ismael.Garcia@uclm.es

Luis Rodríguez-Ruiz is with the University of Castilla-La Mancha, Spain. E-mail: Luis.RRuiz@uclm.es

a guide robot in unknown environments, describes a multi-agent intelligent space, which consists of intelligent cameras and autonomous guide robots. The deployment of the system does not require expertise and can be performed in a short period of time. The cameras detect situations where the robots guiding services are required, inform the robots accordingly, to posteriorly assist the robots navigation towards the goal areas, without the need of a map of the environment. An example of these situations requiring the robot guiding service could be a group of persons entering a museum. To this regard, the authors also discuss an adaptive person follower intended to be the basis of a route learning process, necessary to provide the guiding service.

The last paper of this issue, *Learning in real robots from environment interaction*, describes a proposal to achieve fast robot learning from its interaction with the environment. The proposal is suitable for continuous learning procedures as it tries to limit the inestability that appears every time the robot encounters a new situation it had not seen before. On the other hand, the user will not have to establish a degree of exploration (usual in reinforcement learning) and that would prevent continual learning procedures. This proposal uses an ensemble of learners able to combine dynamic programming and reinforcement learning to predict when a robot will make a mistake. This information can be used to dynamically evolve a set of control policies that determine the robot actions. This paper was the winner of the Best Paper Award of that edition of the WAF.

To conclude this introduction, we would like to thank all the authors for their excellent contributions and to extend our gratitude to all the participants of the XII Workshop of Physical Agents. We also would like to stress our gratitude to the Spanish Network of Physical Agents (http://www.redaf.es) which provides support for all the events related with physical agents in Spain.

Engaging human-to-robot attention using conversational gestures and lip-synchronization

F. Cid, L.J. Manso, L.V. Calderita A. Sánchez and P. Núñez

***Abstract*—Human-Robot Interaction (HRI) is one of the most important subfields of social robotics. In several applications, text-to-speech (TTS) techniques are used by robots to provide feedback to humans. In this respect, a natural synchronization between the synthetic voice and the mouth of the robot could contribute to improve the interaction experience. This paper presents an algorithm for synchronizing Text-To-Speech systems with robotic mouths. The proposed approach estimates the appropriate aperture of the mouth based on the entropy of the synthetic audio stream provided by the TTS system. The paper also describes the cost-efficient robotic head which has been used in the experiments and introduces the use of conversational gestures for engaging Human-Robot Interaction. The system, which has been implemented in C++ and can perform in real-time, is freely available as part of the RoboComp open-source robotics framework. Finally, the paper presents the results of the opinion poll that has been conducted in order to evaluate the interaction experience.**

***Index Terms*—Robotics head, Lip Synchronization, Human Robot Interaction.**

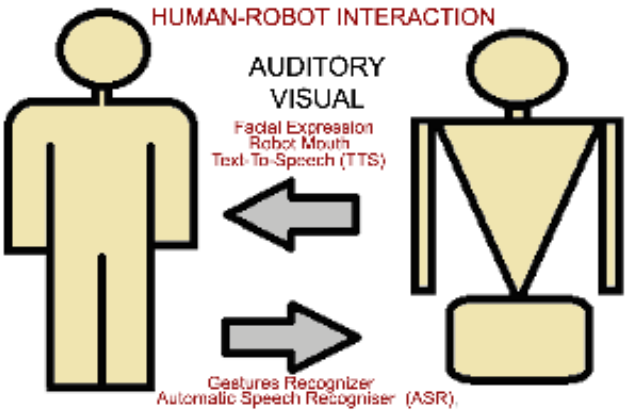

Fig. 1. HRI is usually based on visual and auditory information. For auditory information, depending on the communication direction, TTS or ASR systems are needed.

I. Introduction

DURING the last decade the robotics community interest in social robotics has grown dramatically. It is one of the fields of robotics with more practical applications. Social robots are autonomous robots that interact with humans in daily environments, following human-like social behaviors (i.e., recognizing and expressing emotions, communicating, and helping humans or other robots). During last years the use of social robots has increased for a wide variety of applications (e.g., museum guide robots[1], [2], or assistive and rehabilitation robots[3], [4]). As in other fields of application, robots can offer several key advantages for rehabilitation, such as the possibility to perform (after establishing the correct set-up) a consistent and personalized treatment without fatigue; or its capacity to use sensors to acquire data, which can provide objective quantification of recovery. However, in addition to providing physical assistance in rehabilitation, robots can also achieve personalized motivation and coaching. Thus, it is interesting to study and develop effective mechanisms of interaction between patients and robots.

This interaction between human beings and robots, usually known as Human-Robot Interaction (HRI), represents one of the biggest challenges in social robotics, resulting in new technologies and methods. Different robotic systems have been built and many studies have been conducted unveiling the importance of properly designed human-robot interaction strategies. Some of these works aim to achieve human-like robots in terms of shape [5]. Despite having a similar shape helps achieving higher empathy levels, it is not the only key factor to take into account when developing social robots. The capacity to behave similarly to human beings and to adapt to their emotional state is also a very important issue [7], [8]. Currently, different techniques are being used in order to receive input data from humans (e.g. facial expression recognition [9], skeletal modeling [10], use of corporal language [11], speech recognition [12]), but relatively little scientific research has been done regarding how robots should present the information and give feedback to their users.

In order to perceive their environment, other robots and persons, social robots are equipped with multi-modal sensors like cameras, laser range finders or microphones. Using these sensors, social robots acquire and process the necessary data for establishing communication (e.g., where the interlocutor is, or what is he/she saying or doing). On the other hand, robots working in human environments following social behaviors need methods not only to perceive but also to interact and exchange information between the source and the receiver of the message.

In order to interact with people, robots need to use different communication methods that human beings can easily receive and understand. In this regard, Natural Language (NL), in conjunction with visual information, is a very efficient method for an interaction paradigm with robots (see figure 1). Interactive NL-based communication is comfortable for humans and it is successful handling errors and uncertainties due to the fast-paced loop that such interaction provides. On the other hand, speech perception involves information from more than one

F. Cid, L.J. Manso, L.V. Calderita, A. Sánchez and P. Núñez are with University of Extremadura.
E-mail: fecidb@alumnos.unex.es

sensory modality. In particular, visual information has been proven to be strongly linked with hearing when recognizing speech (McGurk effect [13]). Therefore, it is very likely that mouth synchronization will also help in order to keep the attention of the users in what the robot says. The hypothesis of this proposal is that the HRI experience can be improved using the visual information provided by a robotic mouth whose movements are synchronized with the synthetic voice. Besides, it is analyzed the use of conversational gestures for engaging human-robot interaction.

The perception of the state of the robot and the understanding the voice messages it synthesizes can be improved providing additional information. There are two common information sources for this: **a)** auditory cues (e.g., pitch, pauses, emphasis); and **b)** visual gestures (e.g., lip movements, facial expressions, neck movements). This information allows message senders to acknowledge their emotional state, their intentions and even to transmit concepts. Thus, it is very important to accompany voice with visual feedback and auditory cues in order to ease the correct interpretation of the message to transmit.

This paper presents a robotic head and a synchronization algorithm on a robotic mouth that can perform in real-time with different TTS systems. This algorithm is based on a synchronization algorithm that uses the entropy of the synthetic audio stream for estimating the level of aperture of the robotic mouth. The robotic head, which has a very cost-efficient design, has been included in Ursus social robot, a therapy robot with the shape of a teddy bear [6]. Ursus was designed to improve the therapy of children with developmental disorders like cerebral palsy by making a game of the therapy. Achieving an entertaining therapy for children helps them keeping their attention, which improves the results.

In order to evaluate the initial hypothesis, an opinion poll was conducted with different participants, both roboticists and non-roboticists. The poll took into account: **1)** the impact of the different mouths used in the poll, physical and simulated ones; **2)** how the different TTS systems for voice synthesis influenced user experience; **3)** the impact of the different synchronization algorithms described in the literature [14]; and **4)** how the body language improves the communication of concepts and emotions to the human being. Other factors such as the level of engaging, understanding or acceptance were also evaluated.

The rest of this paper is organized as follows. Section II introduces the state-of-art of the different HRI techniques and their evolution. Section III presents an overview of the proposed system. Next, the robotic mouth designed is described in section IV. Section V presents the synchronization algorithm, describing in detail the different stages of the process. Finally, the results of the experiments proposed in this paper and the conclusions are detailed in section VI and VII, respectively.

II. Related work

Affective communication has been the core topic of different social robotics works. It aims to reduce the communication gap between humans and robots not just by using natural language but also by providing robots with human-like gestures and, to some extent, shape. These techniques allow roboticists to achieve stronger human-robot empathy [15]. Moreover, it is easier for humans to interact with agents with similar characteristics (e.g., appearance, communication mechanisms, gestures). The use of speech-guided dialogue to teach robots [16] allows roboticists and end-users to control and interact with robots using natural language. The first step to achieve this kind of interaction is to be able to send and receive messages through a media that humans can understand. This is done by using technologies such as audio synthesizers (TTS) [17] and speech recognition systems (ASR) [18]. In the last years, these systems are becoming very common in social robotics [19], [20], [21], [22].

Lip synchronization in robotics looks for matching lip movements with the audio generated by the robot. The use of different lip synchronization algorithms not only are limited to use in robotics, but also to the lip animation in virtual models used in HRI systems with computers. These systems allow the user to interact with virtual models through speech and some cases through body language [27], [28], [29]. Several works use synchronization algorithms based directly on the use of audio phonemes to determine the levels of mouth aperture [23], [24]. These approaches require additional information such as dictionaries of phonemes. In this paper it is presented a synchronization algorithm based on the entropy of the synthetic audio signal provided by the TTS system. Despite being a different problem, some authors have successfully used entropy for automatic speech recognition [30], [31], [32] (especially in noisy environments).

Finally, TTS-enabled robots can provide information to humans, but they usually are unembodied monotonous voices, lacking of emotion, pitch changes and emphasys. In [38] it is presented a study of how the prosody of speech influences auditory verbal communication. Similar to [25] and [26], the proposed approach evaluates the different aspects of speech-based interaction.

III. Overview of the System

The main goal of the proposed system is the design of a robotic mouth and the control of a robotic head in order to provide visual information for helping the understanding of the messages synthesized by robots. The mouth is governed by a TTS-lip synchronization algorithm. Thus, the lips move according to the synthesized voice generated using a Text-to-Speech system. This setup helps keeping the attention of social robot users. Figure 2 illustrates an overview of the system. As it is shown in the figure, it is constituted by two layers, hardware and software. The hardware is composed of a previously made robotic head with three degrees-of-freedom, a speaker in order to hear the voice of the robot and the proposed mouth.

IV. Robotic head

The robotic head used in this paper consists of two elements: a neck and a robotic mouth. The robotic neck is driven by three Dynamixel RX-10 servos, allowing pitch, roll and yaw

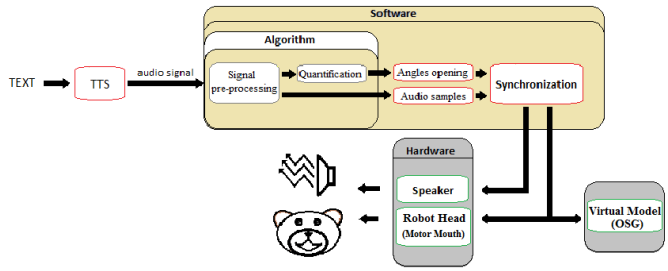

Fig. 2. Overview of the proposed system in this paper. Both, software and hardware layer are depicted in the figure.

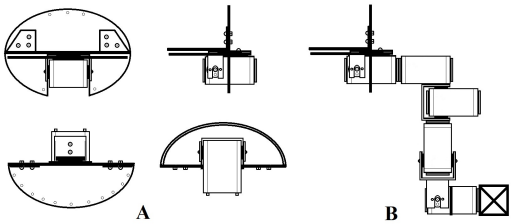

Fig. 3. Different views of the mechanical system. From left to right and from top to down: frontal, profile, top and bottom views.

Fig. 4. On the left hand side is shown, the robotic mouth mounted on Ursus2. On the right hand side is illustrated, the mouth system separated from the rest of the robot, as it is described in section VI

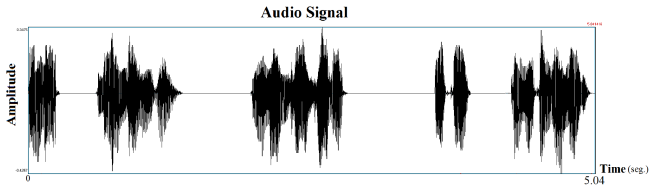

Fig. 5. Audio signal waveform.

movements. The key design considerations for the robotic mouth, which was specially built for this work, are: i) the efficiency of the mechanical system, considering a reasonable range of aperture of the mouth; ii) the suitability of the mouth for its use on the Ursus therapy robot and; iii) the overall price of the mouth. The CAD design of the robotic head (including neck and mouth) is illustrated in figure 3.B. The mechanical structure of the robotic mouth consists of three aluminum planar pieces, corresponding to the chassis of the mouth (figure 3.A), upper and lower lips and a Dynamixel RX-10 servo. The upper aluminum piece is fixed, while the lower lip is moved by the motor. The mouth aperture was set to range between 0 and 45 degrees. Finally, the mechanical pieces are covered by a fabric similar to those used in teddy bears (figure 4).

V. Synchronization Algorithm

A. Text To Speech System

Usually, speech synthesis systems are used in order to directly take the audio output to the speakers. In our case, since we want to make sure that the audio output is synchronized with the mouth movements, the TTS system does not have access to the speakers and it only generates the output audio file. These audio files are then concurrently used for producing both the mouth movements and the audio output.

The proposed lip synchronization algorithm is independent of the Text-to-Speech system. In the experiments shown in this section Verbio TTS system has been used [17] in order to illustrate partial results. This system can generate audio output for different languages, using various audio formats such as OGG or WAV, and allowing adaptive and dynamic intonation.

In particular, the following setup has been used in the approach:

- Language Spanish and English.
- File format OGG.
- Sample rate $Fs = 16Khz$.

Figure 5 illustrates the audio signal obtained for the text: "Hello, my name is Ursus, tell me what is your name".

In section VI, different TTS systems (proprietary and free/libre software) are used for comparison purposes (Verbio, Festival, Ivona and Acapela) in order to demonstrate the correct operation of the lip synchronization algorithm for each one of them. Independent of the TTS system, the proposed algorithm only needs: the sample rate and the output audio file.

B. Signal preprocessing

As it was introduced, mouth movements are based on the entropy level of the audio signal, whose value is calculated on-line for every time window. The input of the algorithm is the audio signal $X(t)$, which has a length of $Fs \cdot T$, where Fs is the sample rate and T the duration of the whole audio signal.

$$X(t) = [0, ..., F_s \cdot T - 1] \quad (1)$$

and obtain the entropy of the windows, the following steps must be taken previously:

- **A)** Obtain the absolute value of the audio signal:

$$V(i) = |X(t)| \quad (2)$$

- **B)** Windowize the signal vector, since the entropy is computed for each window separately.

Time windows have a length of a tenth of a second. It is an adequate length given the nature of the signal (i.e., phonems are usually about a tenth of a second long) and the response time of the motors.. The signal preprocessing step is shown in figure 6.

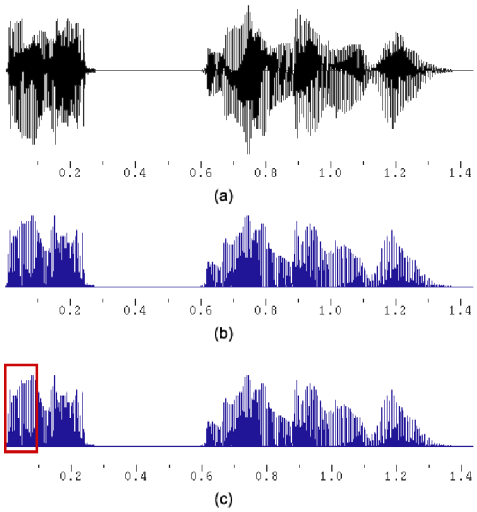

Fig. 6. Signal preprocessing: a) initial audio signal, b) absolute value of the audio signal, b) example of a time window.

C. Quantification

In this work an entropy-based algorithm is proposed in order to set the mouth aperture of the robot given the current audio stream. Since the audio stream is synthetic, it can be safely assumed that the audio is noise free. Thus, the algorithm provides a mouth aperture proportional to the audio entropy for each of the time windows.

Entropy quantifies the existent amount of information in a given signal. Given a set of different samples $1...n$ of a random variable X (which can be interpreted as a signal), the amount of information on it (measured in bits) can be computed as:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i) \tag{3}$$

where x_i is the nth measurement and $P(x_i)$ the probability of finding that measurement within the time window.

Finally, the angle sent to the motor is proportional to the entropy level:

$$angle \propto entropy$$

In our experiments, the proportional constant was experimentally set to 1.5. It might vary depending on the text-to-speech system.

D. Synchronization

The opening levels computed by the algorithm must be synchronized with the audio sent to the speakers. This synchronization is made using the same audio libraries which are used for playback, processing and quantification the audio signals. Thus, the audio samples are simultaneously processed by the audio library and the angles calculated in each time windows are sent to the motors of the robot mouth (see figure 2). Thus, communication delays between the computer and the motors are reduced and the synchronization results are improved.

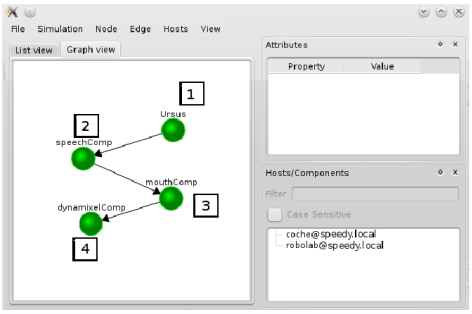

Fig. 7. Screenshot of the *RCManager RoboComp* tool. Ursus main component (1), which is connected to the TTS system (2). The component which transforms the sound in motor movements is labeled as 3. Component moving the servomotor (4).

E. RoboComp Components

The software to control our system is built on top of the robotics framework *RoboComp* [35]. Making use of the components and tools it provides and its communication middleware we developed an easy to understand and efficient architecture (see figure 7).

The main component of the proposed system is ursusComp. It is connected, directly or indirectly, to the rest of the software components controlling Ursus: camera, robotic arms, tracker, etc (figure 7). Not all components have been included in the diagram in order to make it simple. The sentences that Ursus tells its patients to encourage them during their therapy are sent to speechComp (see figure 7-(2)). Then speechComp transforms the sentences into sound using the specific TTS system (e.g. Festival, Verbio). After that, mouthComp (figure 7-(3)) receives the sound and send the motor commands using the synchronization algorithm. Finally, the motor commands are received and executed by dynamixelComp.

Since the system was designed an implemented using component oriented design/programming, these components can be easily used for other purposes, which is a very important feature in robotics development.

VI. Experimental Results

One of the main goals behind the development of the robot head and the synchronization algorithm is to use them as an improvement for Human Robot Interaction. The initial hypothesis was that the use of a robotic mouth moving according to the synthetic voice generated by a Text-To-Speech system allows a) robots to maintain the attention of their users while talking and, b) human beings to interact more efficiently with robots. The idea is that robots equipped with motorized mouths can achieve a better interaction than those which are not. The second hypothesis is that the use of head body language is useful in order to successfully transmit concepts or emotions.

There exist different approaches to evaluate the performance of social robots when interacting with humans. In addition to evaluating the synchronization algorithm, it is also interesting and necessary to analyze how the proposed robot mouth affects humans. For this purpose, different works and researchers propose the use of quantitative measures of human attention or body movement interaction between robots and humans [34],

Fig. 8. First version of the therapist robot Ursus.

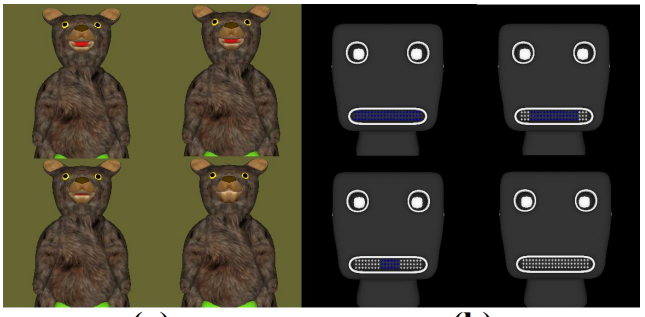

Fig. 9. Screenshots of different mouths. a) Virtual Robot Ursus b) Robotic Model based on LED matrix mouth. Both models are shown in four different positions.

[33]. In this paper, acceptance, engaging and understanding are three factors to be measured in the HRI context. These factors are evaluated using pool-based methods, where the opinion of the user is surveyed.

Thus, the performance of the proposal has been evaluated based on the impression of the participants regarding the synchronization algorithm and the robotic head according to: 1) the difference in perception between a physical robotic mouth and a simulated one; 2) how the different TTS systems for voice synthesis influenced user experience; 3) the impact of the different synchronization algorithms described in the literature [14]; and 4) how the body language improves the communication of concepts and emotions to humans.

A. Robot platform Ursus

Ursus is an assistive robot developed in the Laboratory of Robotics and Artificial Vision of the University of Extremadura (Cáceres, Spain) in collaboration with the Virgen del Rocío Hospital (Sevilla, Spain). It is designed to propose games to children with cerebral palsy in order to improve their therapy. It will also be used as a tool for their therapists to adjust therapy to the needs of the different patients. In order to make it visually pleasant for children, it has a friendly height and has been wrapped into the covering tissue of a teddy bear.

Patients can get feedback of the status of the exercise in real-time by looking at an image that the robot projects on the wall. Along with the messages the robot sends to the patients, this information encourages children to improve the execution of the exercises. Figure 8 illustrates the first version of Ursus. Ursus is composed of two arms, both of four degrees of freedom (DOF), mounted on a fixed torso. These are used so that patients can see how the robot perform the exercise and try to reproduce the movement. A regular USB camera is located in the neck of the robot to capture the arm movements of the users, allowing the robot to provide appropriate feedback about their performance. The speaker and the computer are located on the base of the robot.

B. Comparative study

Social robotics enables robots to interact with diverse groups of people, through simple and friendly means of communication such as as speech [23] [20]. A comparative survey was conducted to assess various aspects of the mouth through a series of questions with a response on a linear scale of 1-5. For the study has been selected 15 persons, each one with different degrees of knowledge about robotics. These degrees of knowledge can be divided into three categories: high (5 persons), medium or moderate (4 persons) and low (6 persons).

The evaluated items were divided into four groups: robotic mouths, TTS softwares, lip synchronization algorithms and body language, which were compared in order to determine the best in each group.

The following elements were evaluated by the participants:

- **A)** Natural behavior
- **B)** Expressiveness
- **C)** Attention engaging capacity
- **D)** Message understanding

Different questions were used in order to obtain an average score for each study as:

- Does the mouth seem to move naturally?
- Does the mouth seem expressive?
- Did the mouth capture your attention?
- Did the mouth, directly or indirectly, help you to understand the message?

Were repeated using different sentences in order to obtain an average value.

C. Comparative study of different robot mouths

The robotic mouth was compared with two different designs used for researching in Human Robot Interaction. First, Figure 9.a illustrates the robot Ursus virtual model created in 3D Studio Max with a mouth, which it is moved according to the proposed synchronization algorithm. The second robotic mouth included in this comparative study is based on other research works that use a LED matrix [14] (see figure 9.b). Instead of developing the hardware LED matrix it was also simulated. It consists on a 21x3 matrix whose elements are enabled according to the synchronization algorithm. Thus, they are turned on as entropy increases. Both mouths are displayed on the screen monitor using a size similar to the Ursus one. In figure 4 it is shown the set up used for evaluating the robotic mouth.

Mouth	Questions			
	A	B	C	D
Animated mouth	67%	68%	69%	78%
Led mouth	42%	46%	59%	73%
Robotic mouth	74%	66%	74%	64%

TABLE I
COMPARISON OF THE DIFFERENT MOUTHS TESTED

TTS	Questions			
	A	B	C	D
Verbio	52%	46%	52%	72%
Festival	60%	56%	52%	80%
Acapela	68%	72%	68%	72%
Ivona	64%	60%	56%	68%

TABLE II
COMPARISON OF THE DIFFERENT TTS

TTS/Robotic mouth	Questions			
	A	B	C	D
Verbio	74%	66%	74%	64%
Festival	50%	45%	50%	60%
Acapela	80%	75%	80%	70%
Ivona	65%	60%	70%	60%

TABLE III
COMPARISON OF THE DIFFERENT TTS IN THE ALGORITHMS OF SYNCHRONIZATION

Algorithms synchronization	Questions			
	A	B	C	D
Entropy	80%	80%	80%	64%
Random	40%	44%	40%	36%
Binary	48%	44%	48%	48%

TABLE IV
COMPARISON OF THE DIFFERENT TTS IN THE ALGORITHM OF SYNCHRONIZATION

The same pool and participants were used for evaluating the features of the different robotic mouths. Results are summarized in table I.

As shown in table I, the robotic mouth presented in this paper performs better compared with other mouths in elements such as: natural Behavior and Attention Engaging capacity, resulting in a greater interaction and attention by the survey participants. In addition to the robotic mouth, the survey shows that the animated mouth presents excellent results in items such as: expressiveness and response in the Message Understanding.

D. Comparative study of the different Text-To-Speech systems

This section describes the evaluation of different TTS systems. In this study four different TTS systems have been used: Verbio, by Verbio Technologies; Festival, by the University of Edinburg; Acapela, by the Group Acapela; and Ivona, by the company Ivona Software.

One of the main aspects to take into account when using a TTS system is the output sample rate. In this study, for each TTS, the following sample rates were used.

- Verbio: $16Khz$
- Festival: $44Khz$
- Ivona: $22Khz$
- Acapela: $22Khz$

The algorithm can be used with any TTS system, as long as it complies with certain parameters such as audio sampling frequency or the ability to produce output files.

For the evaluation of the TTS systems, the questions specified in section VI-B have been used. The evaluation results of the TTS systems are summed up in table II.

The results shown in the table II, demonstrate that the Acapela TTS software performs better than other TTS in aspects such as naturalness or expressiveness. In addition to evaluating the synthesizer, the poll took into consideration the performance of each TTS with the robotic mouth and the algorithm of synchronization, that is the key of this work. The corresponding results to the use of the TTS with the robotic mouth are shown in table III.

Table III shows that the best achieved performance is produced by Acapela in conjunction with the proposed synchronization algorithm and the robotic mouth developed for this paper. Demonstrating how the synchronization algorithm helps improve speech perception, and allowing the user to be able to perceive of simplest form, the elements evaluate in this survey, such as attention engaging capacity, expressiveness or the natural behavior.

E. Comparative study of different synchronization Algorithms

Finally, the comparative study allowed to evaluate the synchronization algorithm compared to other algorithms, such a binary pulse delivery aperture (mouth opened if there is sound) and other that controls movement through random levels of mouth aperture.

For the evaluation of these synchronization algorithms a survey was made with the questions of the subsection VI-B. Results have been summarized in table IV.

The results of the survey demonstrate in the table IV, that the synchronization algorithm based on entropy provides a better user experience in comparison to similar algorithm in all the elements evaluated by survey. Besides, being the best performing in speech perception, it has other features that make it useful for social robotics, as its ability to work with any TTS software.

F. Comparative study of the used of the body language

This comparative study evaluated the impact of body language in human-robot communication. An experiment was conducted with voice messages from a TTS system with modified parameters, as emphasis, pitch, pauses, among others. These changes affected the entire sentence, at the same time that a few words. The set of parameters of the algorithm was not modified, except the size of the time windows after modifying the speed of the TTS. In addition, a second experiment that used not only the voice message, but also a series of movements (body language) to express a concept as doubt, a question and anger. Both two experiments have been conducted with a virtual robot (specifically the robot Ursus).

The results of the survey carried for the two experiments can be seen in Table V:

Body language	Questions			
	A	B	C	D
Only voice	54%	60%	63%	75%
Voice and Move	56%	65%	76%	75%

TABLE V
COMPARISON OF THE USED OF BODY LANGUAGE.

In table V, the experimental results have shown how body language, in comparison to the dialogue based only in the speech, provides a better performance in elements as: natural behavior, expressiveness and attention engaging capacity. Thus, the importance of body language in the interaction of robots with users is demonstrate, giving rise to a natural language based on the motion and speech by the robots from humans, but allowing similar communicate.

VII. CONCLUSION

Social robots need to communicate properly in order to improve their interaction skills with people. In this respect, both visual and auditory sources must be taken into account as it was demonstrated by McGurck[13]. The use of visual feedback (i.e., head and mouth movements) can be used, not only to improve understanding, but also to achieve higher levels of attention and empathy.

The results provided by the survey demonstrate that the proposed algorithm has several advantages over the state-of-the-art algorithms. Moreover, our algorithm performs in real-time and does not require additional training such as other approaches [23], [24].

Currently, the RoboLab group is working in order to be able not only to provide speech information to the user but also to receive it. By removing the need to make the user move (i.e. in order to touch a touchscreen), is expected that user experience will be dramatically enhanced.

ACKNOWLEDGMENT

This work has been partially supported by the Junta de Extremadura project IB10062 and by the Spanish Ministry of Science and Innovation with grant IPT-430000-2010-002.

REFERENCES

[1] W. Burgard, A. B. Cremers, D. Fox, D. Hahnel, G. Lakemeyer. “Experiences with an interactive museum tour-guide robot”.In *Artificial Intelligence (AI)*, pp. 3-55. 2000.

