XVII Workshop of PHYSICAL AGENTS

Book of Proceedings

Málaga, Spain - June 2016

Coord.: J. Ballesteros, A. Bandera, J.P. Bandera, R. Marfil, A. Romero-Garcs, L.V. Calderita
Edits: Servicio de Publicaciones de la Universidad de Málaga

Cover design: Antonio Bandera

Prints: Ruiz de Aloza
Impreso en España - Printed in Spain

ISBN: 978-84-608-8176-6 (Versión papel)
Legal deposit:

Foreword

Robotics is a wide research topic, which covers very different and specific areas. You can be interested on vision or audio processing, or in software architectures and middlewares, or in how a robot must navigate through a dynamic environment meanwhile it simultaneously interacts with the people in the surroundings. The possibilities for research and application are practically infinite. But all of them must converge within a physical agent that have a continuous and fluent interaction with its environment. And this must be our real objective and mission, and also our testbench. Sometimes, the impossibility of extending the time (other very interesting field of research!) makes difficult to address in parallel all our everyday activities, which usually merges teaching and administrative issues with our research activities. Sometimes, it is also needed to take part of this time for preparing the petition of a new project, to close transference projects and contracts, or to rewrite the paper that we need to ask for a new sexenio (and that the reviewer does not understand although it is clear that it is close to perfection!). But in other situations, it is possible to enjoy the time with our passion for robotics, and design a new software or hardware component or test a new option. Or to take some days with other friends in the forum provided by the workshop of Physical Agents, the WAF.

WAF2016 is a collaborative project, made possible by the help of many people. Firstly, it is possible thank to the authors and reviewers. They provide the core of WAF2016, a collection of revised contributions whose reading version constitutes this book of proceedings. But whose interactive version could be only deeply enjoined at the physical workshop. Thus, we also want to thank the School of Telecommunication Engineering of the University of Malaga for letting us use their facilities, where the workshop core is embodied, and to the Plan Propio of the University of Malaga, Robotnik and Avnet for their necessary support. Finally, we would like to thank to Miguel Angel and Vicente for being a fast source of advices and help. And also to Pablo Bustos, for endowing us within this great and nice community.

June 2016 The Organizing Committee

Organization of the WAF2016 workshop

Program Committee

Workshop Organizers:

Joaquin Ballesteros
Antonio Bandera
Juan Pedro Bandera
Adrian Romero-Garces
Luis V. Calderita

Scientific Committee

Adriana Tapus
Agapito Ledezma
Alberto J. Bugarin Diz
Anna Belardinelli
Antonio Fernandez Caballero
Antonio Gonzalez Muoz
Camino Fernandez Llamas
Carlos Vazquez Regueiro
Cristina Urdiales Garcia
David Filliat
Diego Viejo Hernando
Domenec Puig Valls
Eduardo Zalama Casanova
Eugenio Aguirre Molina
Fernando Fernandez Rebollo
Francisco Martin Rico
George Azzopardi
Giulio Sandini
Humberto Martinez Barbera
Ismael Garcia Varea
Jesus Martinez Gomez
Jim Little
Joao F. Ferreira
Joaquin Lopez Fernandez
Jorge Cabrera Gamez
Jorge Dias
Jose Daniel Hernandez Sosa
Jose Maria Armingol Moreno
Jose Maria Cañas Plaza
Jose Vicente Soler Bayona
Judith van der Spank
Lluis Ribas i Xirgo
Luis Miguel Bergasa Pascual
Luis Rodriguez Ruiz
Manuel Mucientes Molina
Miguel Angel Cazorla Quevedo
Miguel Angel Garcia Garcia
Miguel Garcia Silvente
Nikolai Petkov
Pablo Bustos Garcia de Castro
Pedro Nuñez Trujillo
Pilar Bachiller Burgos
Rafael Barea Navarro
Rafael Muoz Salinas
Roberto Iglesias Rodriguez
Vicente Matellan Olivera
Victor Gonzalez Castro

Sponsoring Institutions

University of Málaga, Vicerrectorado de Investigación y Transferencia
Robotnik
Avnet

Table of Contents

Social interactive robots

Visual Perception

Navigating through the environment

Robotic middlewares

CLARC: a Robotic Architecture for Comprehensive Geriatric Assessment

Antonio Bandera, Juan Pedro Bandera, Pablo Bustos, Luis V. Calderita, Álvaro Dueñas, Fernando Fernández, Raquel Fuentetaja, Ángel García-Olaya, Francisco Javier García-Polo, José Carlos González, Ana Iglesias, Luis J. Manso, Rebeca Marfil, José Carlos Pulido, Christian Reuther, Adrián Romero-Garcés, Cristina Suárez

***Abstract*—Comprehensive Geriatric Assessment (CGA) is an integrated clinical procedure to evaluate frail old people status and create therapy plans to improve their quality and quantity of life. In this paper we present CLARC, a mobile robot able to receive the patient and his family, accompany them to the medical consulting room and, once there, help the physician to capture and manage their data during CGA procedures. The hardware structure of CLARC is based on a robotic platform from MetraLabs. The software architecture of the system incorporates a deeply tested framework for interactive robots. This framework, by encoding the whole CGA session using Automated Planning, is able to autonomously plan, drive, monitor and evaluate the session, while also managing robot navigation and data acquisition. CLARC incorporates a series of sensors allowing to collect data automatically, using non-invasive procedures. The healthcare professional can use the platform to automatically collect data while addressing other tasks such as personal interviewing, data evaluation or care planning. First trials will be carried out in hospitals in Seville and Barcelona in June and July 2016, respectively.**