[2] F. Faber, M. Bennewitz, C. Eppner, A. Gorog, “The humanoid museum tour guide Robotinho”. In *Proc. IEEE International Symposium on Robot and Human Interactive Communication* (RO-MAN). September 2009.

[3] X. Ma and F. Quek, “Development of a Child-Oriented Social Robot for Safe and Interactive Physical Interaction”. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems* Taiwan, pp. 2163-2168, October 2010.

[4] T. Mukai, S. Hirano, H. Nakashima, Y. Kato, Y. Sakaida, S. Guo, and S. Hosoe, “Development of a Nursing-Care Assistant Robot RIBA That Can Lift a Human in Its Arms”, In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taiwan, pp. 5996-6001, October 2010.

[5] P.Breen, E. Bowers, W. Welsh, “An investigation into the generation of mouth shapes for a talking head”. In *Proc. of ICSLP 96*, USA, pp. 2159-2162, October 1996.

[6] L.V. Calderita, P. Bachiller, J.P. Bandera, P. Bustos, and P. Nunez, ”MIMIC: A Human motion imitation component for RoboComp”. In *Proc of Int Workshop on Recognition and Action for Scene understanding.*. 2011.

[7] T. Hashimoto, S. Hitramatsu, T. Tsuji, and H. Kobayashi, “Development of the Face Robot SAYA for Rich Facial Expressions”. In *Proc. of 2006 SICE-ICASE International Joint Conference* Korea, pp. 5423-5428, October 2006.

[8] M.J. Matarić , J. Eriksson , D.J. Feil-Seifer1 and C.J. Winstein. “Socially assistive robotics for post-stroke rehabilitation”. In *Journal of NeuroEngineering and Rehabilitation*. 4:5. 2007.

[9] J.A. Prado, C. SimplÃÑcio, N.F. Lori and J. Dias, ”Visuo-auditory Multimodal Emotional Structure to Improve Human-Robot-Interaction”, In *International Journal of Social Robotics*, vol. 4, no. 1, pp. 29-51, December 2011.

[10] J.P. Bandera, ”Vision-Based Gesture Recognition in a Robot Learning by Imitation Framework”, Ph.D. Thesis, University of Malaga, 2009.

[11] A. Aly and A. Tapus, ”Speech to Head Gesture Mapping in Multimodal Human-Robot Interaction”, In *Proc. of the 5th European Conference on Mobile Robots ECMR 2011* Sweden, pp. 101-108, September 2011.

[12] C. Breazeal and L. Aryananda, ”Recognition of Affective Communicative Intent in Robot-Directed Speech”, Artificial Intelligence, pp. 83-104, 2002.

[13] T. Chen, “Audio-Visual Integration in multimodal Communication”. In *IEEE Proceedings*, May, 1998.

[14] M.K Lee, J. Forlizzi, P.E. Rybski, F. Crabbe, W. Chung, J. Finkle, E. Glaser, S. Kiesler, “The Snackbot: Documenting the Design of a Robot for Long-term Human-Robot Interaction”. In *Proc. of HRI 2009*, pp. 7-14, 2009.

[15] R. W. Picard, “Affective Computing”. MIT Press, pp. 88-91, 2000.

[16] J. Gómez, A. Ceballos, F. Prieto, T. Redarce, “Mouth Gesture and Voice Command Based Robot Command Interface”. In *Proc. of IEEE International Conference on Robotics and Automation*, Japan, pp. 333-338, May. 2009.

[17] Verbio Technologies, “Text to Speech (TTS) and Speech Recognition (ASR)”. Available at: *http://www.verbio.com*.

[18] S. Anderson, D. Kewley-Port, “Evaluation of Speech Recognizers for Speech Training: Applications”, In *IEEE Transactions on speech and audio processing*, VOL. 3, NO. 4, pp. 229-241, July 1995.

[19] C. Jayawardena, I. H. Kuo, U. Unger, A. Igic, R. Wong, C. I. Watson, R. Q. Stafford, E. Broadbent, P. Tiwari, J. Warren, J. Sohn and B. A. MacDonald. “Deployment of a Service Robot to Help Older People”. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems* Taiwan, pp. 5990-5995, October 2010.

[20] C. Shi, T. Kanda, M. Shimada, F. Yamaoka, H. Ishiguro and N. Hagita “Easy Development of Communication Behaviors in Social Robots”. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* Taiwan, pp. 5302-5309, October 2010.

[21] W. Zhiliang, L. Yaofeng, J. Xiao, “The research of the humanoid robot with facial expressions for emotional interaction”. In Proc. *First International Conference on Intelligent Networks and Intelligent Systems*, pp. 416-420. 2008.

[22] P. Rybski, K. Yoon, J. Stolarz, M. Veloso, “Interactive Robot Task Training through Dialog and Demonstration”. In *In Proc. of HRI 2007*, USA, 2007.

[23] K.-geune Oh, C.-yul Jung, Y.-gyu Lee, and S.-jong Kim, “Real Time Lip Synchronization between Text to Speech(TTS) System and Robot Mouth”. In *Proc. of IEEE International Symposium on Robot and Human Interactive Communication*,Italy, pp. 620-625, September. 2010.

[24] F. Hara, K. Endou, and S. Shirata, “Lip-Configuration control of a Mouth robot for japanese vowels”. In *Proc. IEEE International Workshop on Robot and Human Communication* pp. 412-418, 1997.

[25] A. Austermann, S. Yamada, K. Funakoshi, M. Nakano, “Similarities and Differences in Users Interaction with a Humanoid and a Pet Robot”. In *Proc. 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)* pp.73-74, 2010.

[26] H. Kamide, Y Mae, T. Takubo, K. Ohara and T Arai. ”Development of a Scale of Perception to Humanoid Robots: PERNOD”. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* Taiwan, pp. 5830-5835, October 2010.

[27] S. DiPaola, A. Arya, J. Chan, ”Simulating Face to Face Collaboration for Interactive Learning Systems”, In *In Proc: E-Learn 2005*, Vancouver, 2005.

[28] K. Waters and T.M. Levergood, ”DECface: An Automatic Lip-Synchronization Algorithm for Synthetic Faces”. In *MULTIMEDIA TOOLS AND APPLICATIONS* Vol. 1, Number 4, pp. 349-366, 1995.

[29] Kaihui Mu, Jianhua Tao, J. che and M. Yang, "Real-Time Speech-Driven Lip Synchronization", In *4th International Universal Communication Symposium (IUCS), 2010* , Beijing ,pp. 378-382, 2010

[30] J-L. Shen, J-W. Hung, L-S. Lee, "Robust entropy-based endpoint detection for speech recognition in noisy environments", In *ICSLP-1998*, 1998.

[31] J.C. Junqua, B. Mak, and B. Reaves, "A Robust Algorithm for Word Boundary Detection in the Presence of Noise", In *IEEE Trans. on Speech and Audio Processing*, Vol. 2, No.3, pp. 406-412, Apr. 1994.

[32] M.H. Savoji, "A Robust Algorithm for Accurate Endpointing of Speech", In *Speech Communication*, Vol. 8, pp. 45-60, 1989.

[33] A. Tapus and M.j. Mataric, "Emulating Empathy in Socially Assistive Robotics", In *AAAI Spring Symposium on Multidisciplinary Collaboration for Socially Assistive Robotics*, Stanford, USA, March 2007

[34] A. Paiva, J. Dias, D. Sobral, R. Aylett, P. Sobreperez, S. Woods, C. Zoll and L. Hall, "Caring for Agents and Agents that Care: Building Empathic Relations with Synthetic Agents", In *Third International Joint Conference on Autonomous Agents and Multiagents Systems*, Vol. 1, pp. 194-201, New York, USA, 2004.

[35] L.J. Manso, P. Bachiller, P. Bustos, P. Nuñez, R. Cintas and L. Calderita. "RoboComp: a Tool-based Robotics Framework". In *Simulation, Modeling and Programming for Autonomous Robots* (SIMPAR). Pages 251-262. 2010.

[36] M. Siegel, C. Breazeal, and M. I. Norton, "Persuasive Robotics: The influence of robot gender on human behavior". In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2563-2568, October 2009.

[37] H. Song, and D. Kwon, "Design of a Robot Head with Arm-type Antennae for Emotional Expression". In *International Conference on Control,Automation and System. ICCAS '07*, pp. 1842 - 1846, October 2007.

[38] J. Cahn. "Generation Expression in synthesized speech". Master's Thesis, MIT Media Lab. 1990.

Obstacle Avoidance in Underwater Glider Path Planning

Josep Isern-González, Daniel Hernández-Sosa, Enrique Fernández-Perdomo,
Jorge Cabrera-Gámez, Antonio C. Domínguez-Brito and Víctor Prieto-Marañón

***Abstract*—Underwater gliders have revealed as a valuable scientific platform, with a growing number of successful environmental sampling applications. They are specially suited for long range missions due to their unmatched autonomy level, although their low surge speed makes them strongly affected by ocean currents. Path planning constitutes a real concern for this type of vehicle, as it may reduce the time taken to reach a given waypoint or save power. In such a dynamic environment it is not easy to find an optimal solution or any such requires large computational resources. In this paper, we present a path planning scheme with low computational cost for this kind of underwater vehicle that allows static or dynamic obstacle avoidance, frequently demanded in coastal environments, with land areas, strong currents, shipping routes, etc. The method combines an initialization phase, inspired by a variant of the A* search process and ND algorithm, with an optimization process that embraces the physical vehicle motion pattern. Consequently, our method simulates a glider affected by the ocean currents, while it looks for the path that optimized a given objective. The method is easy to configure and adapt to various optimization problems, including missions in different operational scenarios. This planner shows promising results in realistic simulations, including ocean currents that vary considerably in time, and provides a superior performance over other approaches that are compared in this paper.**

***Index Terms*—Path planning, underwater gliders, obstacle avoidance.**

I. Introduction

Robotic Unmanned Underwater Vehicles (UUV) have demonstrated to be a valuable tool for a wide range of applications in oceanography and surveillance, including structure inspection, environmental monitoring and control or security. Since the possibilities of human intervention are quite limited during the robot mission, these vehicles can be conceived as physical agents that must perform their tasks with a high level of autonomy. In fact, they are commonly known as Autonomous Underwater Vehicles (AUV). However, it is hard to accomplish this goal as a consequence of the inherent dynamism and uncertainty of the state of both the vehicle and its environment, estimated with a separate model each.

A glider is a type of UUV that operates by modifying its buoyancy in a cyclic pattern. These changes produce vertical impulsion that is transformed into an effective but low surge speed by means of the combined effect of internal mass displacements and the vehicle wings and tail orientation, resulting in a succession of up/down slope or climb/dive transects (see Fig. 1). In terms of power consumption, the glider saw-tooth profile is very efficient, since the gravity is used as the power source for propulsion, that is the most critical task of UUVs autonomy. Besides processing and communication, the batteries are only used intensively during a small fraction of the cycle time to change the vehicle buoyancy, using an electric pump; and, much less demanding, to modify the vehicle attitude and bearing angle while submerged using low consumption actuators. Ocean gliders have been applied successfully in Maritime Research, and they are expected to become one of the reference technologies as observational tool in the coming years [19].

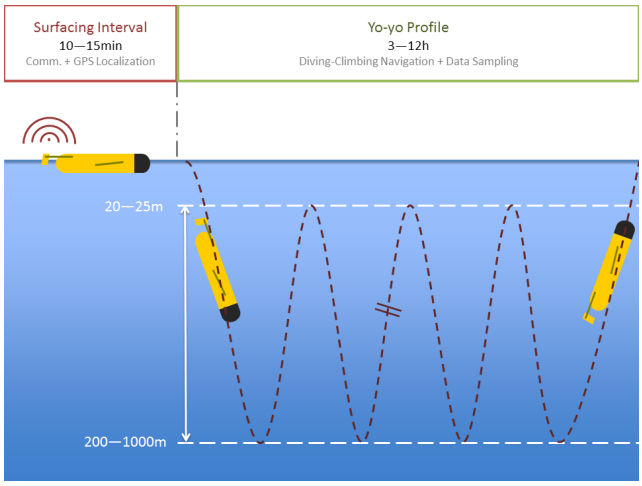

Fig. 1. Glider saw-tooth navigation pattern.

Periodically, the glider surfaces to detect its position by GPS and to communicate data via satellite to the ground station. It waits a few minutes for new orders. While the glider is submersed it does not change its heading. We have taken into account this feature (time discretization) to develop our path planner.

The top view of the surfacing interval and the yo-yo profile

Authors are with University Institute of Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería (SIANI), Universidad de Las Palmas de Gran Canaria

E-mails:
{ jisern, dhernandez, efernandez } @ iusiani.ulpgc.es
{ jcabrera, adominguez, vprieto } @ iusiani.ulpgc.es

This work has been partially supported by the project TIN2008-06068 funded by the Ministerio de Ciencia e Investigación, Spanish Government. It has also been partially supported by project ProID20100062 funded by the Autonomous Government of Canary Islands (Agencia Canaria de Investigación, Innovación y Sociedad de la Información) and FEDER.

stint (Fig. 2) shows how on surface the glider trajectory is known using the GPS. But while submerged, after the diving point it is unknown, although it can be estimated up to some uncertainty. At each surfacing point such uncertainty collapses with the first GPS fix.

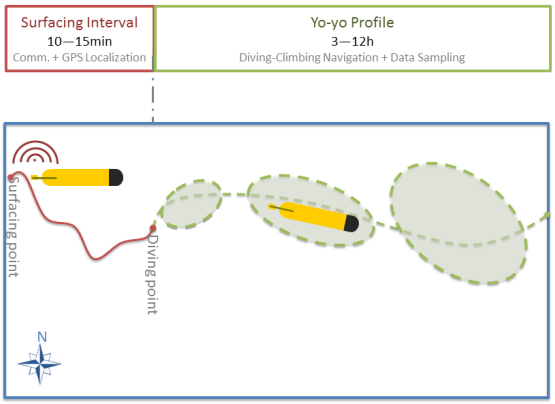

Fig. 2. Top view of a glider navigation pattern.

The main source of uncertainty is the drifting caused by ocean currents. Their low surge speed (aprox. 0.3 - 0.4 m/s) make gliders far more influenced by ocean currents than other UUVs that can overcome them. Gliders may drift significantly from its intended trajectory, making path planning a crucial tool for this type of vehicles, as it might reduce the time spent to reach a given waypoint or save power.

A. Motivation

Our work has been organized around the analysis of path planning requirements in presence of obstacles and the study of its performance in different scenarios. Regarding the former, we have identified several factors that, in our humble opinion, should be assessed at the design phase of a path planner for ocean gliders.

In Robotics, path planning addresses the problem of getting a robot from one point to another. This simple task is very challenging when it is to be solved under the influence of ocean currents. The currents field directly affects the movement of the vehicle so that the cost of displacement is variable and anisotropic at different points in space. Compared to ground mobile robotics, the underwater scenario is much more challenging, since operating conditions can vary notably even on reduced areas and over a relatively short period of time. In the particular case of ocean gliders, all the mentioned difficulties are magnified. Automatic path planning constitutes a key capability because underwater robots are usually commanded in terms of goal navigation waypoints to be hit or target regions to be explored.

For these reasons, most of classical approaches in the path planning field are not directly applicable to this problem. Many path planners apply a certain form of discretization, either on the trajectory or command space, to reduce the computational cost. However, the downside of discretization lies in the presumably degradation of the quality of the results, that might lead to unrealistic trajectories.

The execution time is another factor which is often understated due to the typical long duration of glider missions and immersion periods. Although this is generally true, it is not the case when the path planner must respond within a reduced time interval to face an unforeseen situation.

In this paper we analyze the path planning problem in specially troublesome scenarios, mainly coastal, that include static and dynamic obstacles such as strong currents, land areas or heavy traffic shipping routes. There, the planner pursues the maximization of the distance traveled towards a distant way-point —or, in other words, the minimization of the remaining distance to reach it— over a short and known period of time. This corresponds to a leg/stage range planning with a maximum duration of three or four days and a typical trajectory length around 100 km. For this temporal horizon, ocean current forecasts of high temporal resolution are used. These forecasts can be obtained from some Regional Oceanic Models (ROMs) with hourly outputs. ROMs are forecast systems of currents and other oceanographic variables that are based on numerical models. In such configuration, the path planning problem is clearly performed in a time-varying scenario.

In this work, we present a novel path planning technique for underwater gliders in troublesome coastal environments that introduce an initialization module to avoid obstacles inspired on A*-based search and Nearest Diagram (ND) algorithms [14] that is combined with optimization process. The glider is modeled here as an intelligent agent that senses the speed and direction of forecast of ocean currents via ROMs to generate an optimized trajectory that tries to fulfill a given task. A path planning allows reducing the time, and consequently the energy consumption. Thus, we will have more autonomy. The method is quite flexible, as it can be applied to a number of other optimization problems with few adaptation or configuration. It shows promising results in realistic simulations, under highly time-varying ocean currents. The proposal gives a superior performance when compared with other approaches.

This paper is organized as follows: the next subsection includes a revision of UUVs path planning approaches. The next section presents an explanation of the previous steps of our new approach. Then, in section III, the proposed method is described in detail. Section IV presents the experiments carried out to validate our path planning algorithm. Finally, section V contains the conclusions extracted from this work.

B. Related works

Path planning for UUVs has been a subject of interest for researchers since the introduction of these robotic platforms. Different approaches have been developed applying techniques that include searching algorithms based on artificial intelligence, potential field modeling, multi-objective optimization, etc. Some of the most relevant, in our opinion, are summarized in the following.

There exists a number of works that have addressed this problem with different optimization frameworks. First, graph

methods are adequate to solve the problem assuming not time-dependent ocean currents. The A* algorithm [7] is a classical path planning method from Artificial Intelligence. It's a graph method that discretizes the search space using an uniform grid. For example, Carroll et al. [3] apply this strategy on a quad-tree search space. Probably, the first paper that adapted the A* algorithm to AUV's was contributed by Garau et al. [6]. It explains how to apply A* algorithm to marine vehicles, by means of adapting the cost function and incorporating ocean currents on a uniform grid discretization. Then, Petres [16], [17] proposed a combination of Fast Marching and A*, to obtain the accuracy and efficiency of each. It also addresses the discretization problem of the search space that A* has. Soulignac extended this line by presenting a series of papers [21], [22], [23] that manages strong and time-dependent ocean currents. In both cases, the approach bases on Wavefront Expansion, which is Dijkstra's method [4] in essence. Usually, graph methods have as a main drawback the negative effect of the search space discretization.

The high dimensionality of the search space has led to random exploration based approaches. The rapid random trees or RRT [12] [20] are a good example of this, and have been applied to the case of route planning for AUVs [24] and gliders [18]. This approach is particularly fast, but the path found is sub-optimal and requires further refinement. Post smoothing techniques cannot be applied directly with time-dependent conditions. It builds up an exploring tree with nodes that tend to cover the search space, generating trees from both the start and end points. However, it is not applicable in time-varying scenario and there is no guarantee of finding a route, and even less an optimal trajectory.

The problem has also been modeled as a Boundary Value Problem, using Zermelo optimal navigation formula for time-dependent currents, and Dubins curves for not time-dependent [25]. These techniques require fine tuning, and they only find a solution in simple test cases. Interestingly, it is possible to impose speed and acceleration constrains on the vehicle motion, in the case of Dubins curves.

Bio-inspired methods cover techniques like genetic and optimization algorithms that often have a large convergence time. Evolutionary computing has also been successfully applied to this type of problems. A significant example can be found in [1], where genetic algorithms are used for AUV trajectory planning in environments characterized by time-varying currents. The approaches based on minimization of energy functions are also worth commenting. As good examples, we can cite the work of Kruger et al. [11], that includes the time as an extra dimension in the search space, or Witt et al. [27], that incorporate modeling of time-varying obstacles using potential fields. The problem of local minima has been tackled by means of strategies based on particle swarms, simulated annealing, or genetic algorithms. In other proposals, the currents are modeled as continuous time functions, as is the case of the non-linear trajectory generation or NTG method [13] applied over B-Splines of Zhang et al. [29]. Moqin et al. [15] propose an iterative optimization process for glider path planning. However, the focus of that work is centered on the waypoint precision enhancement, and not in optimal path planning. Furthermore, only static ocean currents are considered. Recently, some authors have applied Genetic Algorithms, Particle Swarms, Simulated Annealing, Swarm Optimization [26]. In all these cases, the main drawback is the high computational cost.

Finally, in the last years a line that has received a lot of attention from researchers is the use of multiple vehicles in a coordinated mission. Some relevant examples include [28] and [2], that face the problem of adaptive sampling of oceanic variables by means of gliders fleets.

II. Evolution of the Algorithm

A. Origins

We started our work following the trajectory of a real glider, RU27 *Scarlet Knight* glider. Our first option was the trivial solution, Direct to Goal algorithm. At each surfacing the next bearing is computed as the direction to the goal point Fig. 3. It does not take into account the forecast of ocean currents. Truly, this is not a path planning algorithm, but it resembles the glider behavior. Its main limitation is that the glider drifts significantly in the presence of strong currents and does not find path which can be benefit from favorable currents when these are not in the direction of the target. Basically we have used this algorithm as reference to compare the new developments.

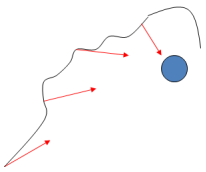

Fig. 3. Direct to Goal algorithm.

In the next step, we adapted A* method to manage ocean currents as in Garau's work [6], using the constrained motion model of Soulignac [21] (Fig. 4). The major drawback of this approach is that it doesn't produce stints of constant time, as gliders do. Also, the optimality is no longer guaranteed, because ocean currents are non-static.

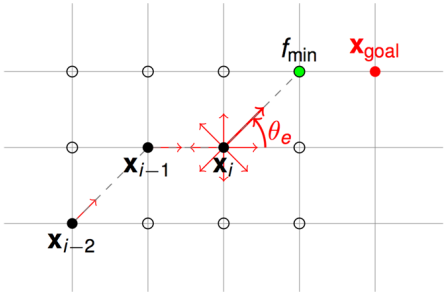

Fig. 4. A* method managing ocean currents.

B. CTS-A* method

To alleviate such limitations, we developed the CTS-A* algorithm [5] (Fig. 5), a variant of A*. At each surfacing

point a set of bearings is considered and for each one, the glider trajectory is integrated for a constant-time stint. The surfacing locations are continuous, although they are stored in a search grid. With this approach have two problems, the bearings space is discretized and if we increase the number of bearings, the computational cost increases exponentially.

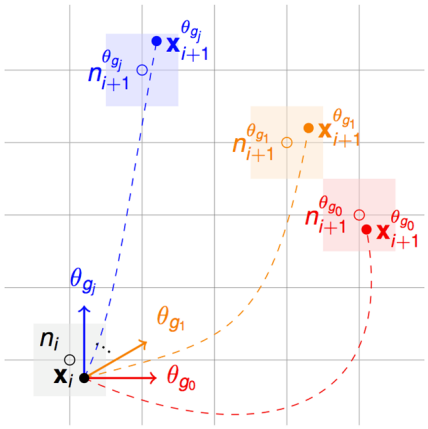

Fig. 5. CTS-A* method.

C. Optimization-based algorithm

Finally we have applied optimization techniques [8], [9] to solve this problem. We have used the distance of the last surfacing to the target waypoint as objective function and the bearings at each submersed stint as variables, which are iteratively optimized to find the path of minimal cost. With this election, the benefit is twofold, avoiding discretization and allowing for a physically realistic simulation. In [10] this method was adapted to coordinate the trajectories of a fleet of gliders.

As commented in the introduction, gliders propel themselves by changing their buoyancy and transforming the resultant vertical motion, of continuous dives and climbs, into a surge movement by means of the combined action of the internal mass displacement and the external control planes. These cycles are repeated typically for 3-12 hours periods, called transects or stints. Once a stint is finished, the vehicle surfaces to communicate the status and data gathered to the control room and receive new orders, commonly the next waypoint or bearing. After 10-15 minutes at the surface, the next immersion period starts. An important fact is that gliders do not communicate while submersed, and the on-board navigation system simply tries to keep the last commanded heading or bearing during the whole stint.

Additionally, the number of variables to optimize is a function of the stint and the total path durations. Therefore when we know the temporal horizon of the trajectory the number of stints is known, consequently the number of bearings that must be commanded and so the number of parameters to be optimized is known. As an example, a 4-day mission using the Slocum Electric Glider would require 12 variables for the standard 8 hours transect. In most cases, the final value returned by the objective function is computed as a distance metric.

The cost function of the optimization process is computed on the basis of a stint simulator that reproduces the glider trajectory combining the commanded bearing with the nominal glider speed and 2D ocean currents. For this purpose, our simulator applies a simple glider kinematic model. Fig. 6 illustrates the strong influence of ocean currents on the resultant glider trajectory, as a consequence of its relative low surge speed. Also, in this figure, it is observed the high variability of currents orientation in only 3 days.

In previous papers we compared optimization-based method with others algorithms. Fig. 7 shows the paths obtained with each method for a particular test case. In all the cases studied, the optimization-based method obtained the best results. In the majority of the test cases the difference was of a few kilometers, but in some cases we found a very large difference, as it happens in the simulation of 4 days that appears in Fig. 8, where the improvement is of approx. 100km with respect to A*.

This approach produces acceptable results for static, moderate-strength ocean currents. However, as indicated previously, in this work we are interested in short-term coastal navigation. There, and due to the complexity of the environment and the coupled nature of the process variables, the optimization can easily get trapped in local minimum or lead to wrong paths, including collisions (Fig. 9).

III. PATH PLANNER

To overcome the limitations of obstacle avoidance, that we have found in the previous versions of our algorithm, we have developed a new path planner, that we call Optimization with Intelligent Initialization. This algorithm integrates a bootstrap module inspired on CTS-A* search and ND algorithms, that generates an appropriate initial set of values to start the optimization phase described thus far.

A. Initialization (obstacle avoidance)

The initialization process makes a division of candidate trajectories in two or more stages. These candidate trajectories use a fixed bearing in all the stints into one stage. The nodes are the division points between stages. In the algorithm the position of these points is flexible (Fig. 10).

First, the initialization process considers a set of angularly equispaced radial vectors emanating from the starting point and simulates the glider trajectory with a fixed bearing for each one, inside the temporal horizon (Fig. 11). Second, a set of points is selected for each trajectory. These points (candidate nodes) are selected at equispaced surfacing points, generally this is corresponded to equispaced time instants (Fig. 12). Third, it considers a set of angularly equispaced radial vectors emanating from every node and simulates a constant bearing for each trajectory for the remaining temporal horizon, that is, recursively, a new set of trajectories is generated for each candidate node, simulating them for the remaining mission time (Fig. 13). Finally, it selects the bearings of the trajectory that reaches the nearest position to the target point as initial guess value for the optimization process (Fig. 14). As an

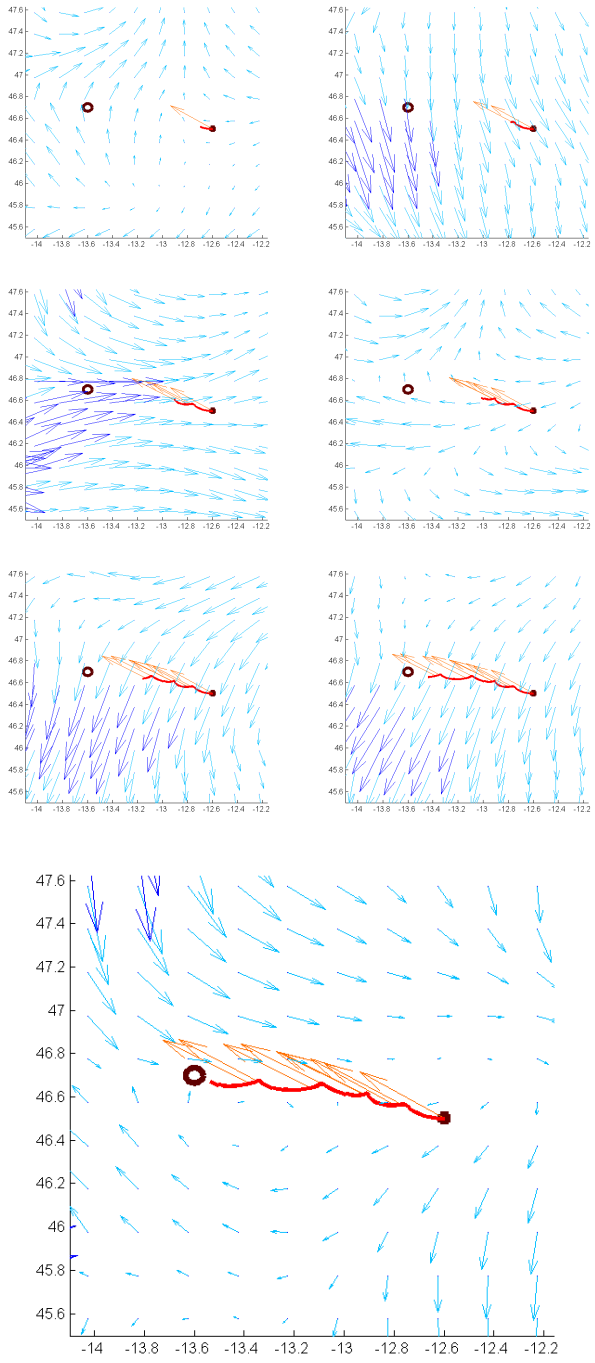

Fig. 6. Snapshots of the optimal trajectory and glider bearings at each surfacing, simulated for a 3-days period with time-varying currents (ocean currents that exceed the glider speed $v_g = 0.4\text{m/s}$ are highlighted) from the start point to the goal point .