***Index Terms*—Gerontechnology, Automated Planning, Intelligent Robots, Medical Robotics, Human-Robot-Interaction**

I. Introduction

COMPREHENSIVE Geriatric Assessment (CGA) is a powerful procedure for the evaluation and treatment prescription of frail older people. CGA first evaluates the patient's clinical, functional, environmental and psychosocial status, and compares its temporal evolution. Then, an overall treatment and follow-up plan is prescribed. CGA is an interdisciplinary effort involving the coordination of different medical staff, which is being carried out all over the world, with the aim of increasing both the quality and quantity of life of frail adults. Some of the benefits of CGA are improving the diagnostic, creating right, customized and proportional therapeutic plans, increasing functional autonomy, and also reducing complications during hospitalizations and mortality. Giving the aging of the world population, with about 810 million people over 60 in 2012, which is expected to grow to more than 2 billions in 2050, CGA importance, and costs related to it, are by no doubt going to be increased.

CGA procedures vary from hospital to hospital but in general they are carried out every 6 months, involve both patients and relatives, and are made of 3 different types of activities: clinical interview, multidimensional assessment and customized care plan. During the clinical interview patient and relatives comment with the physicians about the elder health problems. Next, multidimensional tests are performed to evaluate the overall patient status. In questionnaire-based tests, the patient or relatives answer some questions about patient daily life and his/her ability to perform some activities without help. Depending on the answers a score is given to the patient. The *Barthel* Index test [9] is an example of such tests. Another type of tests involve the observation of the patient performing some activities, like in the *Get Up and Go* test [11], where the patient is asked to get up from the chair, walk for a few meters and come back to the original place. Finally, based on the evidences gathered during the two previous phases and the patient's evolution from the last CGA session, physicians create a personalized care plan to be followed until the next review. A typical CGA session lasts about 3 hours, and there are many parts that could be parallelized or automatized, especially during the multidimensional assessment. For example, some activities must be performed individually by both patient and relatives, so they can be run simultaneously at different rooms, and some tests do not need the presence of a physician to be performed.

In this paper we present the architecture and preliminary implementation of CLARC[1], an autonomous robotic solution to support CGA. It provides a web graphical interface allowing the physicians to specify the tests to be answered by a patient during a CGA session. Once the multidimensional assessment is designed, the robot is able to perform and mark the tests by interacting or observing the patient, store the results, and maintain a record the physician can use to design the treatment plan. CLARC, using both speech recognition and touch-screen interaction, is able not only to automatically collect data from patients and relatives by conducting questionnaires and interview-based tests, but it is also able to perform direct observation (face expressions, body pose and motion, and speech parameters), needed in observation-based tests. By encoding

Antonio Bandera, Juan Pedro Bandera, Rebeca Marfil and Adrián Romero-Garcés are with University of Málaga. E-mail: {ajbandera, jpbandera, rebeca, argarces}@uma.es

Fernando Fernández, Raquel Fuentetaja, Angel García-Olaya, Francisco Javier García-Polo, Ana Iglesias, José Carlos González and José Carlos Pulido are with University Carlos III de Madrid. E-mail: {ffernand, rfuentet, agolaya, fjgpolo, aiglesia, josgonza, jcpulido}@inf.uc3m.es

Pablo Bustos, Luis V. Calderita and Luis J. Manso are with University of Extremadura. E-mail: {pbustos, lvcalderita, lmanso}@unex.es

Christian Reuther is with MetraLabs. E-mail: christian.reuther@metralabs.com

Álvaro Dueñas and Cristina Suarez are with Hospital Universitario Virgen del Rocío. E-mail: alvaro.duenas.ruiz@hotmail.com and cristina.suarez.exts@juntadeandalucia.es

[1] http://echord.eu/essential_grid/clark/

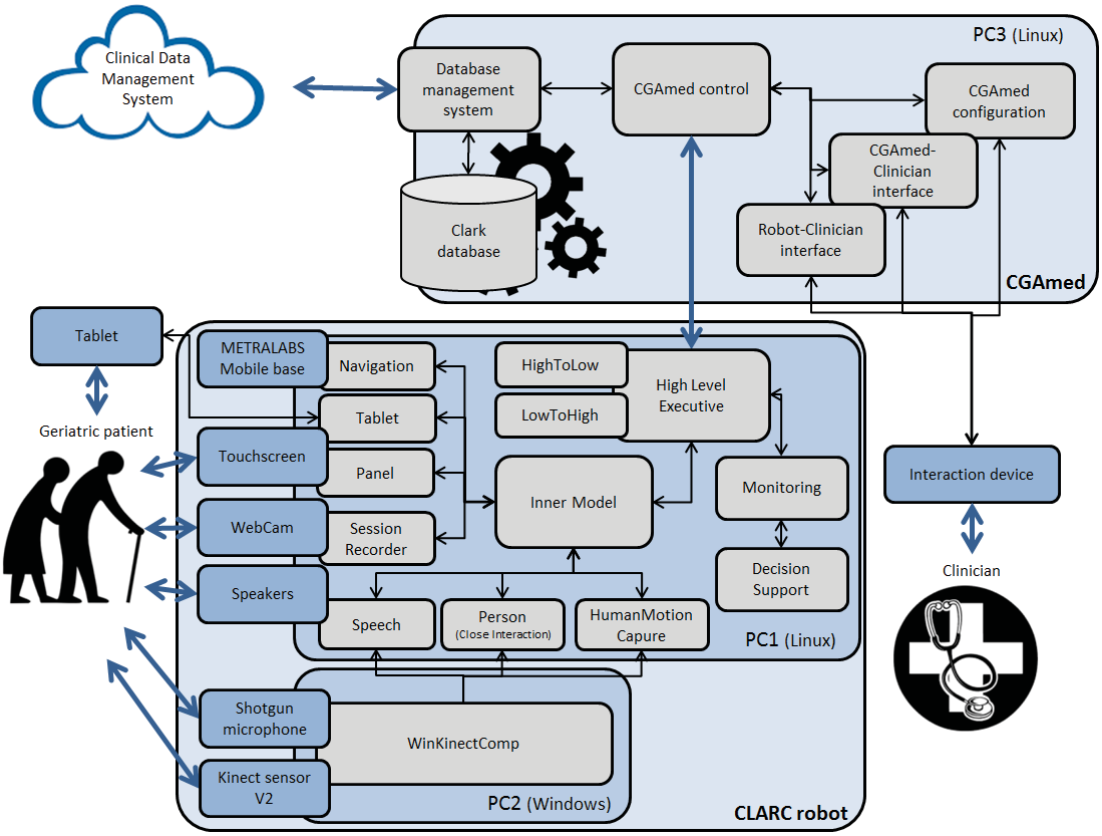


Fig. 1. CLARC conceptual architecture

the whole assessment cycle using Automated Planning it is able to autonomously work, without any help, adapting the test to patient behavior and managing the unexpected events that can appear during a session.

The remaining of the paper is structured as follows: Section II presents the overall system architecture, describing the robot, the human-robot interaction modules, the deliberative module, the interface and the integration with the Clinical Data Management System. An example of a typical CGA session using CLARC is shown in Section III. The current status of the system is described in Section IV, while Sections V and VI describe the related work, and the future developments and the conclusions, respectively.

II. System Architecture

The architecture of the CLARC system is shown in Figure 1. CLARC is composed of several modules, running either on the robot or on external PCs, including the clinician's PC. A total of 3 computers support the architecture, two of them are part of the robot and are used to control it and to interact with patients. The third computer is placed outside the robot and supports the database system and the system-clinician interface, what we called CGAmed.

From a conceptual point of view the system can be divided into three main components; the Robot, the Cognitive Architecture and the CGAmed software. The robot is a mobile platform, based on the MetraLabs SCITOS G3, and equipped with extra sensors to be able to seamlessly perform and record tests and interact with patients and relatives. The cognitive architecture, running on-board the robot, provides it with the needed intelligence to perform its tasks. CGAmed supports the interface of the clinician both with the robot (to configure for example the tests to be performed) and with the generated data (patient profile, recorded sessions, tests marks, etc.). Connection of the robot and the CGAmed is done in two ways. The main link connects the *High Level Executive* of the robot's cognitive architecture to the *CGAmed Control* module. The later commands the former to switch on the remaining robot modules and transfers the information about the tests to be performed. All the configuration information and the results of the session travel through this connection. More details are provided in Section III. Although it is not shown in the figure, there is also a second direct link between the *Session Recorder*

Fig. 2. CLARC robot prototype design

module of the robot and the database.

A. The Robot

A MetraLabs SCITOS G3 platform is being adapted to meet the requirements of the defined use case. The outer shell is currently being redesigned to accommodate the new sensors and to customize it for the specific CGA needs. The robot's locomotion is based on a differential drive system consisting of two powered wheels and a caster wheel for stability. This enables the robot to rotate on the spot and drive at speeds of up to 1 m/s, if necessary. The platform contains a 40Ah lithium battery which allows for up to 18 hours of autonomous operation, and can be recharged fully within 4 hours. A safety bumper socket sensor around the outer perimeter of the robot's shell is used to prevent the robot from exerting force against animate or inanimate objects. The platform is fitted with a LIDAR sensor for localization, navigation and obstacle avoidance.

The SCITOS G3 platform is extended with an extensive human-machine-interface, consisting of a Microsoft Kinect V2 sensor, a shotgun microphone, a touch screen and speakers for multi-modal human-robot interaction, as well as a web cam for recording the sessions. The system is also provided with a external tablet mirroring the touch screen, that the patient can use to interact with the robot if desired. Figure 2 shows a prototypical adaption of the SCITOS G3 platform for the CLARC use case.

B. The Cognitive Architecture

CLARC robot benefits from using the RoboCog [1] cognitive software architecture to control its behaviour. RoboCog proposes a distributed architecture, where action execution, simulation, and perception are intimately tied together, sharing a common representation, the Inner Model In the CGA scenario, this internal representation of the robot, the patient and any other significant event captured from the outer world, is the central part of the architecture for action control. The rest of task-solving elements of RoboCog (the Panel, Tablet, Speech, etc. see Figure 1) use this central representation to share data at different abstraction levels, to get information about the user's state and to plan next actions.

The robot's course of action emerges from the activity of several networks of software components (called *compoNets*), which are connected to the Inner Model through a specific component (called the *agent*). Each compoNet is currently able to endow the robot with a specific ability. Some compoNets are connected to sensors, and they process their raw data to enrich the inner representation with fast perceptions. Some other ones are connected to actuators, which allow the robot to interact with its environment. It is usual that a compoNet manages either sensors or actuators, but this is not a requisite. For instance, the PELEA Deliberative compoNet, in charge of providing the planning, monitoring and high-level learning abilities, works over the data stored in the Inner Model or the CGAmed central server.