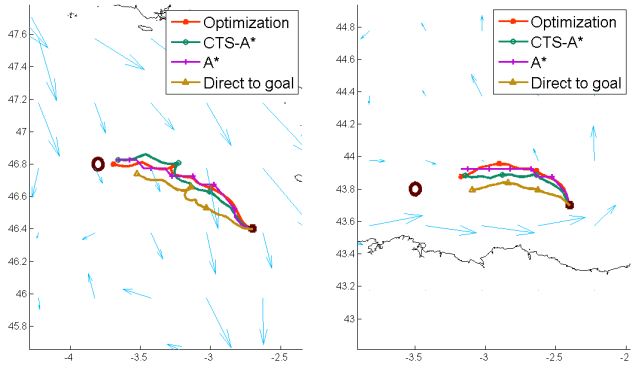

Fig. 7. Example of comparative of trajectories in two different missions of 3 days with glider speed of 0.4 m/s. Light blue arrows show the of ocean currents field. LEFT: Total distance = 95.3 km. Distance to reach the target: Optimization: 8.4 km; CTS-A*: 11.2 km; A*: 9.9 km; Direct to goal: 22.5 km. RIGHT: Total distance = 89.3 km. Optimization: 27.7 km; CTS-A*: 29.9 km; A*: 29.6 km; Direct to goal: 32.8 km.

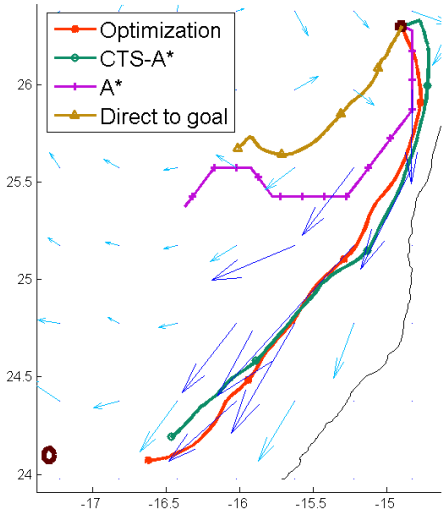

Fig. 8. Example of comparative of trajectories in a mission of 4 days. Light blue arrows show the of ocean currents field and blue arrows the ocean currents that exceed the glider speed (0.4 m/s). Total distance = 344.6 km. Distance to reach the goal point: Optimization: 68.9 km; CTS-A*: 85.1 km; A*: 169.4 km; Direct to goal: 217.6 km.

heuristic, an optimistic estimation of the combined glider-current velocity is computed, allowing to prune non promising trajectories.

In practice, we have observed that it suffices to divide the trajectory in a single turning point (one node). This is a direct consequence of the short path planned, since in a 4-day journey a glider might travel up to approximately 100-150km.

B. Optimization

In this phase, the algorithm takes the initial bearings and applies successive glider stints simulations trying to minimize the distance to the target from the end of trajectory as cost function.

Fig. 15 shows how this new aproach find a trajectory to the waypoint avoiding the coast in the same situation where the previous version hadn't found a good solution (Fig. 9).

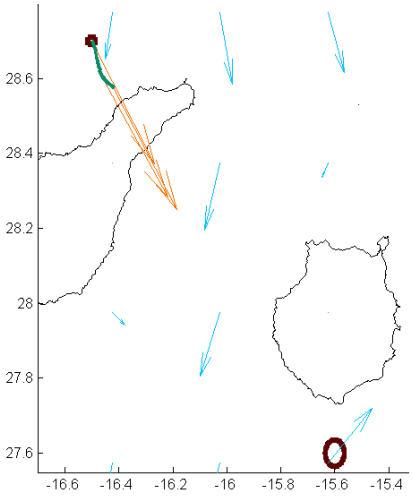

Fig. 9. Response in the presence of obstacles for the optimization method. The path ends after 4 days. Generated trajectories by glider bearings → at each surfacing, with time-varying currents → (ocean currents that exceed the glider speed $v_g = 0.4\mathrm{m/s}$ are highlighted →) from the start point ● to the goal point ○. ends after 4 days period

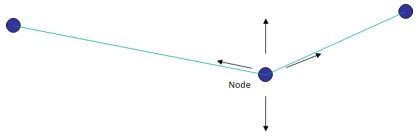

Fig. 10. Flexible location of the division point (node) between stages.

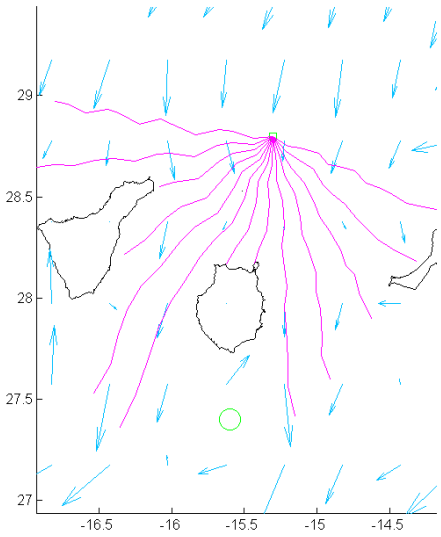

Fig. 11. First level of of the initialization process: Radial vectors emanating from the starting point.

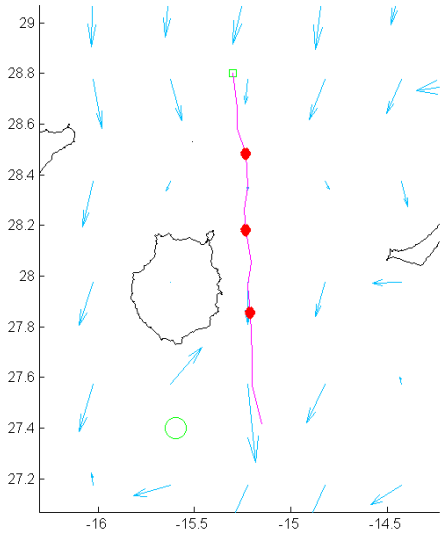

Fig. 12. Second level of of the initialization process: Selection of candidate nodes.

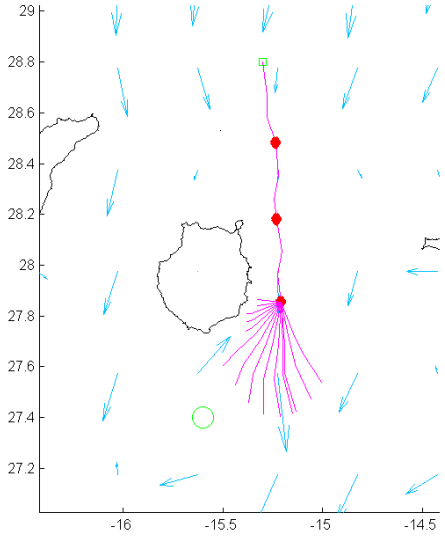

Fig. 13. Third level of of the initialization process: Radial vectors emanating from each candidate node.

IV. EXPERIMENTAL RESULTS

We have carried out several simulations for the path planner presented in this paper using Matlab® to validate the proposal and test its performance. The results have been compared with the ones obtained applying other methods: Direct to Goal, A*, CTS-A* and Optimization-based.

We have simulated different missions in the Canary Islands coast, using real ocean current maps from the ESEOO project model (ESEOCAN domain). This is a ROM model that gives hourly outputs structured in four 24h sets. The simulations described in this paper were configured for a glider speed of 0.2-0.4 m/s and a stint of 8 hours.

The general objective of the simulations have been to obtain the trajectory that leaves the vehicle closer to a goal point navigating for 4 days. For the methods based on the bearing

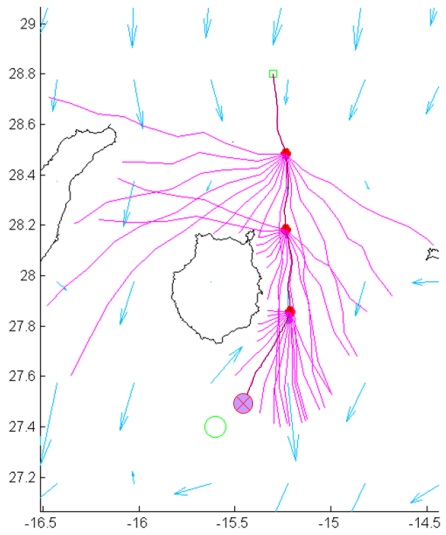

Fig. 14. Fourth level of of the initialization process: Selection of the best trajectory.

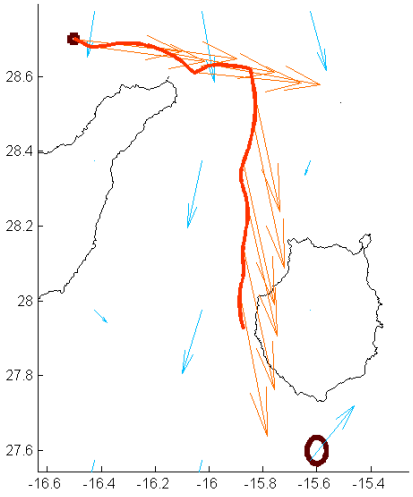

Fig. 15. Response in the presence of obstacles for the Optimization with Intelligent Initialization method. The path ends after 4 days. Generated trajectories by glider bearings → at each surfacing, with time-varying currents → (ocean currents that exceed the glider speed $v_g = 0.4\text{m/s}$ are highlighted →) from the start point ● to the goal point ◎.

optimization, this requires a total of 12 variables. Fig. 6 illustrates one example of the typical results obtained in these tests.

To validate the algorithm presented in this work, we have compared our results with the ones obtained by other algorithms used in the planning of trajectories for gliders. To compare the performance of each path planning method we have simulated and evaluated 45 cases. We have divided the cases in two situations and analyzed them separately. The first set of cases correspond to coastal trajectories while the second one includes only trajectories in offshore scenarios.

Two measures are computed for the comparison of the methods: path quality and computational cost. We have established as a quality measure, for the generated trajectories (the lower the better), the remaining distance from the final glider position to the target point.

We should comment here that the A* results require a special consideration, since the method used in the trajectory generation produces unrealistic non-constant surfacing times that are dependent on the grid size. That is to say, the surfacing points in A* do not correspond with the surfacing points of the glider.

The computational cost is also an important factor to be considered, as sometimes it is necessary to obtain a path in a few minutes. For example, when an unforeseen risky situation occurs while the glider is surfacing, a new bearing needs to be computed before the glider initiates a new transect.

Regarding the algorithms' parameters used in the comparison, we have selected the same equivalent discretization level for each method, when applicable. For example, the spatial grid for A* and CTS-A* is fixed to 1/20 degrees. The CTS-A* algorithm has been configured to use a division of 20° in the bearings rose. For our new approach we have used a division of 5° for the initialization phase, inserting a candidate turning point every 3 surfacings, the equivalent to one day of navigation.

Table I shows the mean and standard deviation of the difference of the remaining distance to reach the goal of each method with respect to our new approach. The global result for all cases and the mean in each environment (near the coast, offshore) is shown separately. The average distance traveled by the glider at 0.4 m/s has been 120 km. Table II shows the computing time for each method, measured on a Intel(R) Core(TM) 2 Quad processor computer running at 2.5 GHz. In the tables and graphics, we labeled as Init-Optim the Optimization with Intelligent Initialization method

Compared with the previous optimization-based method (Optim) it is observed that the new approach gets approximately the same results when no obstacles are present, while

TABLE I
DIFFERENCE OF THE REMAINING DISTANCE TO REACH THE GOAL WITH RESPECT TO THE OPTIMIZATION WITH INTELLIGENT INITIALIZATION METHOD. MEAN AND STANDARD DEVIATION WITHIN BRACKETS, BOTH IN km. SIMULATIONS RUN FOR A GLIDER SPEED $v_g = 0.4\text{m/s}$.

Scen	Optim	CTSA*	A*	Direct
Total	10.3 (21)	5.2 (6)	8.5 (18)	42.4 (46)
Coast	19.6 (26)	5.8 (7)	5.3 (7)	67.4 (39)
Ocean	0 (0)	6.5 (4)	9.1 (6)	13.6 (24)

TABLE II
COMPUTATIONAL TIME. MEAN AND STANDARD DEVIATION WITHIN BRACKETS, BOTH IN SECONDS. GLIDER VELOCITY AT 0.4 M/S AND 0.2 M/S.

Methods	0.4 m/s	0.2 m/s
Init-Optim	26 (10)	24 (12)
Optim	15 (11)	12.5 (10)
CTS-A*	477 (93)	105 (28)
A*	55 (11)	12 (4)
Direct to goal	<1 (0)	<1 (0)

TABLE III
DIFFERENCE OF THE REMAINING DISTANCE TO REACH THE GOAL WITH RESPECT TO THE INIT-OPTIM METHOD. MEAN AND STANDARD DEVIATION WITHIN BRACKETS, BOTH IN km. SIMULATIONS RUN FOR A GLIDER SPEED $v_g = 0.2\mathrm{m/s}$.

Scen	Optim	CTSA*	A*	Direct
Total	6.2 (13)	9.9 (10)	13.5 (49)	18.0 (29)
Coast	16.3 (17)	7.0 (7)	10.5 (13)	30.4 (29)
Ocean	0.2 (1)	11.5 (10)	15.3 (19)	10.5 (27)

it shows an important improvement when there is a need to avoid obstacles. Regarding A* and CTS-A* methods we have observed that, in general, we can obtain better quality in the path with less computational cost. On the other hand, we have verified that the computational cost of the new method when the route is free of obstacles is approximately half the value when the obstacles are present.

Fig. 16 shows two of the cases included in the previous analysis, where the trajectory is near the coastal areas. The distance required to reach the waypoint after 4 days is shown. It must be noted that the currents vary on time and only the last snapshot of them is shown in the figure. Fig. 17 shows the same case presented in Fig. 8 where the trajectory is free of obstacles. Here, it is observed that the new approach obtains results very similar to Optimization-based method.

To test the performance of the algorithms on adverse conditions, the simulation of the 45 cases were repeated using a glider at 0.2 m/s (Table III). The average distance traveled by the glider at this velocity has been 60 km. Table II shows the computing time for each method.

The basic version of the optimization method reduces the difference due to the fact that the obstacles are in the same point and the Optimization with Intelligent Initialization method covers less distance. A* and CTS-A* obtain worse results due to the use of discretization in their implementations and in some cases they are not able to avoid obstacles, so it has a high standard deviation. On the other hand, while the two versions of optimization keep their times, A* and its variant reduce notably their cost. In the first group, the process is the same, as they need to optimize the same number of variables, while in the second one the search area has less extension and the number of nodes visited is reduced.

Fig. 18 shows one of the cases include in the previous analysis where the trajectory is near the coastal areas. The distance required to reach the waypoint after 4 days is shown. In this case all methods goes directly to the land except our new approach.

Finally, the influence of some algorithm parameters has been analyzed. If we reduce the division of the bearing rose from $20°$ to $5°$ in Init-Optim, the results are improved in a 4% at a cost of duplicating the computational cost. Similarly, if we use a search grid of double resolution in A*, the results are improved in a 2%, but the computational cost is 5 times higher.

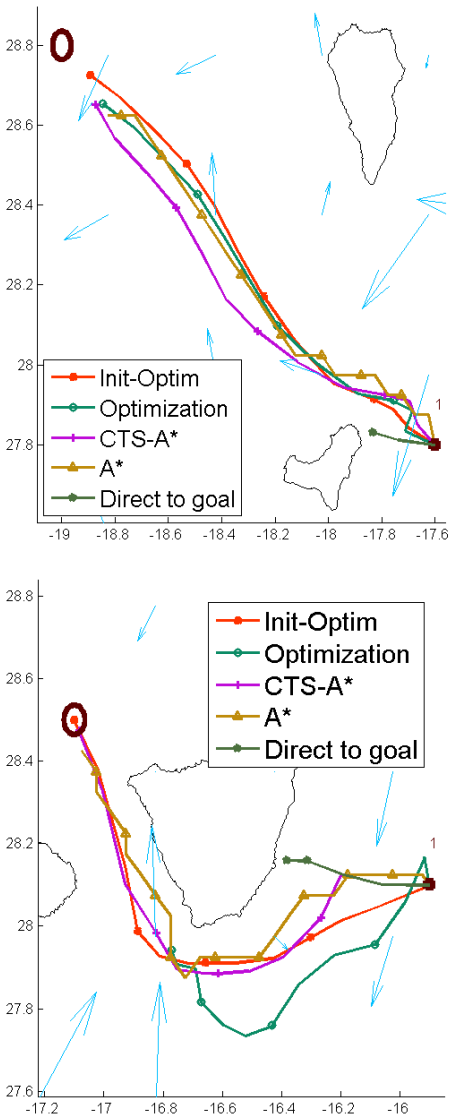

Fig. 16. Two comparatives of trajectories simulated near coastal areas for a 4-days period with time-varying currents → (ocean currents that exceed the glider speed $v_g = 0.4\mathrm{m/s}$ are highlighted →) from the start point ● to the goal point ○. Top graphic: Total Distance = 176.5 km. Distance remaining to reach the goal point: Init-Optim: 13.3 km; Optimization: 22.1 km; CTS-A*: 20.6 km; A*: 25.9 km; Direct to goal: 157.1 km. (stop in land). Bottom graphic: Total Distance = 125.8 km. Distance remaining to reach the goal point: Init-Optim: 0.0 km; Optimization: 69.7 km (stop in land); CTS-A*: 3.2 km; A*: 8.7 km; Direct to goal: 80.0 km. (stop in land).

V. CONCLUSIONS

We have described a novel path planning algorithm for gliders based on optimization that offers promising results in realistic simulations. The pattern of displacement of the gliders (the bearing can not be changed while submerged) allows to easily adapt our method to problems where there is a temporal discretization, in which the size of each time window coincides with the duration of the stints. In addition, our method uses a continuous representation of the bearings space using an optimization method and eliminating the problems discussed. Furthermore, it incorporates an initialization phase that allows for obstacle avoidance, at a negligible computational time penalty. This heuristic-guided process generates a coarse initial solution that is then refined using an optimization process. In sum, our method is suitable for dynamic scenarios with

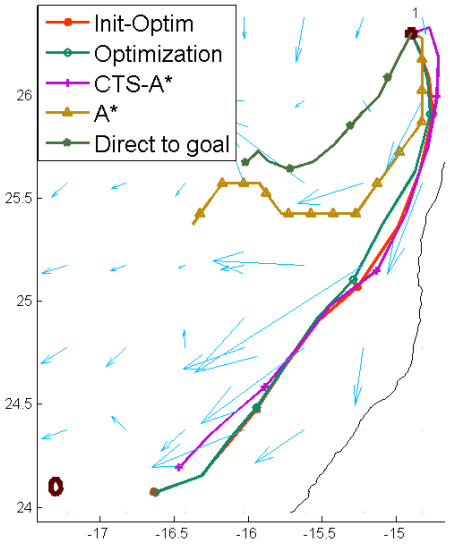

Fig. 17. Comparative of trajectories simulated in offshore area for a 4-days period with time-varying currents → (ocean currents that exceed the glider speed $v_g = 0.4\text{m/s}$ are highlighted →) from the start point ● to the goal point ○. Total Distance = 343.4 km. Distance remaining to reach the goal point: Init-Optim: 67.4 km; Optimization: 68.8 km; CTS-A*: 85.1 km; A*: 169.4 km; Direct to goal: 217.6 km.

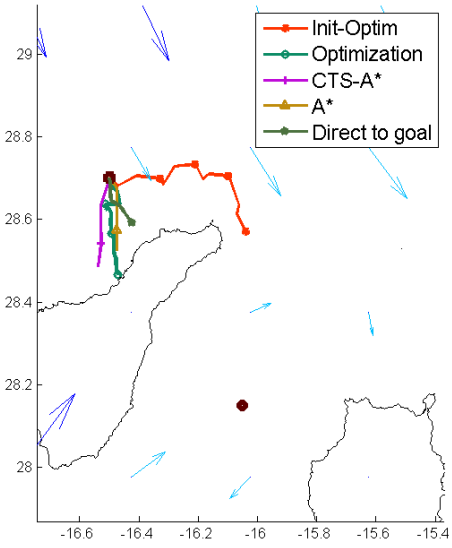

Fig. 18. Comparative of trajectories simulated in offshore area for a 4-days period with time-varying currents → (ocean currents that exceed the glider speed $v_g = 0.2\text{m/s}$ are highlighted →) from the start point ● to the goal point ○. Total Distance = 75.2 km. Distance remaining to reach the goal point: Init-Optim: 46.7 km; Optimization: 54.2 km (stop in land); CTS-A*: 60.7 km (stop in land); A*: 58.9 km (stop in land); Direct to goal: 61.3 km. (stop in land)

obstacles or forbidden areas, making it a practical tool for coastal environments.

The method shows a superior performance when compared with other alternative approaches. In general, classical A* or variants, like the CTS-A* algorithm analyzed here, do not find a path better than optimization methods. Notice that even a slightly improvement of $5\text{-}10\text{km}$ in the path cost is advantageous in many glider missions, e.g. it might reduce the economical cost of collection after the mission. Anyhow, it is in the computational cost where the latter clearly outperform the former.

Finally, the solution presented in this paper is valid for this particular problem, but would not have the same benefits if it is applied for path planning with other kind of vehicle, as long as there is no temporal discretization in their pattern of displacement.

A. Future works

In future research, we would like to incorporate 3D ocean current data and model the glider motion accordingly. Also we pretend validate these results in the navigation of a real glider.

ACKNOWLEDGMENT

The authors would like to thank Puertos del Estado for granting access to the ESEOO Regional Ocean Model.

REFERENCES

[1] A. Alvarez, A. Caiti, and R. Onken. "Evolutionary path planning for autonomous underwater vehicles in a variable ocean". *IEEE Journal of Ocean Engineering*, 29(2):418-429, 2004.

[2] Pradeep Bhatta, Edward Fiorelli, Francois Lekien, Naomi Ehrich Leonard. et al, "Coordination of an Underwater Glider Fleet for Adaptive Ocean Sampling", *in Proc. International Workshop on Underwater Robotics, Int. Advanced Robotics Programmed (IARP)*, Genoa, Italy, November 2005.

[3] K.P. Carroll, S.R. McClaran, E.L. Nelson, D.M. Barnett, D.K. Friesen, and G.N. William, "AUV path planning: an A* approach to path planning with consideration of variable vehicle speeds and multiple, overlapping, time-dependent exclusion zones", *in Proceedings of the 1992 Symposium on Autonomous Underwater Vehicle Technology*, pages 79-84, 1992.

[4] E.W. Dijkstra, "A Note on Two Problems in Connexion with Graphs", *NumerischeMathematik* 1, pages 269-271, 1959.

[5] E. Fernández Perdomo, J. Cabrera Gáamez, D. Hernáandez Sosa, J. Isern Gonzáalez, A. Domínguez Brito, A. Redondo, J. Coca, A. G. Ramos, E. Álvarez Fanjul, and M. García, "Path Planning for gliders using Regional Ocean Models: Application of Pinzon path planner with the ESEOAT model and the RU27 trans-Atlantic flight data", *in Proceedings of the OCEANS 2010 IEEE Sydney Conference and Exhibition*, 2010.

[6] B. Garau, A. Alvarez, and G. Oliver, "Path Planning of Autonomous Underwater Vehicles in Current Fields with Complex Spatial Variability: an A* Approach", *in Proc. 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 194-198.

[7] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths", *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100107, 1968.

[8] J. Isern-González, D. Hernández-Sosa, E. Fernández-Perdomo, J. Cabrera-Gámez, A.C. Domínguez-Brito and V. Prieto-Marañón, "Application of Iterative Optimization Algorithms to Trajectory Planning for Underwater Gliders", *in Proceedings of the Thirteen International Conference on Computer Aided Systems Theory (Eurocast 2011)*, Las Palmas de Gran Canaria, Spain, Feb. 2011.

[9] J. Isern-González, D. Hernández-Sosa, E. Fernández-Perdomo, J. Cabrera-Gámez, A.C. Domínguez-Brito and V. Prieto-Marañón, "Path planning for underwater gliders using iterative optimization", *IEEE International Conference on Robotics and Automation*, 2011.

[10] J. Isern-González, D. Hernández-Sosa, E. Fernández-Perdomo, J. Cabrera-Gámez, A.C. Domínguez-Brito and V. Prieto-Marañón, "Iterative Optimization-Based Path Planning with Variable Costs for Underwater Gliders", *OCEANS 11 IEEE Santander Conference*, 2011.

[11] Dov Kruger, Rustam Stolkin, A. Blum, and J. Briganti, "Optimal auv path planning for extended missions in complex, fast-flowing estuarine environments", *in Proceedings of the IEEE Int. Conf. on Robotics and Automation*, pages 4265-4270, 2007.

[12] S. M. LaValle, *Rapidly-exploring random trees: A new tool for path planning*, Iowa State University, 1998.

[13] M. B. Milam, K Mushambi and R. M. Murray, "New Computational Approach to Real-Time Trajectory Generation for Constrained Mechanical Systems", *in Conference on Decision and Control*, 2000.

[14] Javier Minguez and L. Montano, "Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios", *Robotics and Automation, IEEE Transactions on*, vol.20, no.1, pp. 45- 59, Feb. 2004.

[15] H. Moqin, C. D. Williams and R. Bachmayer, "Simulations of an Iterative Mission Planning Procedure for an Underwater Glider", *in Unmanned Untethered Submersible Technology*, 2000.

[16] C. Petres, Y. Pailhas, Y. Petillot, and D. Lane, "Underwater path planing using fast marching algorithms", *In Oceans 2005-Europe*, vol. 2, 2005, pp. 814- 819.

[17] C. Petres, Y. Pailhas, P. Patron, Y. Petillot, J. Evans, and D. Lane, "Path Planning for Autonomous Underwater Vehicles", *IEEE Transactions on Robotics*, 23(2):331-341, 2007.

[18] Dushyant Rao and Stefan B. Williams, "Large-scale path planning for Underwater Gliders in ocean currents", *in Australasian Conference on Robotics and Automation (ACRA)*, Sydney, December 2-4, 2009

[19] D.L. Rudnick, R.E. Davis, C.C. Eriksen, D.M. Fratantoni and M.J. Perry, "Underwater Gliders for Ocean Research", *Marine Technology Society Journal*, vol. 38, no. 1, 2004, pp. 48-59.

[20] R. Simmons and C. Urmson, "Approaches for heuristically biasing RRT growth", *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003, pp. 1178-1183.

[21] Michal Soulignac, Patrick Taillibert, and Michel Rueher, "Adapting the wavefront expansion in presence of strong currents", *In Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, pages 1352-1358, 2008.

[22] M. Soulignac, P. Taillibert and M. Rueher, "Time-minimal Path Planning in Dynamic Current Fields", *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, 2009.

[23] M. Soulignac, "Feasible and Optimal Path Planning in Strong Current Fields", *IEEE Transactions on Robotics*, vol. 27, no. 1, 2010, pp. 89-98.

[24] C. S. Tan, R. Sutton and J. Chudley, "An incremental stochastic motion planning technique for autonomous underwater vehicles". *In Proceedings of IFAC Control Applications in Marine Systems Conference*, pages 483-488, 2004.

[25] Laszlo Techy, C.A. Woolsey and K.A. Morgansen, "Planar Path Planning for Flight Vehicles in Wind with Turn Rate and Acceleration Bounds", *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, 2010.

[26] Ming-Cheng Tsou and Chao-Kuang Hsueh, "The Study of Ship Collision Avoidance Route Planning by Ant Colony Algorithm", *Journal of Marine Science and Technology*, vol. 18, no. 5, 2010, pp. 746-756.

[27] J. Witt and M. Dunbabin, "Go With the Flow: Optimal AUV Path Planning in Coastal Environments", *Australian Conference on Robotics and Automation*, 2008.

[28] Fumin Zhang, David M. Fratantoni, Derek Paley, John Lund and Naomi Ehrich Leonard, "Control of Coordinated Patterns for Ocean Sampling", *International Journal of Control, special issue on Navigation, Guidance and Control of Uninhabited Underwater Vehicles*, Vol. 80, No. 7, July 2007, pp. 1186-1199.

[29] W. Zhang, T. Inanc, S. Ober-Blbaum and J. Marsden, "Optimal Trajectory Generation for a Glider in Time-Varying 2D Ocean Flows Bspline Model", *IEEE International Conference on Robotics and Automation*, 2008.

Local robot navigation based on an active visual short-term memory

Julio Vega, Jose María Cañas, Eduardo Perdices

***Abstract*—Vision devices are today one of the most often used sensory elements in autonomous robots. Some of their hindrances are the difficulty in extracting useful information from the captured images and the small visual field of regular cameras. Visual attention systems and active vision may help to overcome them. This work proposes a dynamic visual memory to store the information gathered from a continuously moving camera onboard the robot and an attention system to choose where to look at with such mobile camera. The visual memory is a collection of relevant task-oriented objects and 3D segments, and its scope is wider than instantaneous field of view of the camera. The attention system takes into account the need to reobserve objects in the visual memory, explore new areas and test hypothesis about object existence in the robot surroundings. The system has been programmed and validated in a real Pioneer robot that uses the information in the visual memory for navigation tasks.**

***Index Terms*—Visual attention, object recognition and tracking, active vision, camera model, autonomous navigation.**

I. Introduction

COMPUTER vision is one of the most successful sensing modalities used in mobile robotics. It would seem to be the most promising one for the long term. Computer vision research is currently growing rapidly, both in robotics and in many other applications, from surveillance systems for security purposes to the automatic acquisition of 3D models for Virtual Reality displays. The number of commercial applications is also increasing, like traffic monitoring, parking access or face recognition. And we feel that it is well worth continuing with work on the long-term problems of making robotic vision systems.