All the architecture runs in a Linux computer, interacting via the shared inner representation. That means that there is no direct connection between compoNets, which continuously check the data contained in the Inner Model, update it and act in consequence. Figure 1 also shows the existence of a second PC within the robot. It runs the *WinKinectComp* component, which is in charge of handling the data coming from the Kinect sensor and the microphone. It processes and provides a continuous stream of information to those compoNets that need the data related to the person in front of the robot, namely *Speech* recognition, *Person* (Close Interaction) and *HumanMotionCapture*.

1) Patient-Robot Interaction: The so called low-level components of the robot provide the necessary functionality to perform the Patient-Robot interaction. They are driven by the changes on the inner representation, which could be provoked by the Deliberative compoNet (see Section III) or by an incoming perception. Furthermore, the results of the actions are also added to the Inner Model, allowing the Deliberative module to reason about them. The collection of compoNets initially planned to be included within the software architecture includes:

- The *Panel* and *Tablet* compoNets, which manage the interaction with the patient via the touchscreen and the tablet, respectively. The tablet is specially useful in tests, as the *Mini-Mental* one, where there are questions where the patient is asked to hand-write.
- The *Speech* compoNet manages the verbal communication with the patient, being able to both speak and hear to the patient. Patient-robot interaction is redundant in the sense that information is usually shown to the patient using text and voice simultaneously, and the answer can be received also by voice or by selecting on the touchscreen. Accessibility issues have been taken into account to customize both the information provided and the feedback modes to the particular needs of frail older people. Both verbal and graphical interfaces are multi-language.
- The *Person* compoNet is in charge of detecting and

tracking the person sitting in front of the robot (upper-body motion capture, including hands and fingers).
- The *HumanMotionCapture* compoNet is the responsible of capturing the whole motion of the person, providing information about all joints. It is necessary for addressing tests such as the *Get Up & Go*.
- The *Session Recorder* component, as said, manages the data of the on-board webcam to record both the audio and video of the session. The video is temporarily annotated by the deliberative module and stored into the database. That way the clinician can review the video of any session and analyze the patient behavior.

2) Autonomy: The Deliberative module is based on the PELEA [8] Automated Planning and Learning architecture. Complementing the low-level detection of exogenous events, the use of Automated Planning allows to control the robot, providing it with full autonomy. Automated Planning (AP) aims to find an ordered sequence of actions that allows the transition from a given initial state to a state where a series of goals are achieved. The use of AP for robotic control in clinical applications has been tested in previous works [5], where it demonstrated its ability to conduct rehabilitation sessions with children suffering from cerebral palsy in a fully autonomous way. The Planning Domain Definition Language (PDDL) [12] is used to describe the environment and the actions the robot can perform, both in terms of predicate logic. Describing an action in PDDL is as simple as enumerating its preconditions and effects. Preconditions are the facts that must be true in a certain state for the action to be applicable, while effects are the new facts appearing and the facts that are no longer true after the action is applied. The environment is also modeled using predicate logic to describe the objects, their properties and relationships. Adding new actions or facts or changing the existing ones is done easily by just editing a text file. Figure 3 shows an example of an action for the *Barthel* test. Several instances of this action are executed at the introductory part of the test, since introducing the test implies to execute several introductory acts. This is the general action for all of them. Specifically, for the introduction labeled as `?i`, this action can be applied only if there is no external cause that prevents continuing the test (predicate `can_continue`), the robot has been introduced (predicate `finished_introduce_robot`), and the introduction `?i` is the next one, following the test order (next two preconditions). The parameter `?pause` represents the pause in seconds that the robot should perform after executing the action. The effects of the action are that this part of the introduction has finished `(introduction_finished ?i)` and the system is ready for the next part of the introduction, if any.

At the beginning of the execution the Deliberative module receives the goals to pursue from the CGAmed module (for example: perform to patient Paula Smith a *Barthel* test in room A and a *Mini-Mental* [4] test in room B starting at 9:30 am). Taking into account these goals and the description of the environment contained in the Inner Model, a new planning problem is created and a plan is returned. The plan includes the high-level actions the robot has to perform to achieve the goals, for example greet the patient, introduce the test to be performed, say the first question, wait for the answer, etc. Non-expected states and exogenous events are contemplated, so the robot is able to, for example, repeat a question if the patient does not answer, ask him/her to seat if he/she stands up or call the physician if something wrong is detected. Figure 4 shows an example of the first part of a plan generated for the *Barthel* test. In this plan, the actions contain labels, as `INTRO1` or `Q1_S1`, that represent specific acts for the robot. The action 1 refers to an interruption of the test since the patient has not been detected, so the robot should call him/her. The next action is executed when the cause of the interruption has been solved, which allows to continue with the test.

```
(:action introduce-test
 :parameters (?i - introduction ?p - pause)
 :precondition (and
        (can_continue)
        (finished_introduce_robot)
        (is_test_introduction ?i)
        (= (current_introduction)
           (test_introduction_position ?i))
        (after_pause ?i ?p) )
 :effect (and
         (introduction_finished ?i)
         (increase (current_introduction) 1)))
```

Fig. 3. Example of a PDDL action for the *Barthel* test

```
0: (CONFIGURE-TEST BARTHEL SPANISH PATIENT PRESENT)
1: (INTERRUPT BARTHEL PATIENT_ABSENT ROBOT_CALL_PATIENT)
2: (RESTORE-FROM ROBOT_CALL_PATIENT PATIENT_ABSENT)
3: (INTRODUCE-ROBOT ROBOT_PRES1 THE_ROBOT PAUSE_0SG)
4: (INTRODUCE-TEST INTRO1 PAUSE_1SG)
5: (INTRODUCE-TEST INTRO2 PAUSE_1SG)
6: (INTRODUCE-TEST INTRO3 PAUSE_1SG)
7: (INTRODUCE-TEST INTRO4 PAUSE_1SG)
8: (START-QUESTION Q1_S1 Q1 PAUSE_1SG)
9: (SHOW-QUESTION-OPTION Q1_O1 Q1_O1 Q1 FIRST PAUSE_1SG)
10: (SHOW-QUESTION-OPTION Q1_O2 Q1_O2 Q1 FIRST PAUSE_1SG)
11: (FINISH-QUESTION Q1_E1 Q1 PAUSE_10SG)
12: (ASK-FOR-ANSWER Q1_A1 Q1)
13: (RECEIVE-ANSWER Q1_A1 Q1 DUR_6SG)
14: (FINISH-ASK-ANSWER-SUCCESS Q1)
15: (MAKE-QUESTION-TRANSITION Q1_T Q1 PAUSE_4SG)
...
```

Fig. 4. Example of the first part of a possible plan for the *Barthel* test

The plan involves changing the Inner Model for provoking the required response from the low-level components or to read this representation for determining the current state of the world. Thus, actions like switch the camera on, say a given phrase, show a given text at the touchscreen, receive a verbal answer, etc. are translated to inner events on the model that the rest of compoNets must solve. These events are complementary to other external ones such as a motion of the person's face, which is reactively solved by the Person compoNet. Other actions such as determining the position of the patient's arms are solved by examining the Inner Model. For example if the patient is absent as in the plan above, the speech component will receive the "call the patient" action. The Person compoNet, which was the responsible of informing of the absence, will detect whether the patient comes back. Once the patient is again seated, Person compoNet will add the information to the Inner Model, so the Deliberative module

can continue with the next action of the plan.

Following the PELEA structure, the components of the Deliberative module are the following:

- The *High Level Executive* (HLE) module receives the goals from the CGAmed system and invokes the Monitoring module to get a plan achieving them. Then it takes the first action of the plan and invokes the HighToLow module to decompose it into low-level actions understandable by the low-level components of the robot. These actions are then inserted into the Inner Model and executed by the low-level components. Those components update the Inner Model with the results of the actions. HLE looks at the changes in the Inner Model and, after a conversion to high level knowledge performed by LowToHigh, sends them to Monitoring that checks whether the plan is executing conveniently.
- The *Monitoring* module, maintains a high level model of the environment and is in charge of invoking the Decision Support module if any deviation in the execution of the plan arises. It detects for example that the user has not answered a question or is not facing the robot and tries to find alternate plans to solve the problem found.
- The *Decision Support* module creates a plan starting from the current state, the goals to be achieved, the possible states of the world and the description of the changes the actions produce in the world state, all of them expressed in PDDL. To create the plan it invokes an automated planner that returns the sequence of actions achieving the goals. Using PDDL allows the planner to be changed seamlessly, thus benefiting from any improvement in the planning community.
- The *HighToLow* module converts the high level actions of the plan created by the Decision support module into low level actions that can be included into the Inner Model.
- The *LowToHigh* module converts the information contained in the Inner Model, which represents knowledge in the form of binary predicates (see Section II-B4) into n-ary predicates that the Monitoring module uses to reason about the correctness of the execution of the plan.

3) Navigation: Autonomous navigation is realized using MetraLabs' proprietary navigation software CogniDrive. CogniDrive consists of modules for localization, navigation and obstacle avoidance. The localization module uses an adaptive particle filter to track multiple position hypotheses at the same time, and therefore allows for accurate localization even when faced with ambiguity in the sensor data. The navigation module uses an adaptive derivative of the A* planning algorithm to generate global paths, which are adapted on a local scale by the immediate temporal history of local obstacle measurements. Local scale path planning and obstacle avoidance is addressed using the established Dynamic Window Approach (DWA). CogniDrive allows the definition of no-go areas that are to be avoided in local and global path planning, as well as the definition of speed areas which can limit the robot's movement speed in critical environments. MetraLabs has deployed over 200 autonomous mobile robots using CogniDrive, with over 60,000km of autonomous driving experience in complex and crowded environments.

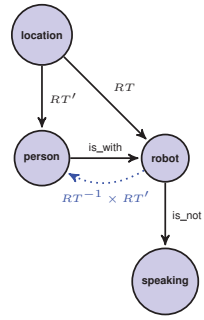


Fig. 5. Unified representation as a multi-labeled directed graph. Edges labeled as `is_with` and `is_not` denote logic predicates between nodes. Edges starting at `location` and ending at `person` and `robot` are geometric and encode a rigid transformation (RT' and RT respectively) between them. Geometric transformations can be chained or inverted to compute changes in coordinate systems.

4) The Inner Model: The Inner Model is a multi-labeled directed graph which holds symbolic and geometric information within the same structure. Symbolic tokens are stated as logic attributes related by predicates that, within the graph, are stored in nodes and edges respectively. Geometric information is stored as predefined object types linked by 4×4 homogeneous matrices. Again, they are respectively stored as nodes and edges of the graph. Figure 5 shows one simple example. The `person` and `robot` nodes are geometrical entities, both linked to the `location` (a specific anchor providing the origin of coordinates) by a rigid transformation. But, at the same time that we can compute the geometrical relationship between both nodes ($RT^{-1} \times RT'$), the `person` can be located (`is_with`) close to the `robot`. Furthermore, an agent can annotate that currently the `robot is_not speaking`.