Vision is the sensor whose main skill lies in giving information about which and where are the objects that the robot is finding over its path. And, although we must be wary when comparing a robot with a biological organism [Nehmzow, 1993], what is clear is that sight is the main sense used by animals when they want to move around the environment.

Humans have an active vision system. This means that we are able to concentrate on particular regions of interest in a scene, by movements of the eyes and head or just by shifting attention to different parts of the images we see. What advantages does this offer over the passive situation where visual sensors are fixed and all parts of images are equally inspected? First, some parts of a scene perhaps are not accessible to a single sensor are viewable by a moving device. In humans, movable eyes and head give us almost a full panoramic range of view. Second, by directing attention specifically to small regions which are important at various times we can avoid wasting effort trying always to understand the whole surroundings, and devote as much as possible to the significant part. For example, when attempting to perform a difficult task such as catching something, a human would concentrate solely on the moving object and it would be common experience to become slightly disoriented during the process.

Active vision can be thought as a more task driven approach than passive vision. With a particular goal in mind for a robot system, an active sensor is able to select from the available information only that which is directly relevant to a solution, whereas a passive system processes all of the data to construct a global picture before making decisions; in this sense it can be described as data driven.

The emerging view of human vision as a *bag of tricks* [Ramachandran, 1990]; a collection of highly specialised pieces of *software* running on specialised *hardware* to achieve vision goals, rather than a single general process, seems to fit in with active vision ideas when a similar approach is adopted in artificial vision systems. High-level decisions about which parts of a scene to direct sensors towards and focus attention on them can be combined with decisions about which algorithms or even which of several available processing resources to apply to a certain task. The flexibility of active systems allows them to have multiple capabilities of varying types which can be applied in different circumstances.

In this work we describe an overt attention system for a mobile robot endowed with a pan-tilt camera, whose will let it to find paper arrows on its surroundings and navigate through the 3D-space avoiding obstacles. This system performs an early segmentation on color space to select a set of candidate objects. Each object enters a coupled dynamics of life and salience that drives the behavior of the attention system over time. That way, our system will continuously keep relevant objects around the robot -such as arrows or parallelograms- in its visual short-term memory and it will know where they are located.

In the next section some related works about vision based navigation and attention systems are described as context of our proposal. Our system description has been divided in two sections, one explaining the visual memory and another one describing the attention subsytem. Some experiments have been carried out with a real robot to validate our approach, they are commented in section V. We end this paper with some conclusions and future lines.

All of them are with University of Rey Juan Carlos.

E-mails: julio.vega@urjc.es, jmplaza@gsyc.es, eperdices@gsyc.es

II. Related works

Vision has been used in robotics almost from its beginning. In the last years its use is increasing, mainly because of the reduction in the camera price, the available computing power and the potential of cameras as source of information about robot surroundings. Many issues have been taclked in the intersection of computer vision and robotic fields. For instance, vision-based control or navigation, vision-based map building and 3D representation, vision-based localization, object recognition, attention and gaze control among others. We will review some examples here.

Regarding vision based control and navigation, Remazeilles et al.[Remazeilles *et al.*, 2006] presented the design of a control law for vision-based robot navigation. The particularity of this control law is that it does not require any reconstruction of the environment, and it does not force the robot to converge towards each intermediary position in the path.

Recently, Srinivasan [Srinivasan *et al.*, 2006] presented a new system to increase accuracy in the optical flow estimation for insect-based flying control systems. A special mirror surface is mounted in front of the camera, which is pointing ahead instead of pointing to the ground. The mirror surface decreases the speed of motion and eliminates the distortion caused by the perspective. Theoretically, the image should present a constant and low velocity everywhere, simplifying the optical flow calculation and increasing its accuracy. Consequently, the system increases the speed range and the number of situations under which the aircraft can fly safely.

Badal [Badal *et al.*, 1994] reported a system for extracting range information and performing obstacle detection and avoidance in outdoor environments based on the computation of disparity from the two images of a stereo pair of calibrated cameras. The system assumes that objects protrude high from a flat floor that stands out from the background. Every point above the ground is configured as a potential object and projected onto the ground plane, in a local occupancy grid called Instantaneous Obstacle Map (IOM). The commands to steer the robot are generated according to the position of obstacles in the IOM.

Goldberg [Goldberg *et al.*, 2002] introduced a stereo vision-based navigation algorithm for the rover planetary explorer MER, to explore and map locally hazardous terrains. The algorithm computes epipolar lines between the two stereo frames to check the presence of an object, computes the Laplacian of both images and correlates the filtered images to match pixels from the left image with their corresponding pixels in the right image. The work also includes a description of the navigation module GESTALT, which packages a set of routines able to compute actuation, direction, or steering commands from the sensor information.

Regarding map building and self-localization maybe the most succesful approach in last years has been the MonoSLAM from A. Davison [Gerardo Carrera y Davison, 2011]. The detection of relevant points in the image and a fast Extended Kalman Filter allow the system to continuously estimate the camera 3D position and orientation and the 3D position of such points. The localization results are impressive. The quality of the maps, mainly as collection of 3D points, was not so good at the beginning but they have improved it even with dense maps in real time [Newcombe y Davison, 2010].

Mariottini and Roumeliotis [Mariottini y Roumeliotis, 2011] presented a strategy for active vision-based localization and navigation of a mobile robot with a visual memory within a previously-visited area represented as a large collection of images. This strategy can disambiguate the true initial location among possible hypotheses by controlling the mobile observer across a sequence of highly distinctive images, while concurrently navigating towards the target image.

Gartshore [Gartshore *et al.*, 2002] developed a map building framework and a feature position detector algorithm that processes images on-line from a single camera. The system does not use matching approaches. Instead, it computes probabilities of finding objects at every location. The algorithm starts detecting the objects boundaries for the current frame using the Harris edge and corner detectors. Detected features are back projected from the 2D image plane considering all the potential locations at any depth. The positioning module of the system computes the position of the robot using odometry data combined with image feature extraction. Color or gradient from edges and features from past images help to increase the confidence of the object presence in a certain location. Experimental results tested in indoor environments set the size of the grid cells to 25 mm 25 mm. The robot moved 100 mm between consecutive images.

In autonomous robots it is important to perform a visual attention control. The cameras of the robots provide a large flow of data you need to select what is interesting and ignore what does not; this is the main goal of visual attention. There are two aspects of visual attention: *overt attention* and *covert attention*. The aim of covert attention [Tsotsos *et al.*, 1995; Itti y Koch, 2001], [Marocco y Floreano, 2002] is to select interesting information within an image. Overt attention selects from the environment surrounding the robot, beyond the field of view, those objects of interest, and it looks at them [Cañas *et al.*, 2008].

The visual representation of the interesting objects around the robot can improve the quality of the robot's behavior and the ability to handle more information when making their decisions. This poses a problem when those objects are not in the immediate field of vision. To solve this problem, some studies used omnidirectional vision, in others using a regular camera and a mechanism for overt attention [Itti y Koch, 2001; Zaharescu *et al.*, 2005], which enables fast-to-take samples of a very broad area of interest. The use of a camera in motion to facilitate object recognition was proposed by [Ballard, 1991], and has been used, for example, to distinguish between different forms in the images [Marocco y Floreano, 2002].

One of the concepts widely accepted in the work area is the *salience map*. It is found in [Itti y Koch, 2001], as a covert visual attention mechanism, independent of the particular task to be performed and composed by all visual stimuli that attract attention from the scene. In such work is considered purely a form of "bottom up", where, as we see in Figure 1 in each iteration the different scene-descriptive maps (as colors,

intensities or directions) compete between each other. Then, they merged into conspicuity maps (one for each feature) and eventually will form a unique and representative salience map.

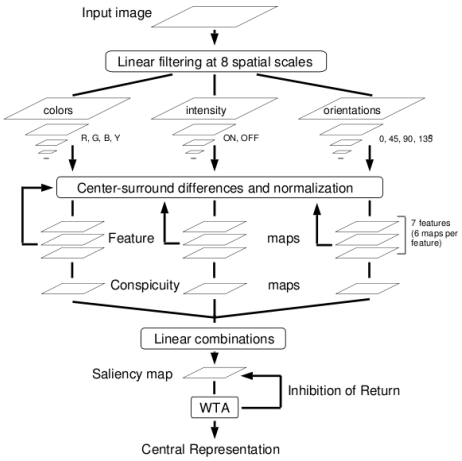

Fig. 1. Salience map as proposed by Itti

Hulse [Hulse *et al.*, 2009] presented an active robotic vision system based on the biological phenomenon of inhibition of return, used to modulate the action selection process for saccadic camera movements. They argued that visual information has to be subsequently processed by a number of cortical and subcortical structures that place it: 1) in context of attentional bias within egocentric salience maps; 2) the aforementioned IOR inputs from other modalities; 3) overriding voluntary saccades and 4) basal ganglia action selection. Thus, biologically there is a highly developed, context specific method for facilitating the most appropriate saccade as a form of attention selection.

Arbel and Ferrie presented in [Arbel y Ferrie, 2001] a gaze-planning strategy that moves the camera to another viewpoint around an object in order to recognize it. Recognition itself is based on the optical flow signatures that result from the camera motion. The new measurements, accumulated over time, are used in a one-step-ahead Bayesian approach that resolves the object recognition ambiguity, while it navigates with an entropy map.

III. 3D Visual Memory

The goal of our system is doing a visual track of the various basic objects in the scene surrounding the robot. Therefore, it must detect new objects, share the focus between them and removing them from the memory once they have disappeared.

The first stage of the system is the 2D analysis, which detects 2D segments present in the current image. Then the 3D reconstruction algorithm places these objects in 3D space according to the *ground-hypothesis* (that is, we suppose that all objects are flat on the floor). And finally, the 3D memory system stores their position in 3D space, calculates perceptual hypotheses and generates predictions of these objects in the current image perceived by the robot.

In this section we will see the various components of our 3D visual memory system, implemented in conjunction with the attention system. Some previous versions of the system are described in [Vega y Cañas, 2009; 2010]. First, an object detector is responsible of identifying basic shapes in the current image. Second, the prediction mechanism will allow the system to predict how the stored items will appear in next images, reducing the computational cost of image processing for them. Third, a 3D reconstruction block is responsible to obtain 3D instantaneous information from objects in the current image and merging them with the objects already stored in the visual memory.

A. 2D Image Processing

The main objective of this part of the system is to extract 2D straight segments as a basic primitive. These primitives are handled by the 3D reconstruction module. The 2D detection module, in turn, is connected to the 3D memory directly, in order to save computation time of reconstruction of objects that may already be stored in memory. It also can be used to confirm/refute the stored instantaneous objects. The current image is useful to confirm structures previously observed partially.

The first step to simplify the image is an edge filter, by using the Canny algorithm. Subsequently we apply the Hough transform to extract only straight segments. To implement these techniques, we use the OpenCV library. In the Figure (2) we see the reconstruction of 3D segments before and after of Hough postprocessing.

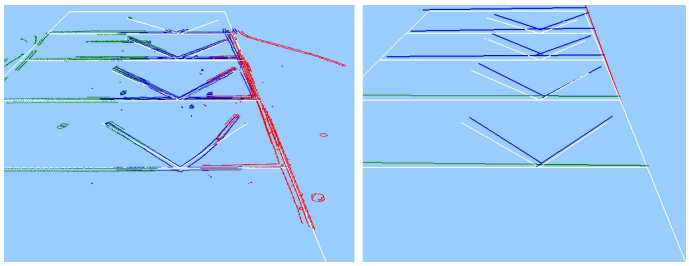

Fig. 2. 3D segments reconstruction, before and after postprocessing

B. Predictions

The 2D analysis system is connected directly to the 3D visual memory to alleviate the computational cost due to image analysis. So before extracting features of the current image, the system makes the prediction of those objects stored in the 3D memory which should be visible from the current position.

We have used our library called Progeo, which provides *projective geometry* capabilities given a calibrated camera. So each 3D visible object is stored and made its projection on the image plane (see Figure 3). The system refutes/corroborates such segments predicted, comparing one of these segments with those obtained by the Hough Transform. This comparison leads to three sets of segments, as seen in Figure 4.

C. Instantaneous reconstruction with 3D segments

The above mechanism extracts a set of 2D segments which must be located in 3D space. To do this, and as we have

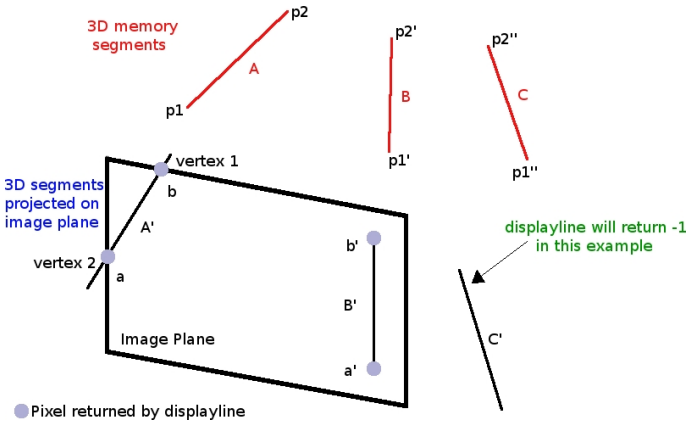

Fig. 3. Segments projection onto the image plane

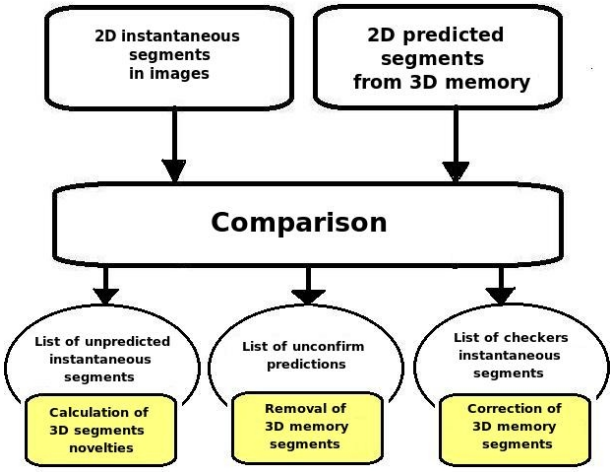

Fig. 4. Match between predicted and instantaneous segments

already mentioned, we rely on the idea of *ground-hypothesis*. Since we have one camera, we need a restriction which will enable to estimate the third dimension. We assume that all objects are flat on the floor.

Once we have the 3D objects, and before inclusion in the 3D memory, post-processing is needed to avoid duplicates in memory due to noise in the images. This postprocessing compares the relative position between segments, as well as its orientation and proximity. The output is a set of 3D segments situated on the robot coordinate system. Figure 5 shows the 3D scene with objects reconstructed by the system, the segments detected in the current image and the segments predicted from such a position.

We use four coordinate systems to define the geometric model, as we can see in Figure 6:

- The absolute coordinate system whose origin lies somewhere in the world where the robot is moving.
- The system located at the base of the robot. The robot odometry gives its position and orientation with respect to the previous system.
- The system relative to the base of the pan-tilt unit to which the camera is attached to. It has its own encoders for its position at any given time, with pan and tilt movements with respect to the base of the robot.
- And finally we have the coordinate system of the camera itself, displaced and oriented in a particular mechanical axis from the pan-tilt unit.

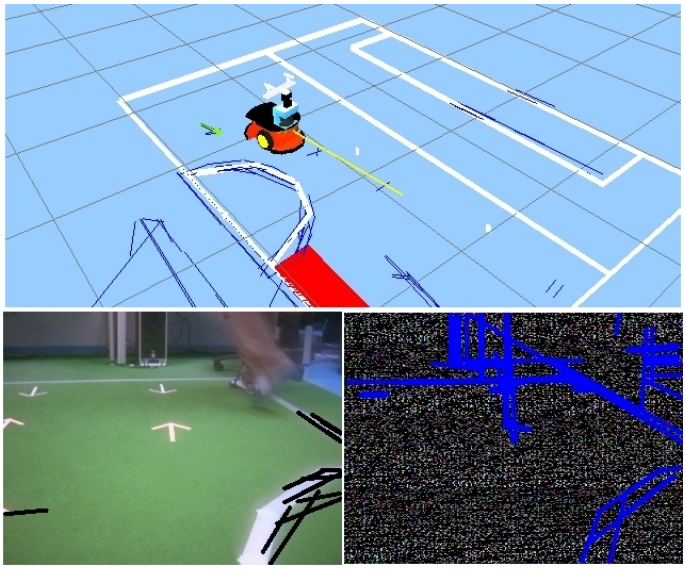

Fig. 5. 3D Scene Reconstruction, predicted and instant segments

D. Inserting segments into the 3D visual memory

3D memory comprises a dynamic set of lists which stores information about the different types of elements present in the scene (position, type or color). The most basic form of structure is the segment. Thanks to the memory we can establish relationships between them to make up more complex elements such as arrows, parallelograms, triangles, circles or other objects.

To store a segment we have a structure called *Segment3D*, consisting in a start and end point and a pointer to other possible structures of which it can be part of: Arrow3D or Parallelogram3D.

Incorporating 3D memory segment basically consists of comparing each segment individually calculated in the snapshot with those already stored. In case of nearby segments with similar orientation, the system combines these segments into a new one taking the longest length of its predecessors, and the orientation of the more recent, as probably it is more consistent with reality (the older ones tend to have more noise due to errors robot odometry).

To make this fusion process computationally lighter, the system has a segment cache with only the segments close to the robot (in a radius of around 4 m.). Its implementation is basically a dynamic list of pointers to these segments. The system always works with subsets of features, which are pointers to the overall 3D memory elements.

E. Predictions: deletion and correction of segments

As mentioned before, the 2D analysis system returns different subsets of segments, as the result of comparison between

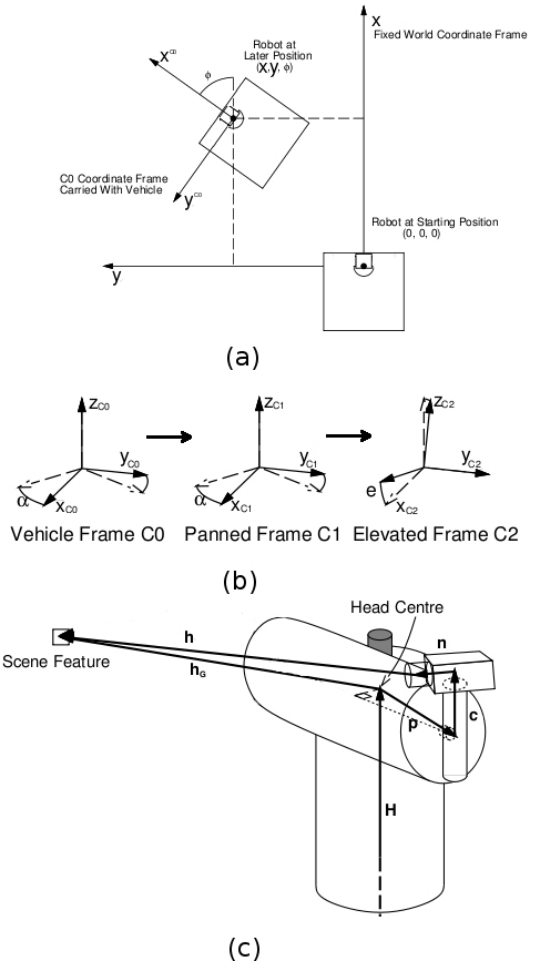

Fig. 6. Coordinate systems used to define the geometric model

instant and predicted segments from 3D memory.

If a segment is identified in the current image and it does not match the predictions, the system creates a new one in 3D which might replace an existing one (replacement or correction) under certain restrictions. To reflect this process, system has a parameter called *uncertainty* which will increase as the time segment remains in memory.

The element deletion process is based on the same principle, but here there are more rigorous restrictions. So, the replacement process is a priority compared to deletion.

F. Perceptual hypothesis generation

Our object model consists of a set of segments whose vertices can belong to more abstract structures like parallelograms. The vertices are labeled with the number of segments that are tied to them. This requires an object model for cases in which certain vertices are not visible at any given time. For instance, for parallelograms the minimum number of visible vertices is small, with three points we are able to estimate the fourth one.

The segments and their corresponding vertices are used to detect parallelograms checking the connection between them and the parallelism conditions. The analysis of the angles formed by each segment provides information about how the segments are connected to each other. In addition, this feature can be used to merge incomplete or intermittent segments. Similarly, we can extract the position of a possible fourth vertex using the information about other edges and/or possible parallelogram vectors. This capability makes our algorithm robust against occlusions, which occur frequently in the real world.

Figure 7-b, c illustrates an example of occlusion that is satisfactorily solved by our algorithm. The results of reconstruction of parallelograms can be seen in Figure 7-a. In this situation, we have a collection of parallelograms spread on the floor. The robot, after several snapshots, captures what is around itself: those parallelograms, and noise extracted by the segmentation algorithm. However, our parallelogram hyphotesis generation is able to extract only the real parallelograms, avoiding such noise.

Similarly, we can abstract other abstract objects such as arrows.

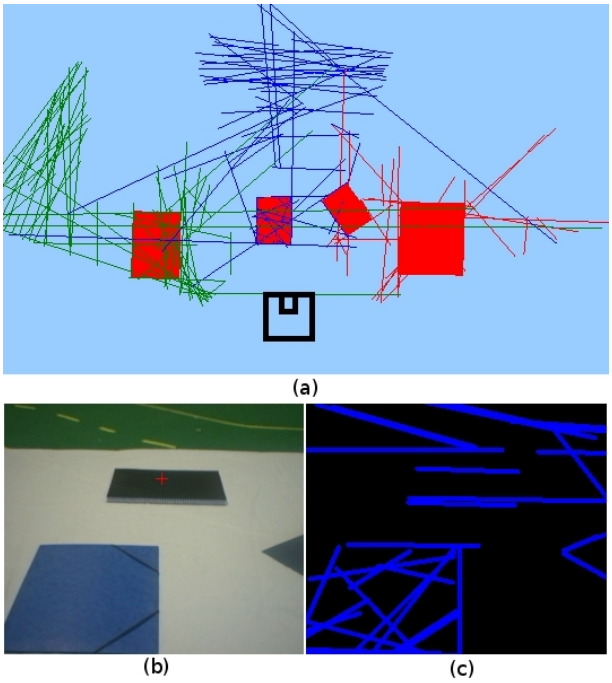

Fig. 7. Generation of hypothesis: parallelogram with occlusion

IV. VISUAL ATTENTION SYSTEM

In the previous section we have described in detail the operation of placing objects on the robot 3D visual memory. Now we will describe the visual attention mechanism implemented based on two of object attributes: *salience* and *life*. The salience is used for deciding where to look at in every moment, while life is the mechanism for forgetting an object, deleting it from memory, when it has disappeared from the scene.

In addition, we have designed a mechanism to control the camera movements, to track objects, and another mechanism to explore new unknown areas from the scene.

A. Gaze control: salience dynamics

Some sort of decision-making mechanism to indicate to the system where to look at in the next instant. Each object in the visual memory has its coordinates in the scene representation. It is desirable to control the movement of the pan-tilt unit to direct the focus to that position periodically in order to reobserve that object. To control the movement of the pan-tilt unit, we introduced the dynamics of salience and attention points. They mainly represent the detected objects in the scene. Each one contains a position in the 3D scene (X, Y, Z), which is translated into mechanical commands to the pan-tilt unit in order to direct the focus at that point.

Anything that attracts attention or stands in a given situation is salient. The focus may be changing over time to look at all the salient points. In this system salience indicates how attention selects the next object to be visited. Each memory element has an associated salience, which grows over time and vanishes every time that element is visited. The focal point with highest salience will be the next to be visited. If the salience is low, it will not be visited now.

$$Salience(t) = \begin{cases} Salience(t-1) = 0 & \text{if object attended} \\ Salience(t-1) + 1 & \text{otherwise} \end{cases}$$

When a point is visited, its salience is set to 0. A point that the system has not visited recently calls more attention than one which has just been attended. The system is thus similar to the behavior of a human eye, as pointed by biology studies [Itti y Koch., 2005]: when the eye responds to a stimulus that appears in a position that has been previously treated, the reaction time is usually higher than when the stimulus appears in a new position.

The designed algorithm allows the system to alternate the focus of the camera between the different objects in the scene according to their salience. In our system, we consider that all objects have the same preference of attention, so all of them are observed during the same time and with the same frequency. If we assigned different priorities to the objects, we could establish different rates of growth of salience. This would cause the pan-tilt unit to pose more times in the object whose salience grows faster.

We assume that a detected object will be found near the location where it was previously.

B. Tracking of a focused object

When the look-sharing system chooses the attention point of a given object, it is going to be looking for a certain time (3 seconds), tracking it if it moves spatially. For this tracking, and to avoid excessive oscillations, we use a *P-controller* (Fig. 8) to control the speed of the pan and tilt and thus continually focus that object on the image center. This driver orders high speeds to the pan-tilt unit if the focus of attention is far from the predicted position; or lower speeds if it requires small corrections. The controller follows next equations.

$$v(Pan) = \begin{cases} 0 & \text{if} \quad \epsilon < 0.3 \\ K_p \cdot (P_t - P_a) & \text{if} \quad 0.3 \leq \epsilon < M_p \\ M_p & \text{if} \quad M_p < \epsilon \end{cases}$$

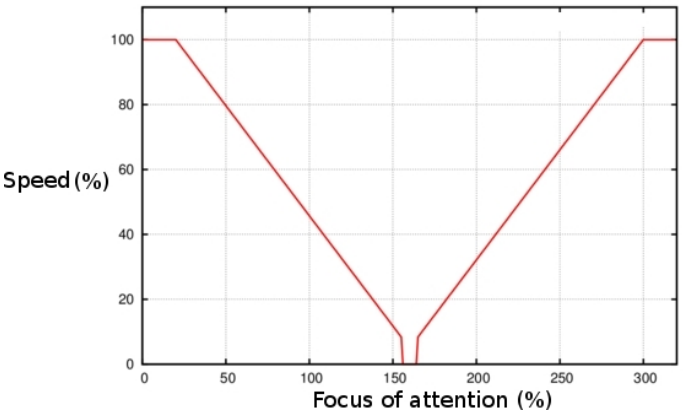

Fig. 8. P-controller mechanism

$$v(Tilt) = \begin{cases} 0 & \text{if} \quad \epsilon < 0.1 \\ K_p \cdot (T_t - T_a) & \text{if} \quad 0.1 \leq \epsilon < M\_t \\ M_t & \text{if} \quad M_t < \epsilon \end{cases}$$

Where: K_p is the P control gain, T_t is the Tilt of the target, T_a is the actual Tilt, P_t is the Pan of the target, P_a is the actual Pan, M_t is the maximum Tilt and M_p is the maximum Pan.

C. Exploring new areas of interest

At any time, it may be interesting to look for new objects in the scene. For that search our system will insert periodically (every *forcedSearchTime*) scanning points with high salience in memory. This search is especially interesting at the beginning, when there are many unknowns areas of the scene where there can be objects of interest.

The scanning points can be of two types: random and systematic ones. The first type are assigned uniformly random coordinates *(pan, tilt)* within the pan-tilt range (pan = [-159, + 159], tilt = [-31, +31]). Systematic scanning points follow a regular pattern to finally cover the whole scene around the robot.

The attention points, whatever their type, have a high initial salience in order to be quickly visited with the camera and thereby check whether any object of interest is found around it. In such a case that object will enter into the memory and into the gaze sharing module.

As we discover objects, the desire to explore new areas will decrease in proportion to the number of already detected objects.

D. Representation of the environment: life dynamics

As already discussed in previous sections, our visual attention system guides the search and tracking of objects within the scene. The objects may eventually disappear from the scene, and then they should be removed from the memory in order to maintain coherence between the representation of the scene and the reality.

To forget such old elements, we have implemented the *life* dynamics. With this mechanism the system can know whether

an object has left the scene or it is still there. The life operation is the reverse of the salience, that is, a frequently visited object will have more life than one less visited. When the life of an object is below a certain threshold, it will be discarded.

Every time the attention system visits an object, its life increases one point, with a maximum limit to provide saturation. The life of unobserved objects will decrease over time. Thus, when the life of an object exceeds a certain threshold, which is still on the scene, whereas when is below it is gone.

$$Life(t) = \begin{cases} min(MAX_{LIFE}, Life(t-1) + \Delta) \\ \qquad \text{if object observed} \\ Life(t-1) - 1 \\ \qquad \text{otherwise} \end{cases}$$

E. Attentive system operation

The objects of the environment surrounding the robot guide the movements of the camera. So the mechanism of attention is so far *bottom-up*. Besides, the underlying mechanism of *top-down* attention is that existing relevant objects are only those that have a certain appearance given by the task at hand: in our examples, parallelograms or arrows. This tendency to look at objects with a certain aspect is similar to the bias detected by ethologists in animals with respect to certain stimuli [Arkin, 1998].