C. The CGAmed module

The CGAmed module manages the communication of the clinician and the robot and provides access to the data stored in the platform. Its components are:

- The *Robot Clinician Interface* provides the clinician with the tools needed to configure a CGA session and to monitor it in real time. It is developed as a web interface that can be accessed from the clinician's computer or from a tablet. Figure 6 shows the interface to schedule a CGA test for a patient. The clinician can select a patient from the list of registered ones and schedule a test for him/her, specifying the time and location where it will take place. Additional parameters, as for example asking the patient about his/her state 6 months ago instead of today, can be also configured. Figure 7 shows the monitoring screen. This interface allows the clinician to start, pause and remotely monitor a CGA session performed by the robot. A live video of the session is shown, as it is recorded by the *Session Recorder* module. Also the log of the session is shown on the right upper

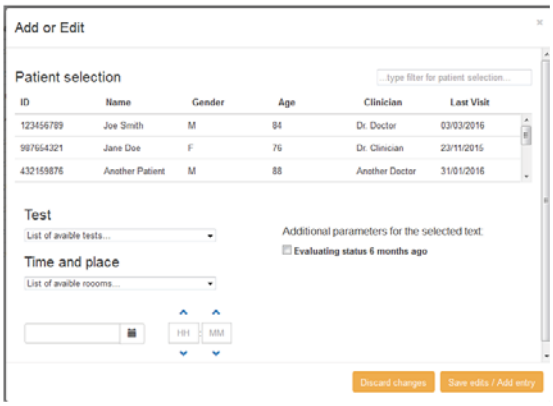


Fig. 6. Robot Clinician Interface: creating a new test

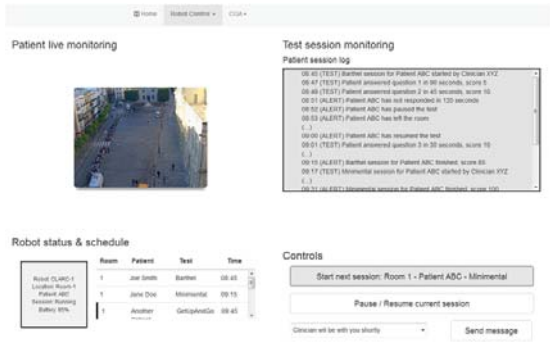


Fig. 7. Robot Clinician Interface: monitoring a live CGA session

part of the screen. The clinician can see a summary of the robot status and its schedule, including next tests to be performed by the robot. Finally, the interface provides the doctor with the ability to use the robot to interact in a limited way with the patient, by sending a series of predefined messages that will be reproduced by the robot speakers and touchscreen.

- The *CGAmed Clinician Interface* allows the clinician to access the clinical data stored into the system. Once a patient is selected, demographic data are shown, along with a list of past tests, including their scores and any additional information deemed important. The clinician can edit the tests to modify the automatic score set by the robot, viewing the video recorded for each part of the test, or comparing the videos for the same parts of tests performed in different dates to assess the patient evolution with time.
- The *CGAmed Configuration* allows the clinician to configure the system. Parameters as the system language can be set.
- The logic under the three former modules is provided by the *CGAmed control*, which is also the gateway to the rest of the system and to the database. This module communicates with the *High Level Executive* sending the information about the tests to be performed and receiving the monitoring data. It also controls the *Data Base Management System* and its integration with the Clinical Data Management System of the Hospital, via the Clinical Document Architecture (CDA) of HL7.

III. A CGA SESSION USING CLARC

From a clinician point of view, a CGA session using CLARC begins by login into the CGAmed web interface and creating the list of tests to be performed next (see Figure 6). When the patient or relative is ready to answer the test, the clinician press the start test button on the computer and accompanies him/her to the room where the robot is (in a near future we plan the robot to autonomously accompany the patient to the room). Once the robot detects the patient at the room, the test begins. The robot starts greeting the patient and explaining the purpose and structure of the test. If it is a question-based test, questions are presented both orally and on the touchscreen and patients can answer by voice or by selecting the right answer on the screen. In the case of an observation test, the robot asks the patient to perform the required activities and monitors its performance using the Kinect sensor. In both cases, the system automatically marks the patient performance and stores the scores into the database. The monitoring abilities of the software architecture allow CLARC to ask for help to the medical expert if needed and to recover from unexpected situations as the patient leaving the room, asking for help or not being able to give an appropriate answer for a question.

Meanwhile the clinician can monitor the session from his/her office (see Figure 7) and change the automatically set scores once the test is finished. Both scores are kept for tracking purposes (see Figure 8). Whichever the type of the test, the whole patient-robot interaction is video and audio recorded by the web cam and temporarily annotated by the Deliberative module. This allows the clinician to offline review the tests and to go directly to the video recording of any specific part of them, even doing side-by-side video comparison of the performance of the patient with that of previous CGA sessions.

From the system point of view, once the physician presses the start button, the tests to be performed and their configuration parameters (patient, room, etc.) are sent to the Deliberative component that creates a plan to fulfill them. It then commands the low-level components to perform the desired activities (introduce the robot, introduce the test, ask for a question, wait for an answer, monitor the patient movements...) by doing appropriate changes in the Inner Model. The low level components perform their tasks and update the Inner Model with the results. In turn, the Deliberative component sends updates about the current state to the CGAmed control module. Figure 9 shows a simplified sequence diagram of a use case where a clinician uses CLARC to perform a patient evaluation based on *Barthel* and *Mini-Mental* tests. It is a simplification since the low-level components of the architecture are not included, so many steps are skipped.