The visual attention system presented here has been implemented following a *state-machine* design, which determines when to execute the different steps of the algorithm. Thus, we can distinguish four states:

- Discuss next goal (state 0).
- Saccade is completed (state 1).
- Analyze image (state 2).
- Tracking object (state 3).

Periodically the system updates the salience and life attributes of the objects that have already stored in memory following previous equations. It checks whether any of them is already outdated, because its life is below a certain threshold. If not, it increases its salience and reduces its life.

Based on the initial state (or state 0), the system asks whether there is any attention point to look at (in case we have an object previously stored in memory) or not. If so, it goes to state 1. If not, it inserts a new scanning attention point into memory and goes back to state 0.

In state 1 the task is to complete the movement towards the absolute position specified in state 0. Once there, we go to stage 2 where we will analyze whether there are relevant objects or not. In any case, it passed the state 0 and back again.

V. Experiments

Our experiments were performed with a real Pioneer 2DX robot (by MobileRobots), endowed with a Dell laptop with an Intel Centrino processor at 1.7 GHz. and Linux Ubuntu 8.04 (hardy) as operating system. It also has installed a pan-tilt unit (46-17.5 Unit Pantilt Directed Perception) with a pan range of [180, -180] and a tilt range of [31, -80] degrees. It works at a minimum speed of 0.0123 deg./sec. and a maximum speed of 300 deg./sec. on both axes. The pan tilt unit has a firewire iSight camera (by Apple) on top, with autofocus and a field of view of 60 and 40 degrees in horizontal and vertical respectively. The power to the pan-tilt unit is supplied by the base of the robot, and it is serial-port commanded.

A. 3D floor reconstruction

In this first experiment the robot has no knowledge of the environment. Initially, and as already mentioned, it did a thorough systematic search for information from the environment. The system commanded saccades to the pan-tilt. These movements are short, accurate and fast, just enough time to examine whether there is any interesting object in the current image received from the camera or not. After a certain time, the system begins to detect segments (see Figure 9).

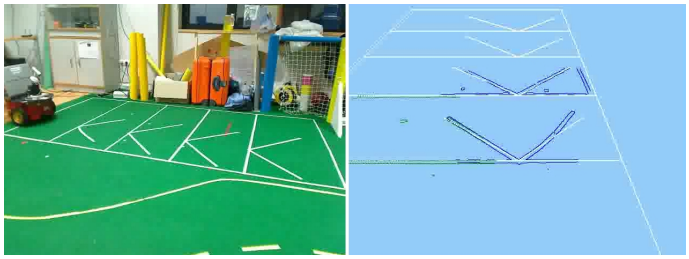

Fig. 9. Land lines reconstruction. Initial stage

After several glimpses the robot is able to plausibly reconstruct the detected segments along its path (see Figure 10). The visual memory periodically performed some post-processing by which unique and refined segments were obtained. In this experiment they perfectly fit those in reality (Figure 2).

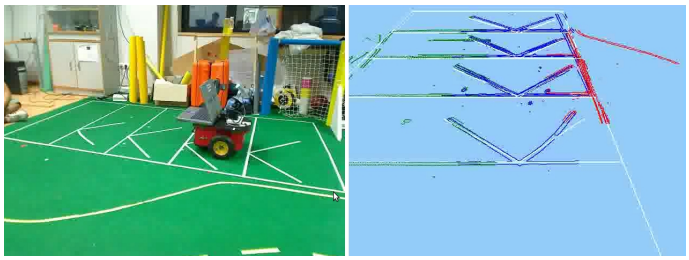

Fig. 10. Land lines reconstruction. Final stage

B. Parallelograms

In the second experiment, besides finding segments of the environment, the system can abstract parallelograms given the characteristics of all segments in the scene.

The *forced-SearchTime* period was 5 seconds, so every 5 seconds new exploring scanning points where inserted in the visual memory. This process is repeated some times, until the robot begins to detect objects of interest in the scene (see Figure 11). When it begins to have several elements

(parallelograms) in memory (see Figure 12), the *forced-SearchTime*is increased. This feature allows to look longer at already detected objects and less to explore new areas or search for new objects. With this increased time finding new objects will become increasingly rare as we have already detected more and more items.

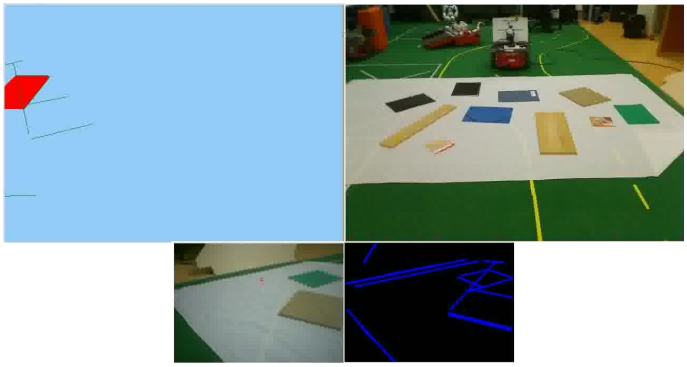

Fig. 11. Parallelograms recognition. Initial stage

Through this mechanism, the system finds step by step almost all the items in the scene (note that some objects cause problems because of their texture).

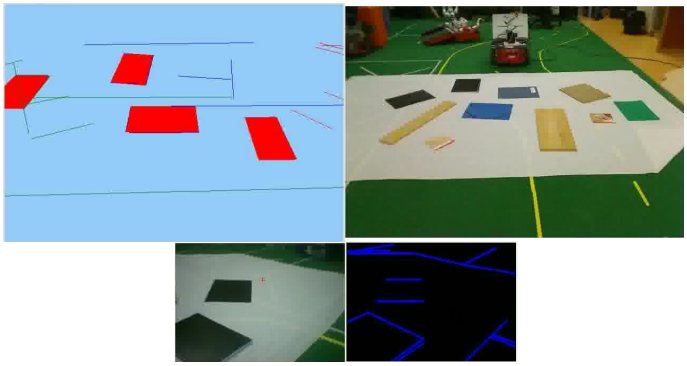

Fig. 12. Parallelograms recognition. Final stage

In Figure 13 the mode of action of the two competing dynamics, salience and life, is shown. It corresponds with a situation where the system has detected two elements. We can see in Figure 13-a how the salience of both evolve. When the system is following an item (blue) its salience decreases, while the other item stored in memory (red) increases until it wins the competition and forces the system to look at it.

The evolution of life on both objects, when both remain on the scene, is shown in Figure 13-b. Its operation is inverse to the salience, that is, every time the system visits an item, its life is increased one point, with a maximum limit score of 100 points to provide saturation.

Figure 13-c reflects a situation in which we occluded one of the two elements, so that the system fails to detect it as such and, therefore, its life begins to fall. When its value is below a certain threshold, the object is discarded and not re-visited.

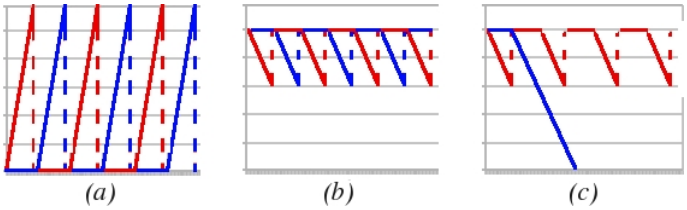

Fig. 13. Time evolution of the salience (a), Life (b) and Life of a disappearing object

C. Arrows as navigation landmarks

In this experiment we rely on the same ideas given above, but in this case we focus on the recognition of arrows in the environment, and the use of this item as a mark of direction for robot navigation. Figure 14 shows when the robot recognizes the arrow as such, having been previously detected the segments which compound it.

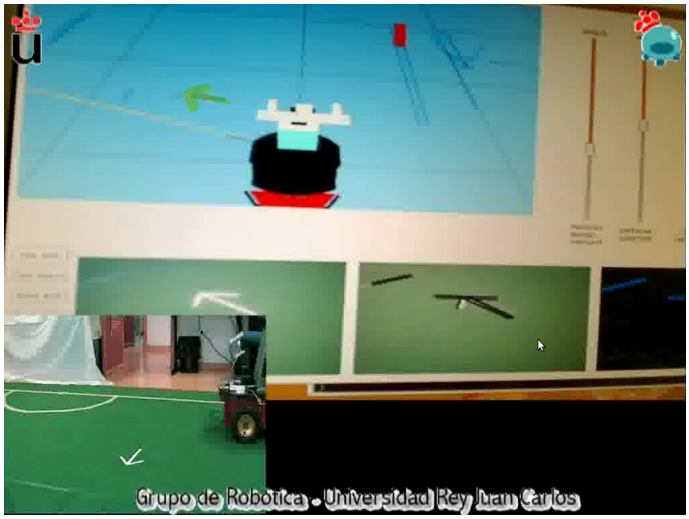

Fig. 14. Recognition of arrows as a mark of direction

Given the characteristics of an arrow, the system is able to abstract the concept *arrow* and represent it as such in the 3D memory (see the green arrow showed in Figure 14). Also, once detected, it automatically guides the direction of the robot (see the yellow line of the robot showed in Figure 14).

In Figure 15 we have mixed objects of different types (parallelograms and arrows) and they are recognized and stored in the 3D visual memory. Also, upon detection of several arrows in the robot's environment, it will only consider the nearest one as navigation landmark and will follow its direction.

D. Robot occlusions

This experiment shows how the system behaves in case of temporary occlusions. They happen very often in real environments where there are dynamic objects which can obstruct the robot field of view.

Fig. 15. Recognition of parallelograms and arrows

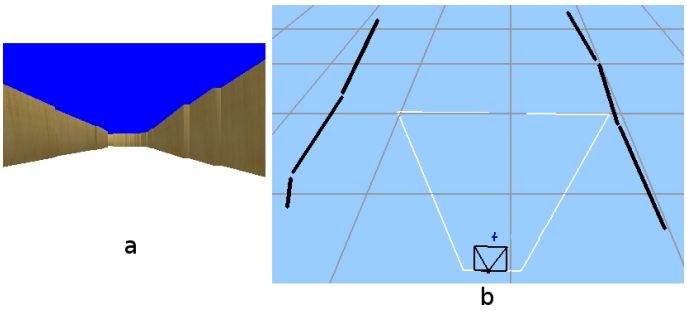

Fig. 16. (a) Obstacle free situation; (b) Short-term memory after a while

The initial situation is showed in figure 16-a. After a few seconds, our robot has recovered environment information thanks to the short-term memory and the visual attention system. This information is showed in 16-b, and it is broader than current camera field of view.

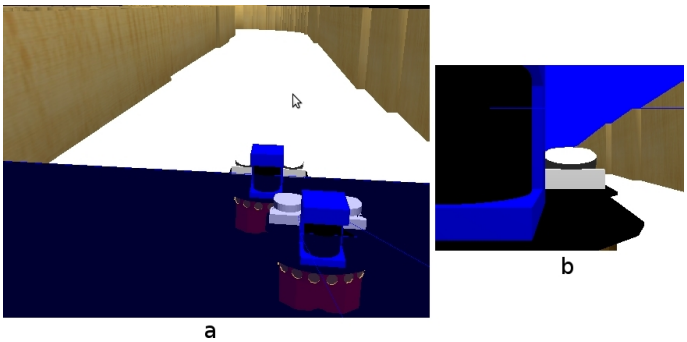

Fig. 17. (a) Situation; (b) Field of view

After a while another robot appears (as showed in Figure 17-a) occluding the field of view of our robot (as showed in Figure 17-b), so it is unable to see anything. This situation continues for some time while the second robot moves away from our robot (18-a,b).

This hindrance is solved by our system because of the

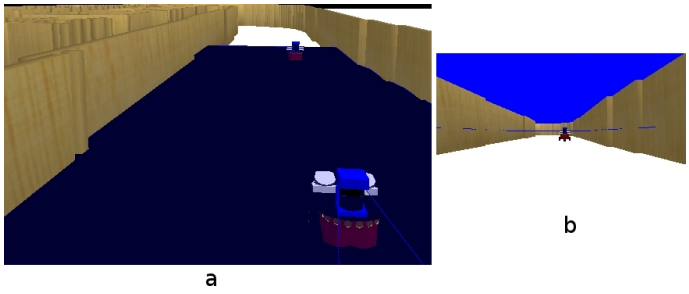

Fig. 18. (a) Situation; (b) Field of view

persistence of the short-term visual memory. As we discussed in section IV-D, the memory is refreshed over time. If it is inconsistent, that is, if what the robot sees does not match with the information stored in memory, we give a confidence interval until this situation is solved and new observations let's confirm or discard the objects in visual memory.

VI. Conclusions

In this paper we have presented an overt visual attention system whose purpose is to find objects in the scene surrounding the robot and to track them. We have developed a mechanism with two concurrent dynamics for gaze control: life (or object quality) and salience. They are defined for objects in 3D, not just the current image. The salience of objects increases in time and is set to zero every time the camera looks at it. The element with the highest salience is the next to be visited and the system controls the pan-tilt unit to look at it with the camera. This double dynamics offers a time sharing in which the robot sequentially looks at all the objects. It also accepts search 3D positions to explore for new relevant objects.

The life of objects decreases in time and grows every time the object appears in the image. Those objects with life above certain threshold are a coherent representation of the items in the scene. Objects with life below another threshold are forgotten and discarded, preventing the robot to pay attention to objects that are no longer there. The system has some patience before forgetting an item, this way the system is robust to some false negatives due to occlusions.

Since the scene is greater than the field of view of the robot camera, we implemented a 3D visual short-term memory. This memory has facilitated the internal representation of information around the robot, since objects may be placed in positions that the robot can not see at any given time but the robot knows they are there and can take them into account for better movement decisions.

Several experiments have been carried out with the visual memory, both in a real robot and in a simulated one. The robot navigates through its environment using this attentive visual memory. They show that the attention behaviors generated are quite similar to a human visual attention system.

With regard to future lines, we are working in extending the visual memory to properly represent dynamic objects. Currently the memory manages slow object movements as far as it observes the new object position close to the old one and can do the matching. Faster movements are not properly

managed, as they cause the creation of *new* objects in the new position and a ghost object remains in its old position before disappearing.

We are also working on playing with different salience dynamics for different object types. For example, let it grow faster in some objects recognized as obstacles, navigation beacons or objects very interesting for the task at hand. This would let robot to achieve a safe navigation.

ACKNOWLEDGEMENTS

This work has been supported by the project S2009/DPI-1559, RoboCity2030-II, from the Comunidad de Madrid and by the project 10/02567 from the Spanish Ministry of Science and Innovation.

REFERENCES

[Arbel y Ferrie, 2001] T. Arbel y F. Ferrie. Entropy-based gaze planning. *Image and Vision Computing, vol. 19, no. 11, pp. 779-786*, 2001.

[Arkin, 1998] C. Arkin. *Behavior-based robotics*. MIT Press, 1998.

[Badal *et al.*, 1994] S. Badal, S. Ravela, B. Draper, y A. Hanson. A practical obstacle detection and avoidance system. *Proc. of 2nd IEEE Workshop on Applications of Computer Vision, pages 97104*, 1994.

[Ballard, 1991] D. H. Ballard. Animate vision. *Artificial Intelligence 48, pp. 57-86*, 1991.

[Cañas *et al.*, 2008] J.M. Cañas, M. Martínez de la Casa, y T. González. Overt visual attention inside jde control architecture. *International Journal of Intelligent Computing in Medical Sciences and Image Procesing. Volume 2, Number 2, pp 93-100, ISSN: 1931-308X. TSI Press, USA*, 2008.

[Gartshore *et al.*, 2002] R. Gartshore, A. Aguado, y C. Galambos. Incremental map buildig using an occupancy grid for an autonomous monocular robot. *Proc. of Seventh International Conference on Control, Automation, Robotics and Vision ICARCV, pages 613618*, 2002.

[Gerardo Carrera y Davison, 2011] Adrien Angeli Gerardo Carrera y Andrew J. Davison. Lightweight slam and navigation with a multi-camera rig. In *Proceedings of the 5th European Conference on Mobile Robots ECMR 2011*, pages 77 – 82, September 2011.

[Goldberg *et al.*, 2002] S.B. Goldberg, M.W. Maimone, y L. Matthies. Stereo vision and rover navigation software for planetary exploration. *Proc. of IEEE Aerospace conference Proceedings, pages 52025,52036*, 2002.

[Hulse *et al.*, 2009] M. Hulse, S. McBride, y M. Lee. Implementing inhibition of return; embodied visual memory for robotic systems. *Proc. of the 9th Int. Conf. on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems. Lund University Cognitive Studies, 146, pp. 189 - 190*, 2009.

[Itti y Koch, 2001] L. Itti y C. Koch. Computational modelling of visual attention. *Nature Reviews Neuroscience 2, pp. 194-203*, 2001.

[Itti y Koch., 2005] L. Itti y C. Koch. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005.

[Mariottini y Roumeliotis, 2011] G.L. Mariottini y S.I. Roumeliotis. Active vision-based robot localization and navigation in a visual memory. *Proc. of ICRA*, 2011.

[Marocco y Floreano, 2002] D. Marocco y D. Floreano. Active vision and feature selection in evolutionary behavioral systems. *Proc. of Int. Conf. on Simulation of Adaptive Behavior (SAB-7), pp. 247-255*, 2002.

[Nehmzow, 1993] U. Nehmzow. Animal and robot navigation. *The Biology and Technology of Intelligent Autonomous Agents*, 1993.

[Newcombe y Davison, 2010] Richard A. Newcombe y Andrew J. Davison. Live dense reconstruction with a single moving camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010*, pages 1498 – 1505, San Francisco, California (USA), June 2010.

[Ramachandran, 1990] V. S. Ramachandran. Visual perception in people and machines. *A. Blake, editor, A.I. and the Eye, chapter 3. Wiley and Sons*, 1990.

[Remazeilles *et al.*, 2006] A. Remazeilles, F. Chaumette, y P. Gros. 3d navigation based on a visual memory. *Proc. of ICRA*, 2006.

[Srinivasan *et al.*, 2006] V. Srinivasan, S. Thurrowgood, y D. Soccol. An optical system for guidance of terrain following in uavs. *Proc. of the IEEE International Conference on Video and Signal Based Surveillance (AVSS), pages 5156*, 2006.

[Tsotsos *et al.*, 1995] J. K. Tsotsos, S. Culhane, W. Wai, Y. Lai, y N. Davis. Modeling visual attention via selective tuning. *Artifical Intelligence 78, pp. 507-545*, 1995.

[Vega y Cañas, 2009] J. Vega y J.M. Cañas. Sistema de atencin visual para la interaccin persona-robot. *Workshop on Interaccin persona-robot, Robocity 2030, pp. 91-110. ISBN: 978-84-692-5987-0*, 2009.

[Vega y Cañas, 2010] J. Vega y J.M. Cañas. Memoria visual atentiva basada en conceptos para un robot mvil. *Robocity 2030, pp 107-128, Madrid, September 15th 2010. ISBN: 84-693-6777-3*, 2010.

[Zaharescu *et al.*, 2005] A. Zaharescu, A. L. Rothenstein, y J. K. Tsotsos. Towards a biologically plausible active visual search model. *Proc. of Int. Workshop on Attention and Performance in Computational Vision WAPCV-2004, pp. 133-147*, 2005.

Multi-agent system for fast deployment of a guide robot in unknown environments

A. Canedo-Rodriguez, V. Alvarez-Santos, C.V. Regueiro, X. M. Pardo and R. Iglesias

***Abstract*—Nowadays, deploying service robots and adapting their services to a new environment is a task which might require several days. This is an important problem of robotics in general, but specially when the goal is to bring robots to our everyday life. In this paper we present a multi-agent intelligent space, which consists on intelligent cameras and autonomous guide robots. The deployment of the system does not require expertise and can be done in a short period of time. The cameras detect situations requiring the robots' guiding services, inform the robots accordingly, and support the robots navigation towards the goal areas, without the need of a map of the environment. An example of these situations requiring the robot guide service could be a group of persons entering a museum. In this sense, we also present an adaptive person follower behaviour intended to be the basis of a route learning process, necessary to offer the guide service.**

***Index Terms*—Guide robot, multi-camera networks, intelligent space, person following, feature weighting.**

I. Introduction

ROBOTS are expected to become part of our everyday life, either as assistants, house appliances, collaborating in the care of elderly people, etc. These service robots need to be able, on the one hand, to work on complex and dynamic environments and, on the other hand, to offer advanced capabilities useful to people. Despite the increasing demand for personal robots able to educate, assist, or entertain at home, or for professional service robots able to sort out tasks that are dangerous, dull, dirty, or dumb, there is still an important barrier between the latest developments on research robots and the commercial robotic applications available. The deployment of robots and the adaptation of their services to different environments usually require the tuning of their software and hardware to the particular conditions of each environment. This task is not trivial and might require several days in most cases, which makes the process of bringing the robots to everyday life extremely inefficient, specifically when the robots are targeted for short term operation, as is typical in sporadic events.

Research groups working on robotics receive numerous requests to take the robots to social events (museums, forums, etc.). However, conducting these demonstrations in different places takes a big effort, unless the robot runs the same programs under boundary conditions known a priori. A fast and easy deployment of robots in new areas is necessary to get robots operating outside research centres and beyond the continuous supervision of roboticists. In this sense, we are currently involved in the development of general purpose guide robots as part of the research project "Intelligent and distributed control scenario for the fast and easy deployment of robots in diverse environments". Through this project we try to develop robots which are able to participate in different social events (e.g. schools, museums, forums, conferences, etc.), providing useful information to visitors. Two of our goals within this project are:

- *Development of deployment strategies which reduce the amount of time required for putting our robots in operation in different environments and conditions, and also the expertise required to carry out this deployment.* Robots must be capable of being installed and put into operation within very short periods of time. Robots should be able to work in these environments without prior knowledge about them (e.g. metric maps), or without requiring significant changes on them. Finally, the deployment of the robots should not require expert knowledge (i.e. it must be as automatic as possible), prioritizing online adaptation and learning over hard-coded and pre-tuned robot behaviours.
- *Improve the quality of the guide robot and the people's opinion on them, providing them with sophisticated capabilities.* Our guide robots should be able to offer the possibility of showing different routes to those people attending an event, or to help them to reach a specific location in the environment. Nevertheless, this requires certain knowledge of these routes or places of general interest. Therefore, our robots should be able to learn routes and points of interest along these routes, by simply following a staff member working in the same building where the robot is. On the other hand, our robots should not only provide these services on people's request, but they also should be able to identify situations in which their services would be useful. Finally, our system must be reliable, ensuring the continuity of correct service as much as possible (i.e. our robots should be able to recover and learn from failures when they occur).

Most of the research done so far has focused on self-contained, stand alone robots that act autonomously, doing all the sensing, deliberation, and action selection on board. Nevertheless, in these systems, the robot sensing is restricted to the capabilities of its on-board sensors, which limits the

A. Canedo-Rodríguez, V. Álvarez-Santos, X. M. Pardo and R. Iglesias are with the Centro de Investigación en Tecnología de la Información de la Universidade de Santiago de Compostela (CITIUS). Campus Vida. Universidade de Santiago de Compostela.
E-mail: adrian.canedo@usc.es

C. V. Regueiro is with the Department of Electronics and Systems, of the Universidade de A Coruña.

ability of the robots to respond to events and to learn from its perceptions and actions. In this sense, technologies from the fields of ubiquitous and pervasive computing, sensor networks, and ambient intelligence could be integrated with robotics, providing a new way to build intelligent service robots that opposes the idea of stand-alone robotic platforms, namely ubiquitous robotics [1]. Intelligent sensor networks, which can be spread out over wider areas, will provide an invaluable source of information from beyond robots' immediate surroundings, allowing the robot to respond to a wide range of events, and making it look like it has initiative. Moreover, intelligent sensors could also enhance robots' local perceptions, supporting the movement of the robot in the environment. For example, a camera might detect a group of people entering a museum, and alert the robot about its presence, so that the robot might approach this group of people, and offer them its services. Finally, these intelligent sensor networks might be used to build high level symbolic representations of their location on the environment, removing or relaxing the need of detailed maps.

In this paper, we present the first stage of the ongoing research aimed at building an intelligent deployment strategy. The deployment of robots consists on creating a multi-agent distributed network of intelligent cameras and autonomous robots. The cameras are spread out on the environment, so that they can detect events (such as groups of people) which might require robots' services, they can inform the robots about these events, and they can also support the movement of the robots in the environment. The robots, on the other hand, navigate safely towards those areas of interest detected from the cameras, to offer the guidance along different routes. To this extent, in this article we also present an adaptive person following behaviour, able to work on crowded environments and/or under challenging illumination conditions

Section II provides an overview of the previous works concerning service robots, focused on their capability to be deployed fast and easily. Section III provides a general overview of our system. Section IV describes the tasks performed by the camera network. Section V describes the basis of the route learning process. Finally, experimental results and conclusions will follow.

II. Previous work

Over the last two decades, there have been remarkable examples of social robots, mainly of informational and guiding type, able to work on social events, like Rhino (1995) [2], Minerva (1999) [3], Tourbot and Webfair (2005) [4], and Urbano (2008) [5]. Both their quality and their deployment times have improved importantly: from 180 days of installation required by Rhino [4], to less than one hour required by Urbano for a basic installation [5]. On the contrary, major ubiquitous robotics projects have improved the quality and applicability of their systems, but the reduction of their deployment time has not received much concern.

Lee et al. (Hashimoto Labs.) [6], proposed the concept of Intelligent Spaces, as rooms which perceive and understand what is happening in them, and which perform tasks for humans. They proposed to distribute sensors (typically cameras) with processing capabilities and used them for supporting robots' navigation [7], [8]. Similarly, Intelligent Spaces have been applied successfully on the tasks of robot localization and tracking from single [9] and multiple cameras [10], and also on the task of robot navigation [11], [12], [13]. Regretfully, these proposals do not deal with the problem of easy and fast deployment in social environments, and they are not likely to be appropriate for it: all of them have been designed and validated to work in small places (less than 50 square meters), typically requiring a great number of highly coupled cameras, relying on a centralized processing and coordination, and wired communications.

The Japan NRS project, and the URUS project are a step closer to our philosophy. The NRS project focuses on user-friendly interaction between humans and networked devices (informative robots, distributed sensors, etc.), and they demonstrated the use of their systems in large real field settings, such as science museums [14], and shopping malls [15], during long term exhibitions. In the same line, the URUS project [16] proposes to deploy a network of robots, sensors and other networked devices in wide urban areas (experimental setup of 10000 m^2), to interact with the people and to perform different tasks (information tasks, transportation, surveillance, etc.). Regretfully, none of these projects consider the efficiency of the "deployment" phase, probably because their systems are intended for long term operation in the same places, where the costs of the deployment are not critical.

To the best of our knowledge, the projects above are the most representative in the context of our work, and none of them target the problem of the efficient robot deployment in unknown environments. Thus, a solution to get robots out of the laboratories to work in different environments within reasonable time and cost is still to be proposed.

III. System description

Our goal is to design and develop guide robots able to offer information and routes to the visitors of different events, in which they will work. We want these robots to be flexible to work in different environments, requiring the minimum adaptation time, effort and expertise in order to get them working, avoiding any software or hardware tuning between environments. Our system would detect groups of visitors, and the robot would navigate safely within the environment towards them, to offer them guiding services along pre-learnt routes. Also, these guide robots should not require prior knowledge of the environment (e.g. metric maps), since they should adapt as fast as possible to it. In the absence of this knowledge, the users of our system will not use a map to teach the robots the routes of interest, but they will make the robots to follow them along the real routes. Therefore, our robots will also need to be able to follow a human teacher through the environment. Usually this environment is also crowded with other people, but this should not confuse or disturb the robot.

With this system in mind, we have identified two different and important phases: deployment phase and use phase. The first one is the configuration phase, consisting on deploying

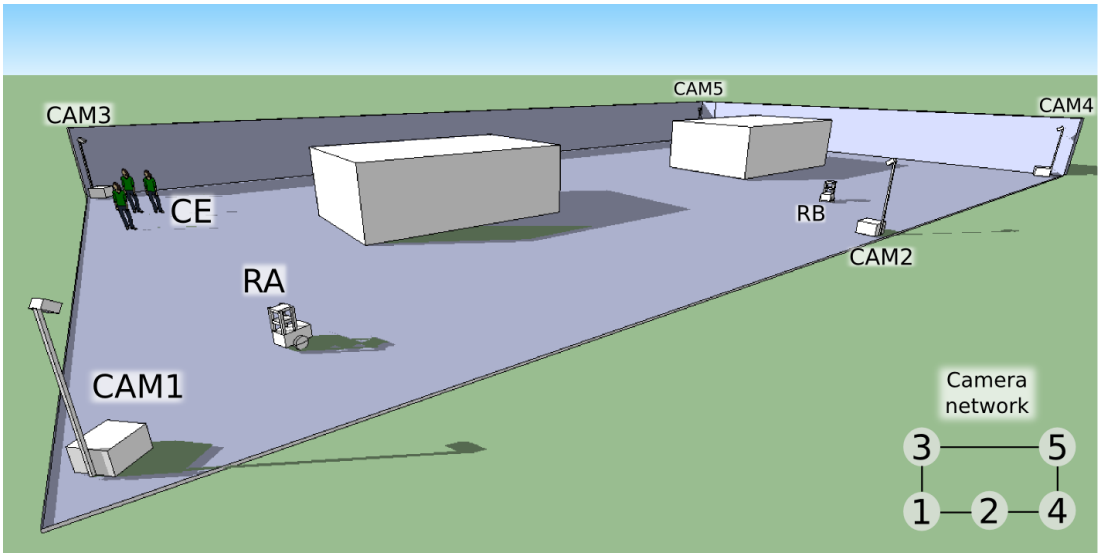

Fig. 1. Example of the deployment of the multi-agent system: camera-agent 3 (CAM3) is detecting a Call Event, while robot-agent A (RA) is being sighted by camera-agent 1 (CAM1) and robot-agent B (RB) by camera 2 (CAM2).

all that it is necessary for the system to work, and on teaching the routes to the robot. We require this configuration phase to be as short and fast as possible, and it also must be easy to accomplish it for anybody (even non expert users). In the second phase, the system should detect groups of visitors that might require the robots' guiding services (Figure 1). Robots should navigate towards this groups of people to offer them the possibility of following routes that have been previously learnt. In short, in the first phase we would deploy and configure the system, while in the second phase the system would actually give the services for which we designed it.