Fig. 8. Robot Clinician Interface: reviewing and editing a CGA session

IV. Current status

CLARC is still under development and the goal is to have a commercial system by 2018, supporting several CGA tests with total autonomy. Currently a first fully functional version of most of the components has been developed and the PDDL descriptions for the *Barthel*, *Mini-Mental* and *Get Up and Go* tests have been created. *Barthel* and *Mini-Mental* are mainly questionnaire-based tests and their scoring is done automatically by the system, although the physician can change the scores anytime. The *Get Up and Go* test needs the evaluation of the patient's body motion, so its scoring is done manually by the clinicians. A first prototype of the CGAmed interface is also ready. The robot is able to perform a complete *Barthel* test, considering the most frequent errors that could appear, as the patient not answering a question or leaving the room.

Trials with volunteer patients for the three previous tests will be conducted at Hospital Universitario Virgen del Rocío in Seville, Spain, in June 2016. In July, the system will be presented to clinicians of Hospital Sant Antoni Abat (Vilanova, Spain), where further tests will be performed to improve the system functionality and to adapt it to user preferences.

V. Related work

To our knowledge there is no currently any robotic system aimed to assist clinicians in performing CGA. CLARC was born in response to a competitive call launched by the The European Coordination Hub for Open Robotics Development (ECHORD++) project[2], where it competes against two other approaches ARNICA and ASSESSTRONIC. Two of these three approaches will continue to be funded after the first trial in Vilanova, July 2016. ARNICA[3] uses a Kompai robot to perform CGA, but no Artificial Intelligence capabilities seems to be provided. ASSESSTRONIC[4] also focuses more on the Human-Robot Interaction, including non-verbal interaction, than in the system intelligence.

[2]More info about the project can be found at http://echord.eu
[3]http://echord.eu/essential_grid/arnica/
[4]http://echord.eu/essential_grid/assesstronic/

Most systems designed to direct questionnaire filling tasks do not rely on the exclusive use of natural interaction channels, and force the user to employ a keyboard or a mouse device [6]. However, recent proposals in assistive robotics deny the use of these interfaces and focus on the use of artificial conversational systems, touch screens or a combination of both. One interesting example is proposed in the ALIAS Project[5]. Our proposal follows the same approach and uses only natural interaction channels (i.e. voice and touch). To our knowledge, these multimodal interfaces have not yet been applied for automated CGA processes.

Gait analysis, on the other hand, has been traditionally achieved using invasive approaches, such as marker-based motion capture systems. These systems are still the most popular option for medical or rehabilitation scenarios, but require a controlled environment and the user to wear specific markers [13]. One of the challenges for the current proposal is to effectively capture human motion using only the sensors mounted in the robot. Such a system will reduce setting up times and will be more comfortable for the user.

CLARC deliberative system can be considered a successor of NaoTherapist [5][10], a robotic platform for rehabilitation of children with cerebral palsy. NaoTherapist is able to autonomously plan, execute and monitor a rehabilitation session, made of different exercises the robot shows and the child imitates. While monitoring the exercises the system is able to fill some of the items of the QUEST [3] test. The gesture monitoring capabilities of NaoTherapist are somehow limited and there is no real verbal robot-child interaction, despite the robot is able to speak to encourage the kid. Robots and sensors, like Kinect, have been also used for rehabilitation sessions including patient monitoring and evaluation [7]. But the evaluation of the patients is done manually by the specialist on the basis of the recorded videos of the session.

On the other hand, the system uses algorithms, taken from previous research [2], to reinforce collected data using facial expression and body language analysis. The endowing of this software architecture within the hardware structure of CLARC is one of the most significant differences of the proposed system with respect to other competitors.

VI. Conclusions and Future work

In this paper we have presented CLARC an autonomous robot to help the clinician in the CGA process. CLARC discharges the clinician from some of the most time consuming CGA procedures, those of performing tests to patients and relatives, and allows him/her to focus on the most important part, the creation of a customized treatment and follow up plan. Currently a fully functional but restricted version of CLARC has been developed, allowing to perform basic *Barthel*, *Mini-Mental* and *Get up and Go* tests. During the next two years the system will be improved to obtain a commercial product, and several other CGA tests will be added.

The Deliberative component will be endowed with more complex execution monitoring features. The scoring process for observation-based tests will be automatically learnt from