To accomplish these tasks, we propose a multi-agent system such as the one illustrated in Figure 1, which consists on two main elements: a) an intelligent control system formed by camera-agents spread out on the environment (CAM1 to CAM5 in the Figure), and b) autonomous guide robots navigating on it (RA and RB in the Figure). Basically, the cameras detect Call Events (CE in the Figure) within their Fields of View (FOVs from now on): typically, groups of visitors. Then, the cameras support the robots to navigate towards the Call Events, so the need of a map is avoided. Once the robots arrive at the destination, they offer information or guiding along routes of interest to these people. The agents coordinate in a fully distributed fashion, based on local interactions. This allows us to introduce more agents in the system without major changes, favouring scalability and robustness. In Section III-A and III-B we describe both agents in more detail.

A. Camera-agents

Each camera-agent consists on an aluminium structure like the one represented in Figure 2, which is easy to transport, to deploy and to pick up. As we can see, the aluminium structure is made up of a box, which contains a processing unit, a WIFI Access Point and power supplies, and a mast which holds one or more video cameras (one for each camera-agent).

On the other hand, the software consists on four concurrent

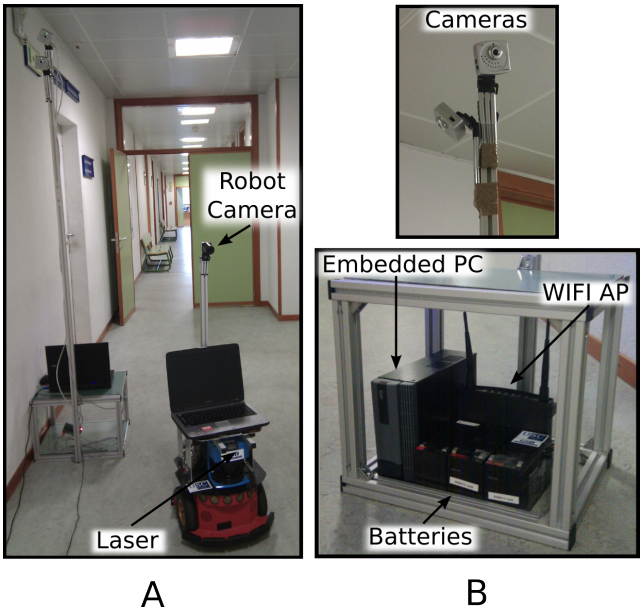

Fig. 2. A) Camera-agent (left) and robot-agent (right). B) Hardware elements that make up each camera-agent.

modules, responsible for carrying out the tasks corresponding to these agents (Section IV):

- *Vision Module*: responsible for the detection and tracking of robots and call events.
- *Network Module*: responsible for wireless communications.
- *Call Module*: responsible of notifying robots about call events present in its FOV.
- *Neighbourhood Module*: responsible of detecting neighbourhood relationships with other cameras by detecting common parts among their FOVs. As we will describe later, by establishing local neighbourhood relationships and by interacting locally with their neighbours, the

cameras will be able to form routes among them, and to support the navigation of the robots.

B. Robot-agents

We work with wheeled robots like the Pioneer 3DX, equipped with a laser scanner and a video camera (Figure 2). The software of the robot is divided in two concurrent modules: a *Network Module* for communications and a *Control Module* for navigation. The Control Module is based on the classic but robust Potential Fields Method: the robot moves towards a goal position which exerts an attractive force on it, while the obstacles detected by the laser scanner exert repulsive forces, so that the robot avoids colliding with them. As we will see later, this goal and therefore the attractive force can be either provided by cameras when supporting robots' navigation, or by the users of the system when the robot is learning a tour route by following any of them.

IV. Deployment of an intelligent network space

As we have stated before, camera-agents are responsible for detecting events requiring robots' presence, and for supporting robots' through the commitment of their duties. In order to fulfil these objectives, the cameras perform the following low-level tasks: dynamic neighbourhood detection, distributed route planning, and support to robot navigation. First, upon distribution in the environment, each camera discovers who are its camera neighbours (*dynamic neighbourhood detection*). When a robot is required to go from one camera to another, the cameras execute a distributed process based only on local interactions with their neighbours, to discover all the possible inter-connection routes (*distributed route planning*). Finally, each camera supports the robot's movement towards the next neighbour camera on the route (*support to robot navigation*). We would like to remark that neither a global camera topology representation, nor the calculated routes are stored anywhere: each camera keeps only information about its neighbour cameras. This will be explained in the following sections.

A. Dynamic Neighbourhood Detection

The correct functioning of the solution we propose requires each camera being aware of which of the other cameras in the intelligent network are its neighbours, and which parts of their FOVs overlap (this overlapping FOVs will be called Neighbourhood Regions from now on). We would like to remark that our system does not require the cameras to know the relative poses (position and orientation) of their neighbour cameras, thus we avoid the introduction of complex matching techniques [17]. Instead, in our system, each camera determines which cameras are their neighbours by detecting simultaneous high-level events, like the detection of a robot from several cameras. Whenever a camera detects a robot, it stores its position and broadcasts it to all the other cameras. Periodically, each camera checks whether the detection of the robot is taking place simultaneously with the detection of the same robot but from any other camera, in which case a neighbourhood relationship will be established.

With this automatic process, the system avoids the need of a metric map of the environment, or even the need of pre-installed knowledge about the distribution of the cameras. Moreover, since the neighbourhood relationships can change dynamically (e.g. if some camera is moved or if it stops working), the cameras are continuously updating this neighbourhood information to adapt to eventual changes, without requiring users to re-configure them.

B. Distributed Route Planning

A route is an ordered list of cameras through which the robot can navigate. Basically, the robot will go from the FOV of one of the cameras to the FOV of the next camera on the route, without needing metric maps of the environment. These routes will be generated on demand as a result of local interactions among the cameras, without the intervention of any central agent.

We will explain the route planning process through an example, depicted in Figure 3. In this Figure, we represent the system as a graph, with cameras as nodes and their neighbourhood relationships as arcs. However, note that this graph is just for illustration purposes, none of the entities in our system handle this global information. In this example, we assume that robot RA is available (willing to accept any call), while robot RB is not. We also assume that camera 1 is detecting a call event (CE), camera 2 is seeing robot RB, and camera 5 is seeing robot RA.

If a camera, like camera 1 in Figure 3-A, detects a call event requiring the presence of a robot, it broadcasts a call to all the robots. If a robot is willing to attend the call, it broadcasts an acceptance to all the cameras, like robot RA does. Then, those cameras which receive this acceptance and which see this robot will forward the message to its camera neighbours, starting a back propagation process to create a route, as camera 5 does in Figure 3-B: through this process, there will be a message being passed from camera to camera, until it reaches the camera which sees the call event. To do so, each camera which receives this message includes its identity in it, and forwards it to its neighbours (except to those through which the message has already passed). In the Figure, it is clear that camera 5 includes its identity in the message and forwards it to its neighbours, camera 3 and 4, which do the same, and so on. Finally, camera 1 receives all the acceptances, so it knows that RA is willing to accept the call, and that it could follow two possible routes: 5-3-1 and 5-4-2-1. After this, this camera would select the route which involves less cameras (5-3-1), and inform robot RA accordingly. It is clear, after this description, that the route planning does not emerge from a globally coordinated process, but from multiple local interactions among agents just handling local information.

C. Support to Robot Navigation

When a robot is following a route, each camera on the route helps it to move towards the next Neighbourhood Region, so the robot advances in the route. For doing so, each camera keeps information both about its neighbours and about its Neighbourhood Regions (overlapping FOV areas). Also, when

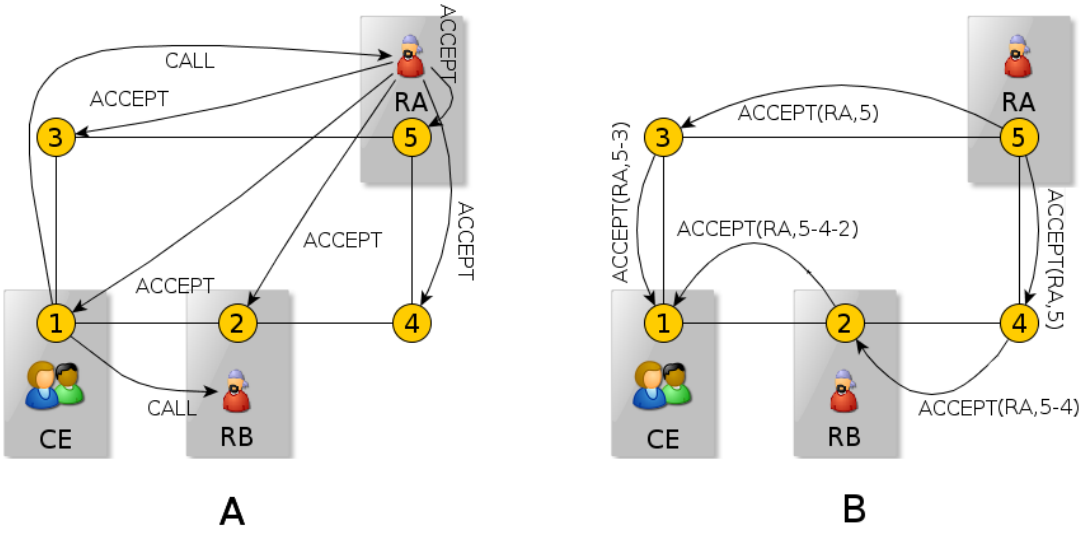

Fig. 3. Distributed route planning procedure. Cameras (1, 2, 3, 4, 5) established their neighbourhood relationships forming a network altogether. A) Call event detection and call for robots. B) Back propagation process for route formation. See the text for a detailed explanation.

a camera detects a robot, it tracks its position and movement direction. This is all the information required by the camera to support robots' navigation.

First of all, if a camera sees a robot, it informs it, after which the robot answers with the route which it is following (in case that it is following any route). Then, the camera informs the robot about the direction that it should follow to get to the Neighbourhood Region shared with the next camera on the route, taking into account the robot's current movement direction. Since the Potential Fields Controller of the robot considers this direction as an attractive force, the robot moves from the FOV of each camera towards the FOV of the next camera on the route, until it reaches the call event.

V. Route Learning

As we have previously said, one of our goals is to build a guide robot which is able to work in different environments. This robot will offer a route service which can be divided in two parts: route learning, and reproduction of previously learnt routes. So far, our work has concentrated on the first part: route learning.

Teaching a new route to the robot should require neither expertise nor time, since it is a process that must be done every time the robot is brought to a new place. On the other hand, we do not want the users to teach the routes using a map of the environment, since maps are not used by our system. For these two reasons, the robot will learn routes while following a human (the target from now on) who plays the role of the teacher. This process must be safe, avoiding collisions of the robot with the environment, and it must also be robust and adaptive, avoiding mistaking the target for the rest of the people present in the same scene (we call *distractor* to each person moving around the robot and that is not the target). Our goal is to create a robust system for person following and target recognition. Our system must be flexible enough to handle important variations in illumination, scene clutter, multiple moving objects, and other arbitrary changes in the observed scene. On the other hand the person being recognized and followed will not need to wear special clothes or gadgets, thus achieving a more natural human-robot interaction. The presence of distractors moving around the robot might occlude the target, or these distractors might even look similar to the target due to changing light conditions. This is critical, especially when the robot is being taught different routes in crowded environments, where confusing the target might cause the robot to learn wrong routes, and to start over the learning of the route again.

In the next sections we will give a brief description of the person following behaviour that we have designed.

A. Person Following Overview

An overview of the system we have developed to solve the person following and target recognition problems can be seen in Fig. 4. On one hand, the inputs to the system are the images provided by a camera which is placed on top of the robot, and data from a laser scanner also located on the robot (Fig. 2 A). On the other hand the system provides the position of the target to a robot controller, which will use this information to determine the motor commands that the robot must carry out so that it follows the target and avoids colliding with the environment.

As we can see in Fig. 4 the system we have developed includes two modules which work together to obtain the target's position:

- *The camera module*: The task of this module is to avoid mistaking the target for the distractors. To do this, this module will recognize the target from its torso; in particular it will build, store, and keep updated information about the visual features of the target's torso. An outline of the tasks that this module carries out for every frame acquired from the camera, is: first, the module detects people in the image by using the algorithm developed by Dalal [18].

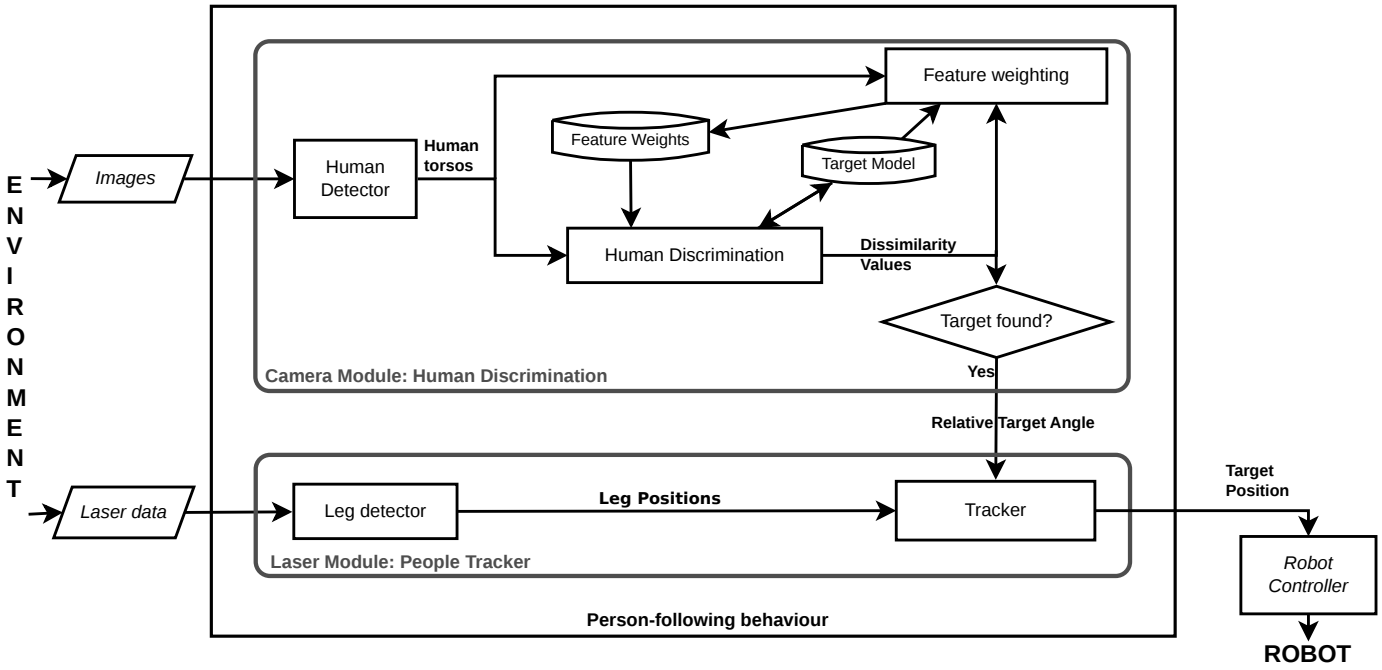

Fig. 4. Schematic representation of the person following behaviour. The camera module identifies the target and sends information about its location to the laser module, which is in charge of tracking its position. In the case that the target is not identified by the camera the laser module can still work.

Using this algorithm it is possible to detect areas in the image in which there seems to be a person. Next, for each one of these areas, this module extracts the visual features of the region that contains the person's torso. These features are then compared with the features of the stored target's torso to determine whether the person in the image is the target or not. Finally, if the person in the image is considered to be the target, the features of the target's torso might be updated using the current detection. On the other hand, this module also determines the angle at which the target is located with respect to the forward direction of the robot, and sends this information to the laser module. When the target is not found no information will be sent to the laser module.

- *The laser module*: The task of this module is to track the robot's target in the course of time. First, this module uses the information provided by a laser scanner located on the robot to carry out a leg detection process. Then, it computes the angles at which the legs are detected with respect to the forward direction of the robot, and sends these relative angles to the person tracker block (tracker in Fig. 4). So far in this description, both sensor modules seem to work separately, nevertheless if the camera provides information about the target, the tracker block will merge it with the information about the leg positions.

According to the previous description, it is straightforward to realize that we use two sensor modalities: a laser scanner, and a camera. The example shown in Fig. 5 summarizes the merging process carried out by the person tracker when it receives information from the camera module. In this figure we can observe three pairs of legs located at the positions L_1, L_2 and L_3. We can assume that these pairs of legs are detected in the laser module, which also computes the angles at which they are detected with respect to the forward direction of the robot: θ_{pair1}, θ_{pair2} and θ_{pair3} in Fig. 5. The point P is the last position where the target was detected, and θ_{target}, is the angle at which the camera module found the target, with respect to the forward direction of the robot.

Using this information, the *tracker block* will decide which one of the pairs of legs (pair 1,2 or 3) corresponds to the target. The decision is taken by assigning a probability p to each pair of legs:

$$p = \frac{c(\Delta\theta) + l(x)}{2} \tag{1}$$

where:

1) $c(\Delta\theta)$ is the probability of each pair of legs being the target, according to the camera module:

$$c(\Delta\theta) = \begin{cases} 1, & 0 \leq |\Delta\theta| \leq 4 \\ 1.5 - \frac{1}{8}|\Delta\theta|, & 4 < |\Delta\theta| \leq 12 \\ 0, & otherwise \end{cases} \tag{2}$$

where $|\Delta\theta|$ is the absolute value of the difference between the angle where the camera finds the target (θ_{target}), and the angle where the laser locates the group of legs (θ_{pairN}).

2) and $l(x)$ is the probability of each pair of legs being the target, according to the laser:

$$l(x) = \begin{cases} 1 - \frac{2}{3}x, & 0 \leq x \leq 1.5 \\ 0, & otherwise \end{cases} \tag{3}$$

where x is the distance in meters from the position of each group of legs L_N to the last position where the target has been detected, P.

The equation for the $c(\Delta\theta)$ probability was obtained heuristically after checking the behaviour of the robot when it moves

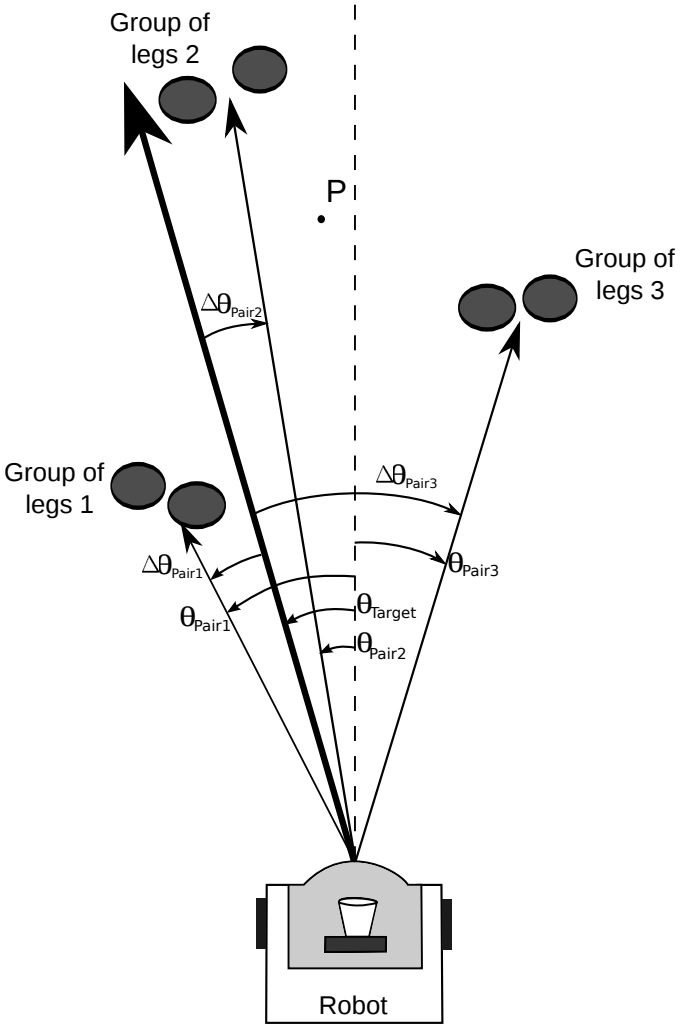

Fig. 5. Example situation where the sensor fusion process takes place.

on an uneven floor. In general we have observed that the vibrations due to this kind of floor might make the camera oscillate in a range of sixteen degrees (eight degrees to each side). This is the reason why $c(\Delta\theta)$ shows a tolerance in that range, i.e., all the pairs of legs which are located within 8 degrees with respect to the direction of the target detected by the camera, should be considered as belonging to the target with a probability higher than 0.5 (Eq. 2).

To build the equation Eq. 3 for $l(x)$ we assume that the walking speed of the target is not higher than 1.7m/s (the average human walking speed is about 1.33m/s). In this case the target should not move more than 0.2m every 100ms (which is the elapsing time between two consecutive laser scans). We also decided to increase the margin of tolerance, due to the high probability of getting noisy measurements; the central position of a pair of legs can be wrong if only one is properly detected, etc. This is the reason why we assume that the position of the target should not change more than 0.75m in two consecutive acquisitions (region around P with a probability of finding the target higher than 0.5, Eq. 3).

We have focused our work on the camera module. In particular we want to find a strategy of combination of visual cues that allow the discrimination of people in different environments, that is robust to changing illumination conditions, and that is able operate in real-time. The achievement of this would allow robust human-robot interactions, and good person following behaviours, in which misclassifications of the target are less frequent despite of the existence of temporary occlusions, or periods of time in which the target is out of sight. Another good reason for focusing our work in the camera module is due to the fact that, nowadays, it is much more expensive to include in the robot a laser scanner than a conventional camera. This allows cheap robots to perform human recognition.

In the next two subsections we describe in detail the two most important parts of the camera module, the human discrimination algorithm and the online feature weighting process.

B. Discrimination algorithm

The discrimination algorithm pursuits the differentiation of the demonstrator (target being followed by the robot) from the rest of the people moving in the same area (distractors). This algorithm runs inside the camera module (*human discrimination* block in Fig4). Basically, the discrimination algorithm will use the information of the torsos extracted from the people detected in the image, to identify whether the target is present or not. To understand the process we must realize the fact that there are two clearly different states: *initialization* stage, and *running* stage.

During an *initial stage*, when the robot is about to follow the instructor, it is assumed that the target is located in front of the robot. During this stage the system builds a model of the target by extracting the distribution of the following features from his torso: hue, lightness, saturation (colour features from the HLS colour space), local binary patterns [19] and the edges detected with the Canny method [20] (texture features). This *initial stage* is very fast and goes unnoticed for the instructor. After few instants the robot will start moving and following the target, differentiating it from the distractors (running stage).

During the *running stage*, the discrimination algorithm computes the dissimilarity between the target's model and the torsos detected in the image. First, in order to obtain the dissimilarity value, we need to compute the inverse Bhattacharyya distance between the histogram of each torso's feature, h_f, and the histogram of the same feature in the target's model h_{tf}:

$$d_f = \sqrt{1 - \sum_{i=0}^{b-1} \sqrt{h_f(i) h_{tf}(i)}} \tag{4}$$

where b is the number of bins in the histograms. Using these distances we can define the dissimilarity between each torso and the target's model as the average value of the d_f for the five features being used:

$$dissimilarity = \frac{1}{n} \sum_{f=1}^{n} d_f \quad \in [0,1] \tag{5}$$

where n is the number of features, five in our case.

Finally, the discrimination algorithm will decide whether there is a torso that is similar enough to be considered the target, and whether the system should update the target's torso using the current detection. Since our dissimilarity value oscillates from 0 (very similar or equal) to 1 (a completely different torso) we have set two thresholds. The first one ($thr_1 = 0.4$) is the limit to consider a torso as an instance of the target, while the second one is a more restrictive threshold ($thr_2 = 0.2$), and it is used to decide when the target's model can be updated with the torso which is being identified as the target. This dual-threshold strategy avoids the pollution of the target's model with a false positive detection.

However, the dissimilarity measure described before (Eq. 5) is not yet robust enough to cope with real world conditions, such as strong illumination reflections, shadows, occlusions, and situations where the algorithm has to discriminate among very similar torsos. Because of this, we have designed an adaptive weighting process to dynamically enhance the differences between the target and the distractors, this process is described in the following section.

C. Online Feature Weighting

This section describes the process called 'Feature weighting' in Fig. 4. This process consists on dynamically selecting the most appropriate weights for each feature to adapt to the changes in the environment, such as the illumination conditions or people's clothes, and thus enhance the discrimination of the target from the distractors. This process has been studied in the area of image retrieval with query relevance feedback. It consists on measuring the discrimination ability of each feature when differentiating between two classes. In robotics, we can define online feature weighting for human discrimination as the process of dynamically assigning high weights to those features that show a high discrimination ability between the target and the distractors. This is very useful when target and distractors show a similar distribution on some features but differ on the others. We can think of, for example, a target and several distractors wearing similar colour clothes but with different patterns. In this case it would be more useful to focus the dissimilarity on the texture while discarding the colour features.

First, to be able to enhance dissimilarity between the target and the distractors, we need to store information about the last distractors detected from the robot in a list. The distractors list is built and updated every frame according to the following rules:

1) If a torso can be classified as the target, the rest of the torsos that have been detected in the same frame will be placed on the distractor list provided that they do not overlap in the image with the torso corresponding to the target.
2) If there is no torso that can be classified as the target, those with a dissimilarity value higher than 0.5 will be put on the distractor list.
3) The list has a size limit of five torsos. When the limit is reached, the oldest torsos will be removed. This size limit of the list is set to consider only the most recently seen torsos. A larger distractor list would save torsos which will not be seen again in a short period of time, thus reducing the performance of the feature weighting process.

To assign the most suitable weights to the subset of features being used, we need a scoring function which should measure the discrimination ability of each feature at each instant. This is why we have proposed to use the Bhattacharyya distance as a score function [21]: our score is based on Eq. 4. The idea behind this score is that the best features should be the ones that minimize the distance between the last torso classified as the target and the previous target model ($d_{f,target}$, in Eq. 6) and, at the same time, it also maximizes the average distance between the distractors and the current target model ($\bar{d}_{f,distractors}$ in Eq. 6):

$$score_f = \bar{d}_{f,distractors} - d_{f,target} \tag{6}$$

Using the aforementioned function we can score the discrimination ability of each feature, and thus weight the importance of each feature in the human discrimination algorithm (Section V-B). In particular we replace Eq. 5 with a new measurement that considers the weights of the different features:

$$dissimilarity = \sum_{f=1}^{n} w_f d_f \quad \in [0,1] \tag{7}$$

Initially, the weights are the same for all features, i.e., $w_f = \frac{1}{n}$, $\forall f = 1, ..., 5$, but as the robot proceeds moving in the environment while following the target, the weights will be updated according to Eq. 8:

$$w_f = w_f + score_f \tag{8}$$

where w_f is the weight for feature f. Every time the weights change, it is important to re-normalize them, so that their sum is one.

VI. Implementation and Experimental Results

We have tested the system on the Department of Electronics & Computer Science, at the University of Santiago de Compostela, Spain. The robot used in the tests is a Pioneer P3DX with a SICK-LMS200 laser and a PointGrey Chameleon CMLN-13S2C with a FUJINON FUJIFILM Vari-focal CCTV Lens (omnidirectional). On the other hand, each camera agent used either an Unibrain Fire-i camera, or an omnidirectional camera like that of the robot. The processing units where either a DELL Latitude E550 (Intel(R) Core(TM) 2 Duo P8600 @ 2.4 GHz, 4 GB RAM) or a Toshiba Satellite A100-497 (Intel(R) Core(TM) 2 Duo T5600 @ 1.83 GHz, 4 GB RAM). Regarding the software of the controllers, it was implemented using the Player(v-3.0.2)-Stage(v.4.0.0) platform for the robot, and the OpenCV 2.2 library [22] for image processing. Finally, messages were passed over an IEEE 802.11g local wireless network via UDP.

Although the intelligent camera space and the sensor data from the robot already let us record a route, we are still working on its reproduction, thus the experiments that will be presented in this section do not include route reproduction. We

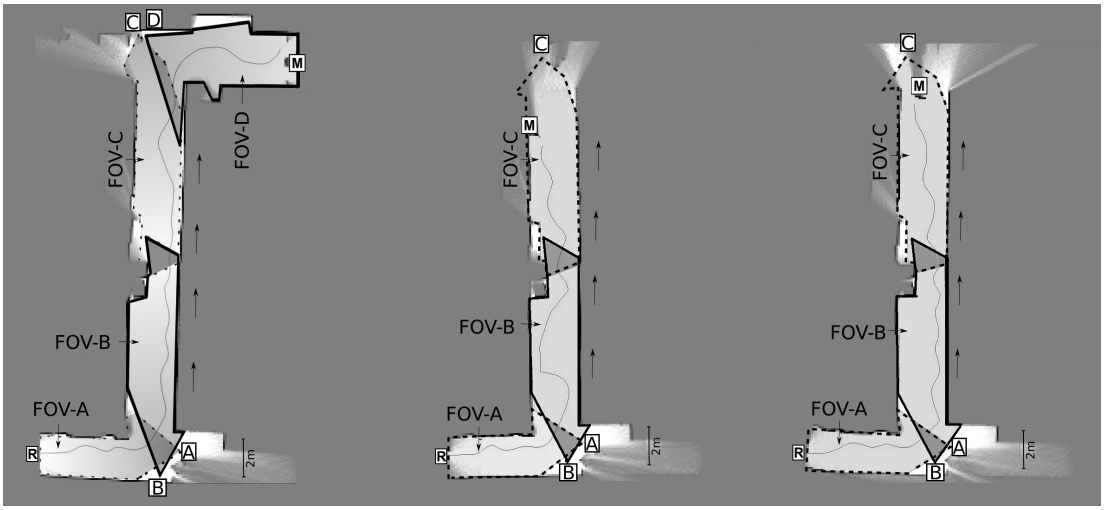

Fig. 6. Trajectories described by the robot in three tests. In the leftmost test, the robot R navigates from the FOV of camera A, passing through B, C and finally D, which triggered the call upon detection of the call event M. In the other tests, the robot navigates from the FOV of camera A, passing through B, towards C. The map and trajectory of the robot was obtained from the robot's odometry and laser readings using the PMAP SLAM library. In the figure we can also see the approximate positions of the cameras and their FOVs.

evaluated our system in two different experiments. In the first one, we tested how the cameras detect a call event, establish neighbourhood relationships among them, generate global routes, and support robot navigation to reach the call event. For simplicity, we used a colour marker in order to simulate the call event, but in the future we will include the detection of groups of people. The second experiment consisted on following a human through the department simulating a route learning process. In this experiment we focus on evaluating the performance of the human discrimination algorithm, and we also give some commentaries about the robot behaviour when following the person.

A. Experiment I - Robot navigation to attend a call event

We deployed a multi-agent network over the Department of Electronics and Computer Science at the University of Santiago of Compostela. We have performed three different tests, represented in Figure 6.

In the first test (Fig. 6 on the left), the network consisted on a robot-agent (R), four camera-agents (A, B, C, D) and a call event (M). Camera D was the one which sighted the call event (M), started the robot call process, and triggered the route formation. Once the robot, R, received the route to follow, it started navigating through the network towards M, supported by A, B, C, and D, while avoiding the obstacles detected.

In the second and in the third tests, the network consisted on three cameras (A, B, C) instead of four. As we show in Figure 6, the performances of these tests were similar to the performance of the first one, described above.

The robot's trajectories and the maps shown in Figure 6 were obtained from the robot's odometry and laser logs using the PMAP SLAM library, compatible with Player-Stage. Nevertheless, these maps have only been used for visualization purposes, but not during the functioning of the system. We repeated these tests a few times, obtaining in all of them a satisfactory performance of the robot.

In this experiment each camera had only two neighbours, nevertheless the FOV of three or more cameras might overlap and therefore the number of neighbour cameras can be higher than two. Our system would be able to cope with this: the route that is finally selected to be followed by the robot is always the one that involves the fewest number of cameras. As part of our future work, we plan to run new experiments in wide environments where the number of cameras with overlapping FOV will be higher than two (for example in big halls).

B. Experiment II - Person following

We have also tested the person-following behaviour through several routes in the same environment where we run the previous experiment. In each route a target walked at least 50 meters while the robot followed him, and at least one distractor was walking close to the target to evaluate the discrimination power of the online feature weighting algorithm. We have recorded one of these routes, and used it to evaluate the benefits of using our proposal over the classic approaches based on the comparison of features without weighting them. This sequence also let us test the system when the illumination conditions change significantly, altering the colour and texture properties of the torsos.

On the recorded sequence, the torsos are expected to be discriminated using texture and lightness features since both are mainly black but one has light grey drawings on it. Figure 7 shows that the results confirm what we expected: two of the features represent 80% of the total sum of weights during most of the video sequence. Analysing Figure 7 we can also notice that there is a small time interval at the beginning (from T_0 to T_1), in which there is no predominant feature and the weight values are similar to each other. This is due to the illumination conditions and pose of the target. Nevertheless this situation changes soon, achieving a small set of prominent features that increase the separability between target and distractors.

We have also selected frames where confusion could arise: the selected were all but those where the target was the only

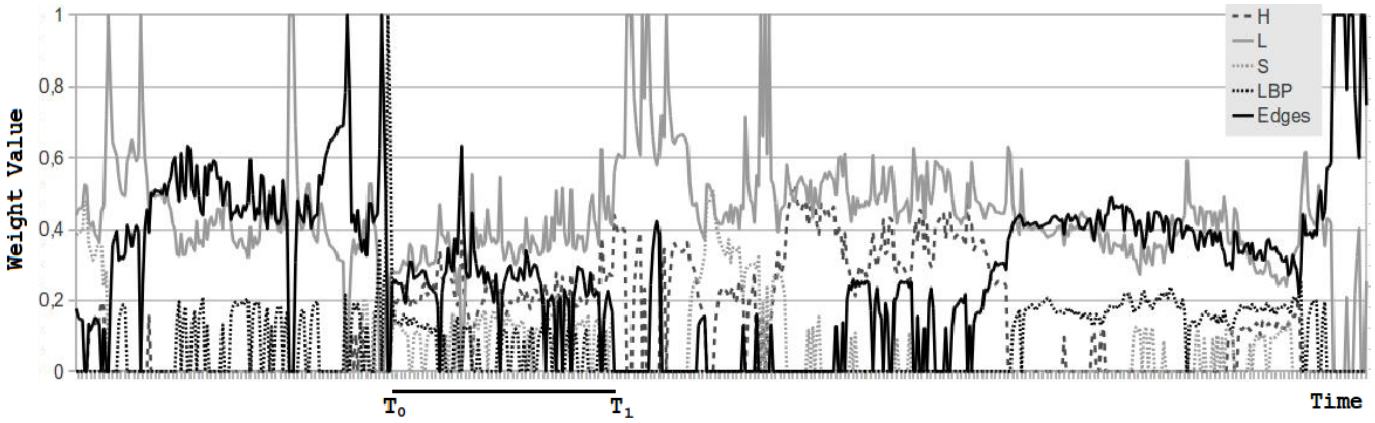

Fig. 7. Evolution of feature weights during one experiment. These weights determine the importance of each feature in the person following behaviour. According to the evolution of the weights it is noticeable that 'edges' and 'lightness' are the most important features during most of the experiment. Our algorithm is able to find the relevant features in most cases, although there are still some cases like the one between T_0 and T_1 where it is hard to find a relevant subset of features due to fast changing of the illumination conditions or the pose of the target.

TABLE I
CONFUSION MATRIX USING THE CLASSIC APPROACH

Actual\Classif.	Target	Distractor
Target	139(80.4%)	34(19.6%)
Distractor	8(3,6%)	212(96,3%)

detected human and thus confusion was not possible. With this selection we have built a confusion matrix out of the results observed in these frames. Table I shows the confusion matrix when using the classic approach and Table II shows the confusion matrix when using our proposed weighting of the feature space. We can see that the recognition ratio of the target increased from 80% to almost 100%. False positives decreased from eight to three reducing the number of times that the robot might follow a distractor as if he were the target.

TABLE II
CONFUSION MATRIX USING OUR APPROACH

Actual\Classif.	Target	Distractor
Target	142(99.3%)	1(0.6%)
Distractor	3(1.3%)	217(98.6%)

These results confirm that our system is capable of adapting to difficult conditions maximizing the dissimilarity target-distractor. We have also tested the person-following controller operating on the real robot during several 10 minute walks around the hall and different corridors of the same location as the other experiments. The robot had to follow the target when both the corridors and the premises of the building were usually crowded with students walking around. The robot's maximum speed was set to 1 m/s, thus allowing the target to walk at normal speed. The robot was able to avoid collisions with the environment thanks to a potential fields method implemented on the robot controller. The robot was able to follow the target keeping a distance that ranged from 0.4 meters to 6 meters, although the average distance between the robot and the target was 2 meters.

VII. SUMMARY, CONCLUSIONS, AND FUTURE WORK

In this paper, we have presented a system for fast and easy deployment of guide robots in unknown environments, together with a person following behaviour, which is the basis of the route learning ability usually desirable for any advanced guide robot. The work presented here is part of a bigger project, in which we aim at developing robots which are able to participate in different social events, providing useful information to visitors. We based our design in the requirements of scalability, robustness, flexibility, and adaptability. On the one hand, we achieve fast and easy deployment by not requiring prior knowledge of the environment (e.g. metric maps), nor big expertise in order to deploy it. Moreover, we have designed it to not require any software or hardware tuning, so as to be as self-contained and automatic as possible. We have also established the basis of the route learning ability of the guide robots, by presenting an adaptive person following behaviour.

Our system consists on a distributed multi-agent network formed by two kinds of agents: intelligent cameras spread out on the environment, and autonomous robots navigating within it. The camera network senses the environment, informs robots about events happening out of their immediate surroundings (which enhances their initiative), and supports them on their duties, removing or relaxing the need of a map. We did not use any centralization or hierarchy, but biologically inspired self-organization processes, based on distribution, inter-agent independence, and emergent behaviours out of local interactions, instead of global plans. This resulted on a system highly independent of environment changes, redundant, and flexible enough to cope with a wide range of spatial distributions of the cameras.

On the other hand, the person following behaviour which we have built combines a laser based tracker, with the discrimination power of a camera. The discrimination algorithm running on the camera is inspired on image retrieval systems, which are able to adapt a set of weights for each situation that the robot might encounter. These weights enhance dissimilarity between the target being followed and the other people present in the scene, avoiding getting confused with them.

The work presented here is just the beginning of a bigger project. In future stages of our research, we will include the detection of real world events requiring robot's presence, such as groups of people interested on the services offered by the robot. Also, we will automatize the detection of overlap FOVs attending at people moving within the environment. At more advanced stages, we will explore more flexible camera arrangements, including multiple

increase the complexity of the relationships between the cameras and even remove the FOVs' overlap restriction at more advances stages. Moreover, we plan to improve the robustness of the robot navigation, by learning from people trajectories. Finally, we plan to keep improving the person following behaviour by including gesture recognition to enhance the process of route learning with a more natural interaction between the human and the robot.

ACKNOWLEDGEMENT

This work was supported by the research projects TIN2009-07737, INCITE08PXIB262202PR, and the grant BES-2010-040813 FPI-MICINN.

REFERENCES

[1] J. Kim, K. Lee, Y. Kim, N. Kuppuswamy, J. Jo, "Ubiquitous robot: A new paradigm for integrated services," *IEEE International Conference on Robotics and Automation*, pp. 2853-2858, 2007.

[2] J. Buhmann, W. Burgard, A. Cremers, D. Fox, T. Hofmann, F. Schneider, J. Strikos, S. Thrun, "The mobile robot rhino," *AI Magazine*, vol. 16, no. 3, pp. 31-38, 1995.

[3] S. Thrun, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, et al., Minerva: "A second-generation museum tour-guide robot," *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, pp. 1999-2005, 1999.

[4] P. Trahanias, W. Burgard, A. Argyros, D. Hahnel, H. Baltzakis, P. Pfaff, C. Stachniss, "Tourbot and webfair: Web-operated mobile robots for tele-presence in populated exhibitions," *IEEE Robotics & Automation Magazine*, vol. 12, no. 2, pp. 77-89, 2005.

[5] D. Rodriguez-Losada, F. Matia, R. Galan, M. Hernando, J. Montero, J. Lucas, "Urbano, an interactive mobile tour-guide robot," *Advances in Service Robotics*, Ed. H. Seok. In-Teh, pp. 229-252, 2008.

[6] J. Lee, H. Hashimoto, "Intelligent space concept and contents,"*Advanced Robotics*, vol. 16, no. 3, pp. 265-280, 2002.

[7] J. Lee, H. Hashimoto, "Controlling mobile robots in distributed intelligent sensor network, "IEEE Transactions on Industrial Electronics," vol. 50, no. 3, pp. 890-902, 2010.

[8] J. Lee, K. Morioka, N. Ando, H. Hashimoto, "Cooperation of distributed intelligent sensors in intelligent environment," *IEEE/ASME Transactions on Mechatronics*, vol. 9, no. 3, pp. 535-543, 2004.

[9] D. Pizarro, M. Mazo, E. Santiso, M. Marron, D. Jimenez, S. Cobreces, C. Losada, "Localization of mobile robots using odometry and an external vision sensor," *Sensors*, vol. 10, no. 4, pp. 3655-3680, 2010.

[10] C. Losada, M. Mazo, S. Palazuelos, D. Pizarro, M. Marron, "Multi-camera sensor system for 3d segmentation and localization of multiple mobile robots," *Sensors*, vol. 10, no. 4, pp. 3261-3279, 2010.

[11] I. Fernandez, M. Mazo, J. Lazaro, D. Pizarro, E. Santiso, P. Martin, C. Losada, "Guidance of a mobile robot using an array of static cameras located in the environment," *Autonomous Robots*, vol. 23, no. 4, pp. 305-324, 2007.

[12] P. Steinhaus, M. Walther, B. Giesler, R. Dillmann, "3d global and mobile sensor data fusion for mobile platform navigation," *Proceedings ICRA'04. 2004 IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3325-3330, 2004.

[13] P. Steinhaus, M. Strand, R. Dillmann, "Autonomous robot navigation in human-centered environments based on 3d data fusion," *EURASIP Journal on Applied Signal Processing*. vol. 2007, no. 1, pp. 224-224, 2007.

[14] M. Shiomi, T. Kanda, H. Ishiguro, N. Hagita, "Interactive humanoid robots for a science museum," *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pp. 305-312.

[15] M. Shiomi, T. Kanda, D. Glas, S. Satake, H. Ishiguro, N. Hagita, "Field trial of networked social robots in a shopping mall," *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, pp. 2846-2853.

[16] A. Sanfeliu, J. Andrade-Cetto, M. Barbosa, R. Bowden, J. Capitan, A. Corominas, A. Gilbert, J. Illingworth, L. Merino, J. Mirats, et al., "Decentralized sensor fusion for ubiquitous networking robotics in urban areas," *Sensors*, vol. 10, no. 3, pp. 2274-2314, 2010.

[17] S. Funiak, C. Guestrin, M. Paskin, R. Sukthankar, "Distributed localization of networked cameras," *Proceedings of the Fifth International Conference on Information Processing in Sensor Networks*, pp. 24-42, 2006.

[18] Dalal, N., and Triggs, B.: "Histograms of oriented gradients for human detection," *CVPR 2005, IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 886-893, 2005.

[19] Ojala, T., Pietikainen, M., and Harwood, D.: "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51-59, 1996.

[20] Canny, J.: "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1631-1643, 1986.

[21] Alvarez-Santos, V., Iglesias, R., Pardo, X. M., Canedo-Rodriguez, A., Regueiro, C. V.: "Online Feature Weighting for Human Discrimination in a Person Following Robot," *IWINAC 2011*, Part I, LNCS 6686, pp. 222-232, 2011.

[22] Bradski, G., and Kaehler, A., "Learning OpenCV: Computer vision with the OpenCV library," *O'Reilly Media*, 2008.

Learning in real robots from environment interaction

P. Quintía, R. Iglesias, M.A. Rodríguez, C. V. Regueiro and F. Valdés

***Abstract*—This article describes a proposal to achieve fast robot learning from its interaction with the environment. Our proposal will be suitable for continuous learning procedures as it tries to limit the instability that appears every time the robot encounters a new situation it had not seen before. On the other hand, the user will not have to establish a degree of exploration (usual in reinforcement learning) and that would prevent continual learning procedures. Our proposal will use an ensemble of learners able to combine dynamic programming and reinforcement learning to predict when a robot will make a mistake. This information will be used to dynamically evolve a set of control policies that determine the robot actions.**

***Index Terms*—continuous robot learning, robot adaptation, learning from environment interaction, reinforcement learning.**

I. Introduction

ON line robot learning and adaption is a key ability robots must have if we really want them working in everyday environments. Robots must become part of everyday life as assistants, be able to operate in standard human environments, automate common tasks, and collaborate with us. Robotic devices are meant to become a nearly ubiquitous part of our day-to-day lives. Despite the increasing demand for personal robots able to educate, assist, or entertain at home, or for professional service robots able to sort out tasks that are dangerous, dull, dirty, or dumb, there is still an important barrier between the latest developments on research robots and the commercial robotic applications available. To overcome the frontier amongst commercial and research robots we believe that, like humans, robots should be able to learn from their own experiences when emulating people or exploring an environment. The mistakes and successes the robot makes should influence its future behaviour rather than relying only on predefined rules, models or hardwire controllers. This will result in robots that are able to adapt and change according to the environment. These robots should not use pre-defined knowledge, on the contrary most of their competences should be learned through direct physical interaction with the environment and human observation.

From our experience working with robots, we think that it is true to say that no matter how perfect our robot controller is, there are always unexpected situations or different environments that will make our robot fail. We always promoted the use of reinforcement learning as an interesting paradigm that can be used to learn from robot-environment interaction [7]. Nevertheless, the application of reinforcement learning algorithms still suffer from the same problem just described. Generally, reinforcement learning is applied to get a robot learning a behaviour on simulation and, once the robot-controller is learnt it is placed on the real robot. Nevertheless, if the real robot misbehaves, it would be necessary to investigate the reasons behind the robot mistakes and to learn the behaviour once again trying to include situations similar to those that caused the failure. There are some previous publications that highlight the interest of real robot learning from reinforcement and for different applications [8], [9]. Nevertheless, most of these works achieve the desired behaviour through a careful parametrization of the action space or the reinforcement function but they do not re-design the classic reinforcement learning algorithms to get real-time learning processes. Some other works deal with the problem of real robot learning from scratch. One of them is [1]. In this work the robot explores the environment and it uses its own experiences to learn a model of the environment. This model is used to improve the control policy and thus achieve quasi-online reinforcement learning. In our case we will explore on-line learning without models, i.e. without probabilistic transition matrices.

We are interested in getting continuous learning procedures that are never stopped. The idea is that robot would move in the environment using a behaviour learnt on simulation, nevertheless, the robot would be able to adapt and modify the behaviour according to the environment where it is moving. The achievement of continuous learning requires the development of systems able to fulfil three characteristics:

1) The learning must be as fast as possible
2) Every time the robot encounters new problems, it will have to learn and improve the controller. Nevertheless, this should not cause important instabilities or make the robot forget important aspects of what had been learnt before
3) It should be possible to incorporate new knowledge or destroy old one, at any time, without causing important robot misbehaviours

II. Achieving Fast Learning processes

We need robots that are able to learn the suitable action they must carry out for every different situation (state) they might encounter, and thus reach a particular behaviour. Reinforcement Learning is suitable to get robots learning from their own experiences and environment interaction. Nevertheless, from our previous work [2], [3], [4], [5], [6] we know that reinforcement learning is too slow and requires too many trials to learn a particular task. This makes its application on a real robot almost impossible. Due to this, instead of building a learning system that needs to determine the suitable action

M.A. Rodríguez, R. Iglesias, P. Quintía are with the Centro de Investigación en Tecnologías de la Información de la Universidade de Santiago de Compostela (CITIUS). Campus Vida. Universidade de Santiago de Compostela. E-mail: roberto.iglesias.rodriguez@usc.es

C. V. Regueiro is with the Department of Electronics and Systems, of the Universidade de A Coruña.

F. Valdés is with the Electronics Department, Polytechnic, University of Alcalá.

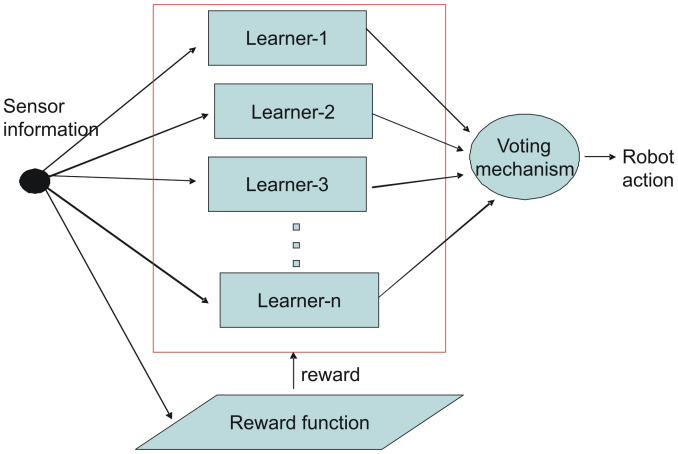

Fig. 1. Ensemble of independent learners to achieve fast learning processes

.

for every state of the robot, we prefer to build an ensemble of parallel learners able to determine, each one of them, the interval of actions most suitable for each state of the robot [3], [4], Figure 1. There is a clear example to illustrate this: Imagine a very diligent student who after three hours studying learns by heart the multiplication tables and makes no mistakes when the teacher asks him. On the other hand imagine a group of not-so-diligent students who prefer to have fun and prepare the exam at the last moment. In this case, given the lack of time, each student decides to learn only a random selection of times tables. These students can still get a good mark if they make the exam altogether. Whenever the teacher enquires about the result of a multiplication the answer is the number voted by the majority of the students. The qualification will be good provided that each student studied a random selection of the multiplication tables and those students who don't know the answer to the teachers question make it up (random guess and independent answers). In this case, given the question of the teacher, those students who didn't study the right times tables will not agree (most probably situation), while the rest of the students will agree with each other thus achieving majority.

Therefore, we have built an ensemble of independent learners like the one shown in Figure 1. This ensemble will use a voting mechanism to decide the action to be executed by the robot at every instant. Each one of the learners will have to learn a mapping between world states and actions. This mapping, also called policy, enables a robot to select an action base upon its current world state.

A. Using Fuzzy ART networks to building a representation of the world

Each learner of the ensemble shown in Figure 1 will have to build a map between world states and actions. This is a problem that lies at the heart of many robotic applications. This mapping, also called policy, enables a robot to select an action based upon its current world state. Therefore, the first problem to deal with is how to represent the world through a finite set of states. In our case, and as we can see in Figure 2, each learner will build a representation of the environment that will dynamically increase to include new situations that have not been seen before. We shall call to these new and distinguishable situations, detected in the stream of sensor inputs, states. This dynamic representation of the environment will be independent for each learner, i.e., each learner can see the world differently from the others. To quantify the sensor space we decided to use a The Fuzzy Adaptive Resonance Theory (Fuzzy ART) [10] to build the state representation for each learner. The FuzzyART clusters robot sensor readings into a finite number of distinguishable situations that we call *states*. Therefore, the FuzzyART network will achieve a sensor-state mapping that will dynamically increase to include new situations, detected in the stream of sensor inputs, and that have not been seen before.

Basically the FuzzyART will divide the sensor space into a set of regions (Vector Quantization). Each one of these regions will have a representative or prototype representing it. The FuzzyART works on the idea of making the input information resound with the representatives or prototypes of the regions into which the network has divided the sensor space so far. We call to these regions, states. If resonance occurs between the input and any of the states, this means that they are similar enough; the network will consider that it belongs to this state and will only perform a slight update of the prototype, so that it incorporates some characteristics of the input data. When the input does not resound with any of the stored states, the network creates a new one using the input pattern as its prototype.

The input of the Fuzzy ART will be an *M*-dimensional vector, where each of its components is in the interval [0, 1]. In our case, the input data we are dealing with comprise the information provided by a laser rangefinder and sonar sensors, but other sources of information are valid (e.g. grey levels of an image, or joint angles in a robotic arm). The prototypes of the states will be codified as arrays of M dimensions with values in [0, 1]: $w_j = (w_{j1}, , w_{jM})$. We shall use the letter N to refer to the number of states learnt by the network so far.

The behaviour of the Fuzzy ART is determined by two parameters: learning rate $\beta \in [0, 1]$; and a vigilance parameter $\rho \in [0, 1]$. The way the Fuzzy ART network operates can be summarised in the following steps (there are some important differences in comparison with the general proposal described in [10]):

1) After presenting an input $\mathbf{I}$ to the network, there will be a competitive process after which the states will be sorted from the lowest activation to the highest. For each input $\mathbf{I}$ and each state *j*, the activation function T_j is defined as

$$T_j(\mathbf{I}) = \frac{|\mathbf{I} \wedge w_j|}{|w_j|} \tag{1}$$

the fuzzy operator AND $\wedge$ is $(x \wedge y)_i \equiv min(x_i, y_i)$ and the norm $|\cdot|$ is defined as

$$|x| \equiv \sum_{i=1}^{M} |x_i|. \tag{2}$$

2) The state with the maximum activation value will be selected to see if it resounds with the input pattern **I**

$$J = arg\_max_j\{T_j : j = 1...N\}. \tag{3}$$

3) The Fuzzy ART network will enter in resonance if the matching between the input **I** and the winning state **J** is greater or equal than the vigilance parameter ρ:

$$|\mathbf{I} \wedge w_J| \geq \rho|\mathbf{I}| \tag{4}$$

If this relation is not satisfied, a new state will be created and the new prototype vector will be equal to the input **I**.

4) When the network enters in resonance with one input, the prototype vector w_J is updated:

$$w_J^{(new)} = \beta\mathbf{I} + (1-\beta)w_J^{(old)}. \tag{5}$$

A proliferation of states can be avoided if inputs are normalised:

$$|\mathbf{I}| \equiv \gamma, \forall \mathbf{I}, \gamma > 0 \tag{6}$$

The complement coding normalisation rule achieves normalisation while preserving amplitude information. The complement coded input *I* to the recognition system is the 2M-dimensional vector

$$\mathbf{I} = (\mathbf{a}, \mathbf{a}^c) \equiv (a_1, ..., a_M, a_1^c, ..., a_M^c), \tag{7}$$

where $a_n^c = 1 - a_n$. Using complement coding, the norm of the input vector will always be equal to the dimension of the original vector.

The vigilance parameter ρ is the most important parameter for determining the granularity of the classification. Low values for the vigilance parameter will create few classes. As the value of ρ approaches one, there will be almost one state for each sensor reading.

Each learner of the ensemble that we suggest (Figure 1) will use a near-random vigilance value to build a state representation from the sensor inputs. Since the vigilance parameter is different for each learner, so will be the partition of the sensor space into regions; the size of the regions into which the sensor space is divided will change from learner to learner, Figure 2. This helps to get a better generalization during the learning process.

Other artificial neural networks, such as the *Echo State Networks* [11] have been used in the past to learn from robot-environment interaction. Nevertheless, these networks are most appropriate to learn from demonstrative processes in which a user teaches the robot the desired control policy. In our case we need to use unsupervised techniques able to quantify the sensor space in a set of regions according to how similar the values coming from the sensors are, the best action for every one of these states will have to be discovered by the robot.

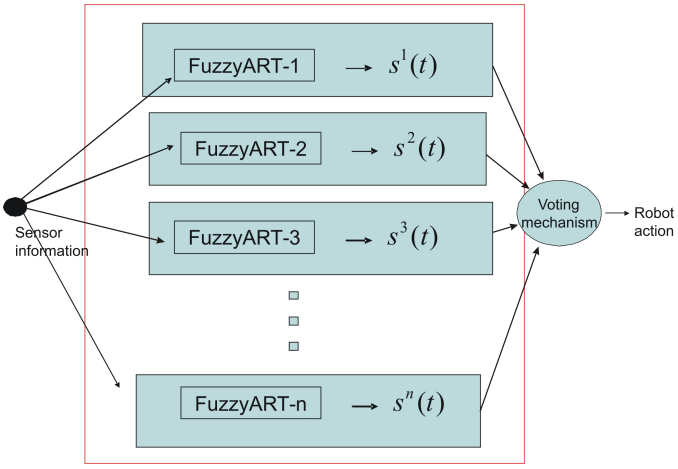

Fig. 2. Each learner of the ensemble builds a its own representation of the sensor space, i.e. the current state of the world is not described in the same way for all learners

.

B. Policy derivation and performance

As mentioned before, each learner of the ensemble (Figure 1) will achieve a control policy that maps states into actions. Therefore, besides the problem representing the environment (described in the previous section), we also need to deal with the problem of how to represent the robot actions. In our case, and similarly to what happened with the states, each control policy divides the action space A in a set of intervals. Nevertheless, this set of intervals is different for each control policy ($A^1, A^2, ..., A^N$), Figure 3. We decided to use different partitions of the action space for each learner in response to the necessity of improving the generalization ability of our proposal. Each partition A^l, $\forall l = 1, ..., N$ is built randomly, but it must verify the following properties:

$$\begin{aligned} &A^l = ... \\ &... = \{A^l(1) = [a_1^l, b_1^l), A^l(2) = [a_2^l, b_2^l), ... \\ &..., A^l(P) = [a_P^l, b_P^l]\}, \forall l = 1, ..., N \end{aligned} \tag{8}$$

- A^l covers all the action space, A, $\forall l = 1, ..., N$:

$$\cup_j A^l(j) = A$$

- A^l is a pairwise disjoint collection of intervals, $\forall l = 1, ..., N$:

$$A^l(j) \bigcap A^l(k) = \phi$$

The cardinality (number of intervals) is the same for all partitions:

$$cardinal(A^i) = cardinal(A^j), \quad \forall i, j = 1, ..., l$$

Therefore, in our case, a control policy π is a function that determines for every possible state of the robot, the interval of actions that seems to be suitable for the task. Since we used a different FuzzyART network for every learner, i.e., the sensor-state mappings are different for each learner, and so are the actions intervals, it is true to say that:

$$\begin{aligned} \pi^l : \quad & S^l \quad \rightarrow \quad A^l \\ & s \in S^l \quad \rightarrow \quad \pi^l(s) \in A^l \end{aligned} \tag{9}$$

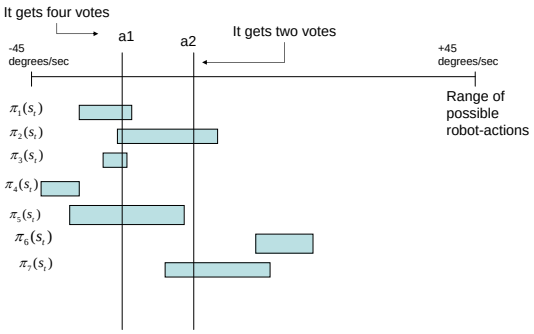

Fig. 4. Example of a voting procedure. Each learner suggests an interval of actions. The action most voted is the one the robot finally executes.

The final action the robot performs is selected after a voting procedure that considers these control policies (Figures 3 and 4). The action most voted is the one that the robot finally performs.Although there are a large variety of techniques to combine different sources of information, we selected *majority vote* since it is one of the simplest and most known strategies.

There is still an unanswered question: how does each learner know which interval of actions vote at each state?. The answer to this question is a utility function of states and action-intervals called Q. Each learner l from the ensemble will learn a utility function Q^l. These utility functions $Q^1, ..., Q^N$, are functions of states an action-intervals: $Q^l(S^l \times A^l)$. Basically each value $Q^l(s \in S^l, A^l(j))$ represents how good is executing any action a, included in the interval $A^l(j)$, when the robot is in state s. To learn these functions we have used the algorithm *Improving Time before a Robot Failure* [3], [4]. In this algorithm $Q^l(s^l, A^l(j))$ represents the expected time interval before a robot failure when the robot starts moving in $s \in S^l$, performs an action of the interval $A^l(j)$, and follows an optimal control policy thereafter:

$$Q(s, A^l(j)) = E[-e^{(-Tbf(s_0=s^l \in S^l, a_0=a \in A^l(j))/50T)}], \quad (10)$$

where $Tbf(s^l, A^l(j))$ represents the expected time interval (in seconds) before the robot does something wrong, when it performs any action $a \in A^l(j)$ in s^l, and then it follows the best possible control policy. T is the control period of the robot (expressed in seconds). The term $-e^{-Tbf/50T}$ in Eq. 10 is a continuous function that takes values in the interval $(-1, 0)$, and varies smoothly as the expected time before failure increases.

If we are able to predict the consequences of performing any action of a particular interval of actions in a state (time that will elapse before the robot makes a mistake), it is straightforward that we can achieve the best control policy by simply selecting the interval with the highest Q-value for every state. This will give us the control policy that maximizes the time interval before any robot failure; this is called greedy

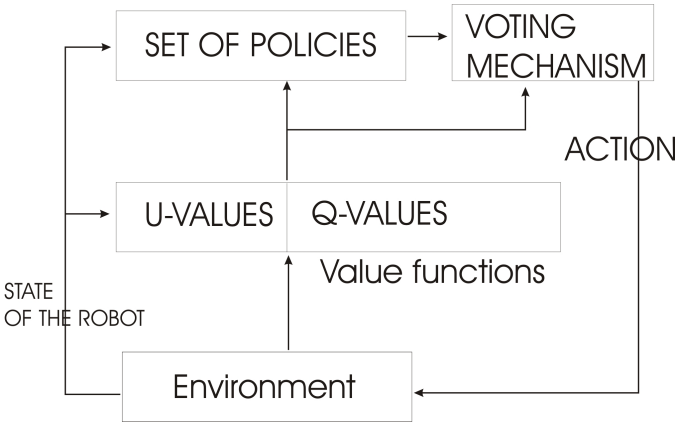

Fig. 5. General schema of our proposal

policy π^*:

$$\pi^{l*}(s^l) = arg\_max_k\{Q^l(s^l, A^l(k))\}, \quad , \forall s^l \in S^l \quad (11)$$

There is a greedy policy for every control policy in our system. Since the $Q^l(,)$ values and $Tbf(,)$ are not known, we can only refer to their current estimations $Q^l_t(,)$ and $Tbf_t(,)$. Starting from the Q-values we can determine the expected time before a robot failure for an action-state pair:

$$Tbf^l_t(s, a) = -50 * T * Ln(-Q^l_t(s, A^l_a)), \quad (12)$$

where A^l_a is the interval of A^l that contains action a. We need to use the super-index l for Tbf^l_t since it is estimated from the Q^l values. We can also determine the expected time before a robot failure for a state:

$$Tbf^l_t(s^l) = max_k\{-50 * T * Ln(-Q^l_t(s^l, A^l(k)))\}, \quad (13)$$

We can notice that the time before failure, determined for a given state, considers the best expectation, i.e., when the robot follows the greedy policy.

Initially, each learner will build a control policy that coincides with its corresponding greedy policy:

$$\pi^l(s \in S^l) = \pi^{1*}(s \in S^l), \quad \forall l = 1, ..., N \quad (14)$$

These greedy policies $\pi^{1*}, ..., \pi^{N*}$ are got from the Q-values following the straightforward process indicated in Eq 11. The final action the robot performs would be selected after a voting procedure (Figure 4). Nevertheless, to increase the robustness of our proposal, the voting procedure considers a new value-function (Figure 5) that determine how good is a particular policy for a given state, i.e., whether the interval of actions that particular policy suggests for the state seems to be right for the task or not. We represent this new value function with the letter U. Thus $U^j(s) = 1$ means that the j-control policy is invalid for state s, and hence its vote can be discarded. On the contrary $U^j(s) = 0$, means that the control policy j seems to be right for the state s, and therefore its vote must be taken into account. Eq. 15 summarizes the voting procedure:

$$a_t = argmax_{\forall a \in A}\{\sum_1^N \delta[a \notin \pi^l(s^l(t))] * (1 - U^l(s^l(t)))\} \quad (15)$$

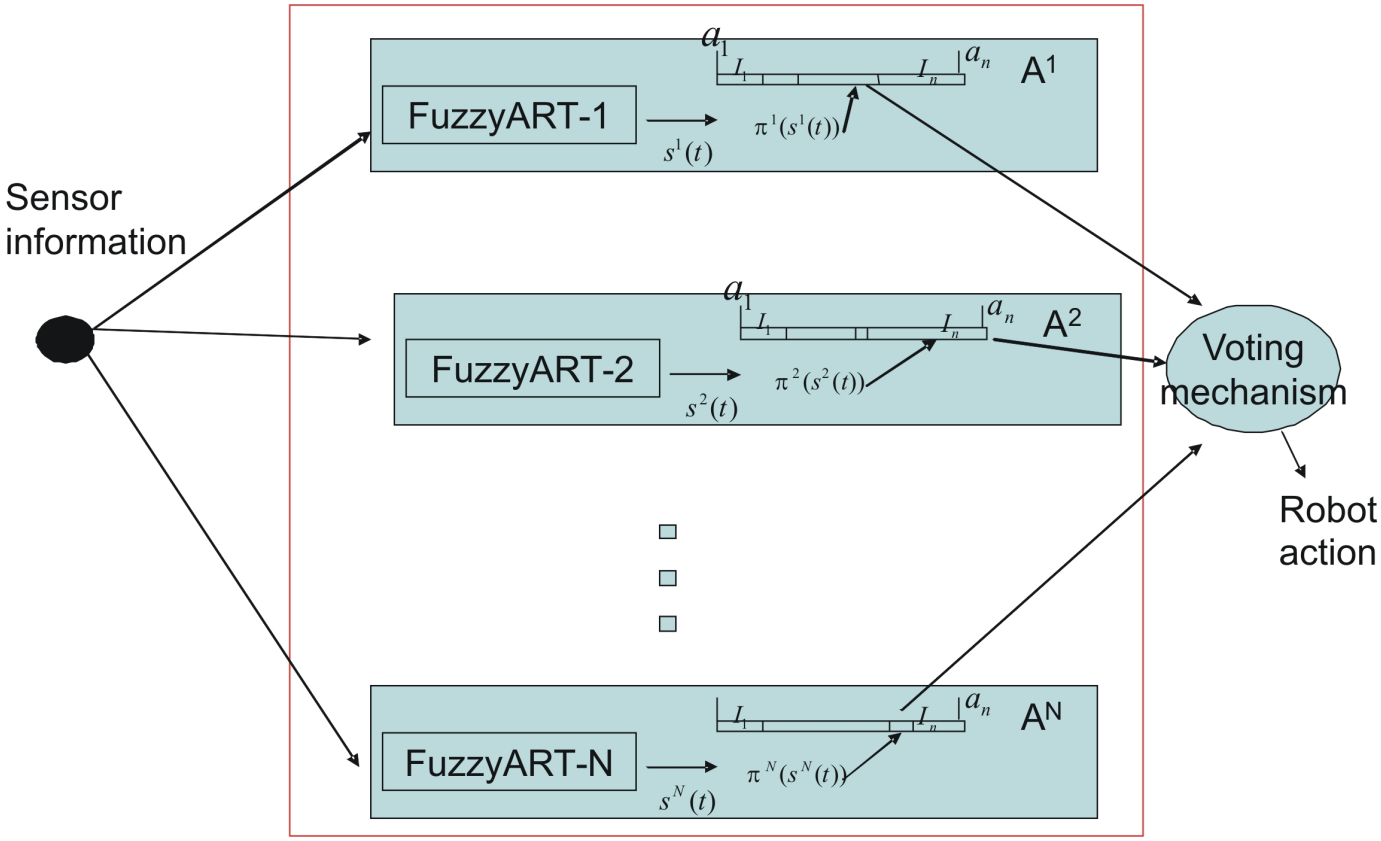

Fig. 3. In our proposal each learner of the ensemble builds a quantization of the sensor space and the action set that is different and independent from the rest of the learners.

.

δ is the kronecker delta, i.e:

$$\delta[a \notin \pi^l(s^l(t))] = \begin{cases} 1 & if a \in \pi^l(s^l(t)) \\ 0 & if a \notin \pi^l(s^l(t)) \end{cases} .$$

As we will see in the next subsection, the new utility function U will also determine when the control policies $\pi^1, ..., \pi^N$ are updated using the greedy policies $\pi^{1*}, ..., \pi^{N*}$.

C. Learning of the Utility Functions

As we have described in the previous section, the action the robot should execute at each instant is determined using a voting procedure that is highly influenced by two utility functions: Q and U (Figures 5 and 6). Both utility functions are independent for each learner of the ensemble: $Q^1, ..., Q^N$ and $U^1, ..., U^N$. In this section we will describe how the values of all these utility functions are inferred from the robot experiences when it interacts with the environment.

When the robot is moving interacting with the environment performing different actions, we can dynamically update the Q-values. Thus, if we consider a robot that has performed the action a_t in the current state s_t, and as an outcome the robot has moved to state s_{t+1} and has received the reinforcement r_t, the Q^l values corresponding to every policy, can be updated taking only into account the relationship amongst consecutive states:

$$Q^l_{t+1}(s^l_t, A^l_a) = \begin{cases} -e^{-1/50} & \text{if } r_t < 0 \\ Q^l_t(s^l_t, A^l_a) + \delta & \text{otherwise} \end{cases} \quad (16)$$

where,

$$\delta = \beta_L(e^{\frac{-1}{50}} * Q^l_{max}(s_{t+1}) - Q^l(s_t, A^l_{a_t})). \quad (17)$$

r_t is the reinforcement the robot receives when it executes action a_t in state s_t, $\beta_L \in [0, 1]$ is a learning rate, and it is the only parameter whose value has to be set by the user. Finally, $A^l_{a_t}$ is the interval of A^l that contains action that has been performed by the robot, a_t.

Equations 16 and 17 allow the updating of the Q-values using the experiences of the robot. Basically the robot would move in the environment, trying the execution of different actions and, at every instant, the Q-value of the last action performed by the robot would be updated according to the current state of the robot and the reinforcement that the robot received. Nevertheless, this would lead to very slow learning processes. A common solution to speed up this learning consists on updating the Q-value of each action performed by the robot considering not only the immediate consequences of the execution of this action, but also what the robot does and the reinforcement it receives during a short interval after the execution of the action which is being considered. Thus, to obtain the utility values $Q^l(s, A^l(k))$, the robot begins with an initial set of random values, $Q^l(s, A^l(k)) \in [-1, -0.95]$, and then it initiates a stochastic exploration of its environment. The robot will move and collect data during a maximum period of time or until it makes an error. The data collected will later be used to update the Q-values:

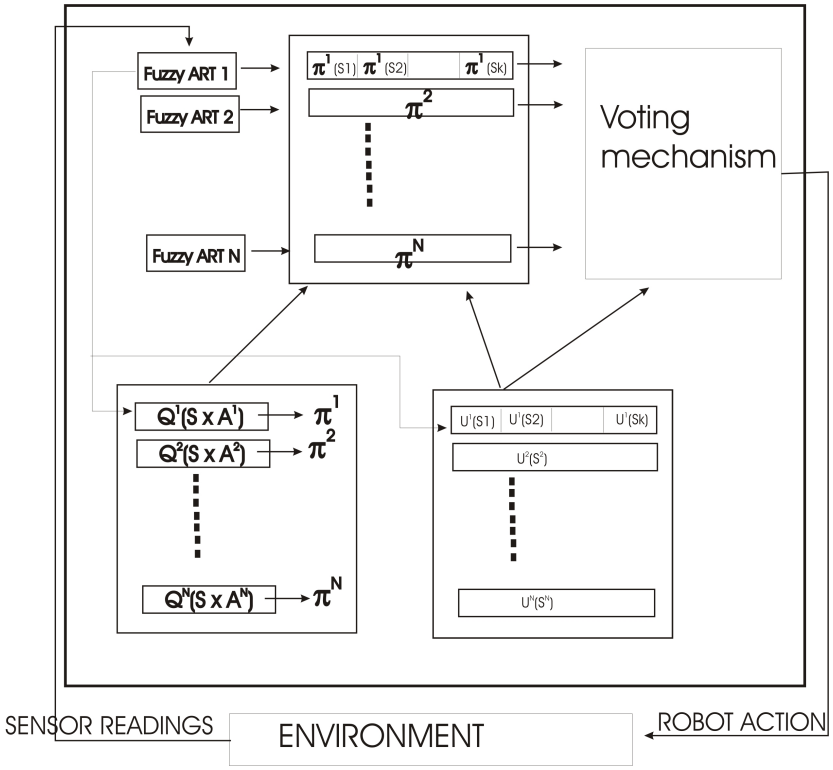

Fig. 6. Detailled schema of our proposal

First stage, collecting data

1) $m = 0$
2) At each instant t and while the robot does not receive a negative reinforcement or moves for a maximum period of time do:
   a) Observe the current state in every learner of the ensemble, $s^l(t)$.
   b) Select an action a_t to be executed by the robot (through a voting procedure).
   c) Perform action a_t, observe new states s^l_{t+1} and reinforcement value.
   d) If $r_t >= 0$ then $m \leftarrow m + 1$ and go back to the first step, otherwise stop collecting data and do not shift the m-index

Second stage, updating the Q-values:

1) Update *time before failure* for every learner of the ensemble:
   a) for $k = 0, 1, \ldots, m$ do:
   if $k = 0$ then:

$$Tbf^l = \begin{cases} T & \text{if } r_{t-k} < 0 \\ Tbf^l(s_{t-k}) & \text{otherwise} \end{cases}$$

   else $Tbf^l \leftarrow \lambda * (Tbf^l + T) + (1 - \lambda) * Tbf^l(s_{t-k})$.
2) Update the Q-values of the first record in the experience set:
   - $\Delta Q^l_t(s_{t-m}, A^l_{a_{t-m}}) = \beta_L(-e^{-Tbf^l/50T} - Q^l_t(s_{t-m}, A^l_{a_{t-m}}))$, $\forall l = 1, .., N$
3) Shift the index *m* which represents the number of states recorded in the experience set and which have not been updated: $m \leftarrow m - 1$
4) if $m = 0$ exit, otherwise go to the first step, 1)

The parameter λ (second stage, first step) is a parameter that determines the relative importance of the last robot-environment interactions. The higher the values of lambda, the more the Q-values are altered considering the last robot-environment interactions. Low values of lambda are recommended when either the rewards or the sensor readings are noisy.

The updating of the new utility function, *U*, is straightforward: those policies that voted for the actions the robot executed far from any failure will see their *U* values decreased. On the contrary, those policies that voted for the actions executed just before a robot failure will see their *U* values increased, thus reducing their future consideration for the same states. Finally, those policies that voted for actions that finally weren't executed will not see their *U* values altered. This way of updating the *U* values means that the robot will tend to repeat the same sequences of actions that it has already tried in the past and did not cause any trouble, but the robot will change its behaviour whenever the actions it carried out lead the robot to some kind of failure.

Figures 5 and 6 give an overview of our system. Each time the robot makes a mistake and it receives negative reinforcements, the set of policies used to control the robot will evolve trying to improve robot's behaviour. Basically,

each policy will mutate in those actions that have lost their vote ($U^j(s)$ is not null), in this case the new interval of actions will coincide with the greedy policy:

- for i=1,...,N
  - for j=0,...,maximum number of states in S^i
    - if $U^i(j) > 0$ $\pi^i(s^i(j)) = \pi^{i*}(s^i(j))$

The use of the greedy policies to update the control policies, allows faster and faster recoveries from robot misbehaviours due to the use of the past experiences learnt by the robot. On the other hand, we must be aware of the fact that any time the robot makes a mistake the use of the function U limits the number of mutations and prevents the system from falling into heavy instabilities. This is something new that allows us to improve the generalization ability of our system.

To get independent and uncorrelated policies, only a subset of policies will take part in the decision making at each instant. This subset of M control policies ($M < N$) that takes part on the decision making, is determined randomly, and it will be the same until the end of the learning process. Nevertheless, every time the robot makes a mistake this subset is increased with some new randomly selected control policies. Adding new control policies during the learning process represent a way of incorporating new knowledge into the system (this was the third characteristic mentioned in the introduction and that continual learning would require). These new policies are initially random but they start learning and evolving from the moment they are incorporated into the system.

III. Experimental results

We have performed an experimental study of the strategy described in the previous section. In particular we have implemented it on a Pioneer 3DX robot. This robot is equipped with a SICK LMS-200 laser rangefinder, a ring of 16 ultrasound sensors and bumpers. In all the experiments the linear velocity of the robot was kept constant (15.24 cm/s $\equiv$ 6 inch/s), and the robot received the motor commands every 300ms (value of T in Eq. 10). Through the experiments the robot had to learn a common reactive task: wall following. To teach the robot how to follow a wall located on its right at a certain distance interval, we used a reinforcement signal that is negative whenever the robot goes too far from or too close to the wall being followed. We tested the algorithm in both, real and simulated environments. We must emphasize that the behaviour itself is not the objective of this work, but a benchmark for all the tests we want to carry out.

A. Simulation results

Fig.7 shows the simulated environment where the robot learnt the wall following behaviour. In this figure we can also see the trajectory of the robot once the task has been learnt. To simulate the movement of the robot we have used Player&Stage [14]. We consider that the behaviour has been learnt when the robot is able to move without making any mistake during ten minutes (several laps in the same environment). The learning time is the time elapsed since the robot starts moving until it achieves a valid control policy (the policy that is able to move the robot without mistakes for at least ten minutes).

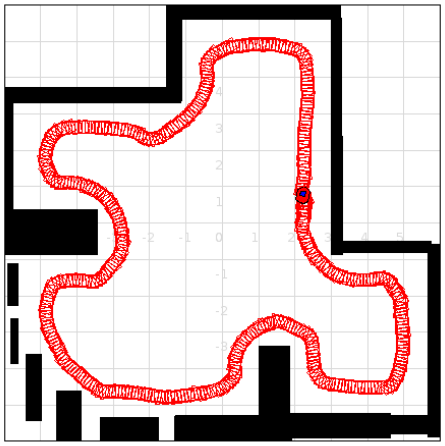

Fig. 7. A robot learning how to follow a wall located on its right. This graph was obtained during the last stages of the learning process

To learn the behaviour we used all the sensor information available. The laser sensor we used on simulation provides 177 measures – one measure per degree from 0º to 176º –. On the other hand we also used the information provided by 16 ultrasound sensors surrounding the robot. We then scaled the values in the interval [0,8] to the interval [0, 1):

$$I_i = \frac{1}{(1 + input_i)} - \frac{1}{9} \tag{18}$$

The learning parameters we used are: learning coefficient $\beta_l = 0.25$, $\lambda = 0.5$, the vigilance parameter for each FuzzyART neural network was selected randomly in the interval $[0.86, 0.92]$. and, finally the learning coefficient for the FuzzyART neural network was $\beta = 0$. The average learning time after 30 experiments was 14.55 minutes and the standard deviation was 8.8 minutes.

After this first set of experiments we carry out a second set of 30 experiments, but this time the processing of the sensor information was different. In this second case we scaled the sensor values using Eq. 19:

$$I_i = \frac{(8 - input_i)^2 * e^{\frac{-input_i}{1.5}}}{64} \tag{19}$$

In this second case, the vigilance parameters were selected randomly in the interval $[0.8, 0.92]$, the rest of the parameters kept the same values as in the first set of experiments. After 30 experiments we got an average learning time of 6.61 minutes and the standard deviation was 4.45 minutes.

In general, through the simulated experiments we noticed the importance of the U-function described in sections (B and C). This utility function works as a gate enabling or disabling learners in the voting mechanism. We noticed that the use of this function allows achieving the desired control policies sooner that if we do not use it.

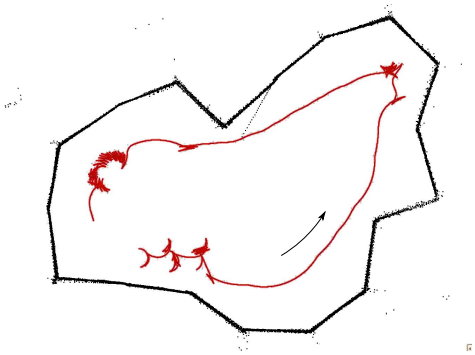

Fig. 8. First lap of a real robot learning the wall following behaviour in a real environment. The robot moves a bit backwards every time it makes a mistake and receives negative reinforcement, this can be appreciated in the robots trajectory.

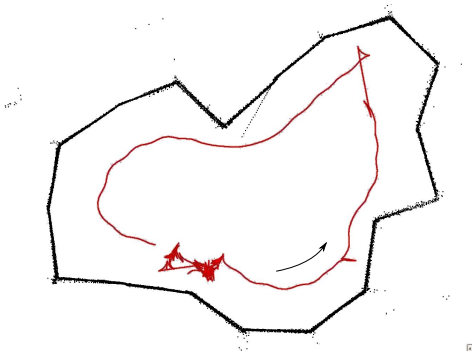

Fig. 9. Third lap of a real robot learning the wall following behaviour in the same real environment as in Figure 8.

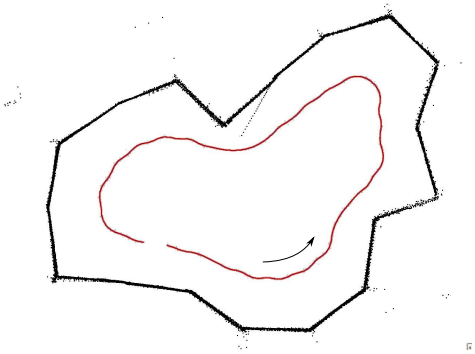

Fig. 10. Fourthlap of a real robot learning the wall following behaviour in the same real environment as in Figure 8.

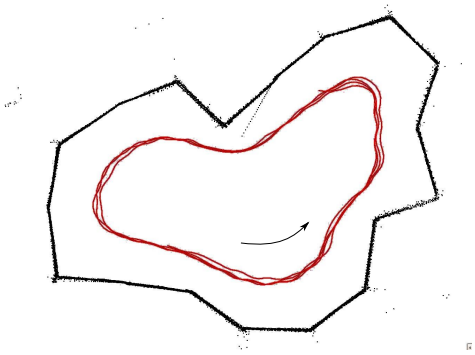

Fig. 11. In this figure we can observe several laps of a real robot following the wall located on its right in a real environment. We can appreciate the robots trajectory during several laps (from the fourth onwards) once the behaviour has been practically learnt.

B. Learning on the real robot

Since our learning algorithm proved to be very efficient and very fast on simulation, we decided to apply it to learn the behaviour on a real robot. It is really relevant to be aware of the fact that the learning process starts from scratch (i.e., our robot does not have any kind of prior knowledge about the task or how to solve it). This can be easily appreciated in the first lap of the robot in the environment, shown in Figure 8. Every time the robot receives a sequence of negative reinforcements, it stops moving and it goes back until it reaches a position where it does not receive negative reinforcements. Figures 8,9, 10 and 11 show the progress of the learning procedure on a real robot working on a real environment. We must emphasize that these results represent a huge achievement since little work has been done with learning processes on a real robot interacting with the real environment. Moreover, these learning processes on the real robot overcome the limitations of the simulation, i.e., when the simulator is not realistic enough the behaviour learnt might not work on the real robot.

IV. Conclusion

In this paper we have described a system that moves us close to continuous reinforcement learning procedures in a real robot operating in real environments. Basically our system combines a set of control policies that is being evolved considering two utility functions. These two utility functions learn the experiences accumulated by the robot when it interacts with environment, and prevents too unstable behaviours every time the robot encounters a situation it had not seen before. It is also important to mention the fact that our system incorporates new knowledge dynamically, as the learning process progresses (this new knowledge is included as new learners in the ensemble). This not only does not cause any instability but, on the contrary, helps the robot to behave better and better improving both, robustness and generalization. The experimental results achieved, show the achievement of extremely fast learning process able to work even on real robots learning continuously in a real environment.

It is important to be aware of the fact that our system learns the representation of the environment (states created by the Fuzzy ART neural networks) and the control policy (i.e. the actions the robots must execute for each state) simultaneously. This obviously increases the complexity of the problem, although a quantification of the increase of the complexity is still part of our future work.

Acknowledgment

This work was supported by the research grants TIN2009-07737 and INCITE08PXIB262202PR

References

[1] B. Bakker, V. Zhumatiy, G. Gruener, J. Schmidhuber. *Quasi-Online Reinforcement Learning for Robots*. Proceedings of the International Conference on Robotics and Automation (ICRA-06), Orlando, Florida, 2006.

[2] M. Rodriguez, R. Iglesias, C. V. Regueiro J. Correa and S. Barro, *Autonomous and fast robot learning through motivation*, Robotics and Autonomous Systems, vol. 55, pages: 735–740, 2007.
[3] Pablo Quintia, Roberto Iglesias, Carlos V. Regueiro, Miguel A. Rodriguez, *Simultaneous learning of perception and action in mobile robots*, Robotics and Autonomous Systems, vol 58, pages: 1306–1315, 2010.
[4] M. A. Rodriguez, R. Iglesias, P. Quintia C. V. Regueiro, *Parallel robot learning through an ensemble of predictors able to forecast the time interval before a robot failure*, XI Workshop of Physical Agents, 2010.
[5] T. Kyriacou, R. Iglesias, M. Rodriguez, P. Quintia. *Unsupervised Complexity Reduction of Sensor Data for Robot Learning and Adaptation*, 11th Towards Autonomous Robotic Systems (TAROS'2010). 2010
[6] Pablo Quintia, Roberto Iglesias, Miguel Rodriguez, Carlos Vazquez Regueiro, *SIMULTANEOUS LEARNING OF PERCEPTIONS AND ACTIONS IN AUTONOMOUS ROBOTS*, 7th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2010), 2010.
[7] R. S. Sutton, *Reinforcement learning: An introduction*, MIT Press, 1998.
[8] Thomas Kollar, Kicholas Roy, *Using reinforcement learning to Improve Exploration Trajectories for Error Minimization*, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2006), 2006.
[9] Andrea L. Thomaz, Guy Hoffman, Cynthia Breazeal, *Real-Time Interative Reinforcement Learning for Robots*, AAAI 2005 Workshop on Human Comprehensible Machine Learning, 2005.
[10] G. A. Carpenter, S. Grossberg,D. B. Rosen, *Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system*, Neural Networks, volume 4, pages: 759–771,1991.
[11] M. Oubbati, B. Kord, and G. Palm, *Learning Robot-Environment Interaction Using Echo State Networks*, SAB 2010, LNAI 6226, pp. 501-510, 2010
[12] Amanda J.C. Sharkey, *Combining Artificial Neural Nets: Ensemble and Modular Multini-Net Systems*. Springer. 1999.
[13] Lior Rokach, *Pattern classification using ensemble methods*. World Scientific. 2010.
[14] *Player & Stage Project*. http://playerstage.sourceforge.net. (Accessed on 26 February 2012).