[5]http://www.aal-europe.eu/projects/alias/

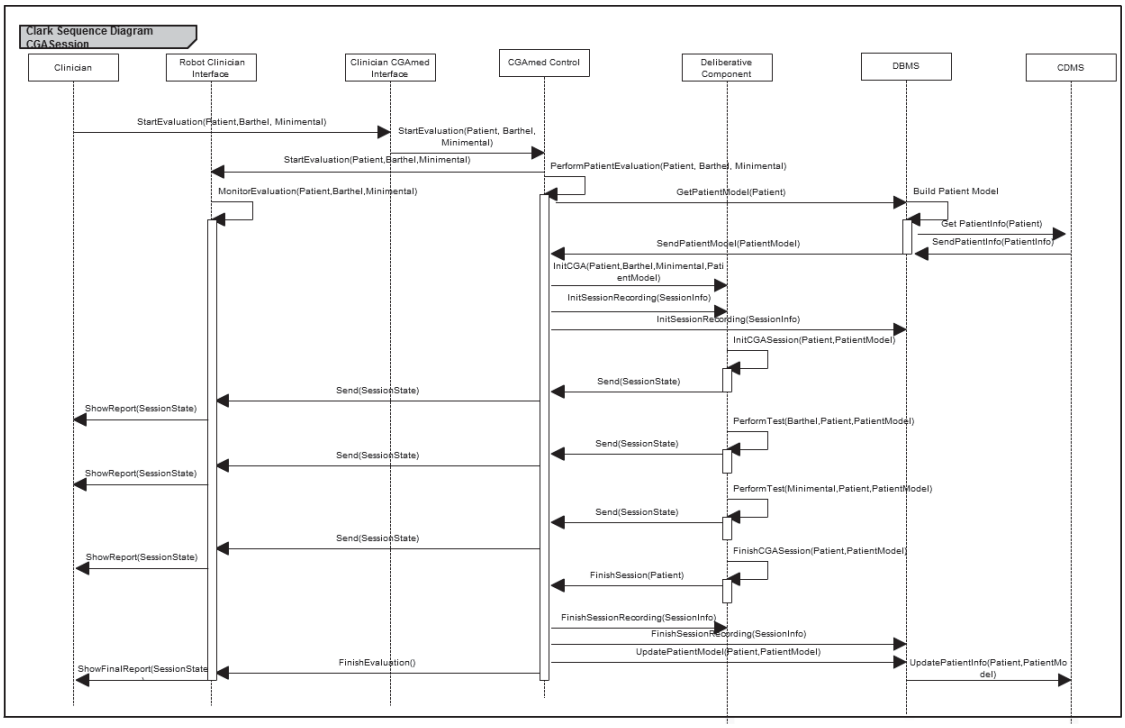


Fig. 9. A sequence diagram showing the interactions between components for a simplified CGA based on a *Barthel* and a *Mini-Mental* test

annotations of medical experts on real sequences using Machine Learning techniques. These techniques will be also used to parametrize the tests, for example learning the questions where patients need more time to answer, or further explanations. Also the whole use case, from patient greeting to good-bye will be encoded in PDDL and executed, reacting and generating new plans when something not expected happens.

ACKNOWLEDGMENT

This work has been partially funded by the European Union ECHORD++ project (FP7-ICT-601116) and the TIN2015-65686-C5-1-R Spanish Ministerio de Economía y Competitividad project.

REFERENCES

[1] Luis Vicente Calderita, Luis J Manso, Pablo Bustos, Cristina Suárez-Mejías, Fernando Fernández, and Antonio Bandera. Therapist: Towards an autonomous socially interactive robot for motor and neurorehabilitation therapies for children. *JMIR Rehabil Assist Technol*, 1(1):e1, 2014.

[2] F Cid, J Moreno, P Bustos, and P Nez. Muecas: A multi-sensor robotic head for affective human robot interaction and imitation. *Sensors*, 14(5):7711–7737, 2014.

[3] Carol Dematteo and Nancy Pollock. Quality of Upper Extremity Skills Test. *Physical Occupational Therapy in Pediatrics*, 13:1–18, 1992.

[4] M. Folstein, S. Folstein, and P. McHugh. Mini-mental state: a practical method for grading the cognitive state of patients for the clinician. *Journal of Psychiatric Research*, 12:189–198, 1975.

[5] José Carlos González, José Carlos Pulido, Fernando Fernández, and Cristina Suárez-Mejías. Planning, Execution and Monitoring of Physical Rehabilitation Therapies with a Robotic Architecture. In Ronald Cornet, Lúacrúamioara Stoicu-Tivadar, Alexander Hörbst, Carlos Luis Parra Calderón, Stig Kjær Andersen, and Mira Hercigonja-Szekeres, editors, *Proceedings of the 26th Medical Informatics Europe conference (MIE)*, volume 210 of *Studies in Health Technology and Informatics*, pages 339–343, Madrid (Spain), May 2015. European Federation for Medical Informatics (EFMI), IOS Press.

[6] LC Gray and R. Wootton. Comprehensive geriatric assessment "online". *Australasian Journal on Ageing*, 27(4):205–208, 2008.

[7] A Guneysu, RD Siyli, and AA Salah. Auto-evaluation of motion imitation in a child-robot imitation game for upper arm rehabilitation. *Robot and Human Interactive Communication, 2014 RO-MAN: The 23rd IEEE International Symposium on*, pages 199–204, 2014.

[8] César Guzmán, Vidal Alcázar, David Prior, Eva Onaindía, Daniel Borrajo, Juan Fdez-Olivares, and Ezequiel Quintero. Pelea: a domain-independent architecture for planning, execution and learning. In *Proceedings of ICAPS'12 Scheduling and Planning Applications woRKshop (SPARK)*, pages 38–45, Atibaia (Brazil), 2012. AAAI Press.

[9] F.I. Mahoney and D. Barthel. Functional evaluation: the Barthel index. *Maryland State Med Journal*, 14:56–61, 1965.

[10] Alejandro Martín, José C. González, José C. Pulido, Ángel García-Olaya, Fernando Fernández, and Cristina Suárez. Therapy monitoring and patient evaluation with social robots. In *Proceedings of the 3rd 2015 Workshop on ICTs for Improving Patients Rehabilitation Research Techniques*, REHAB '15, pages 152–155, New York, NY, USA, 2015. ACM.

[11] S. Mathias, USL Nayak, and B. Isaacs. Balance in elderly patients: the "get-up and go" test. *Arch Phys Med Rehabil*, 67:387–389, 1986.

[12] Drew McDermott. PDDL - the planning domain definition language. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998.

[13] W Tao, T Liu, R Zheng, and H Feng. Gait analysis using wearable sensors. *Sensors (Basel, Switzerland)*, 12(2), 2012.

Percepts symbols or Action symbols? Generalizing how all modules interact within a software architecture for cognitive robotics

R. Marfil, L.J. Manso, J.P. Bandera, A. Romero-Garcés, A. Bandera, P. Bustos, L.V. Calderita, J.C. González, A. García-Olaya, R. Fuentetaja and F. Fernández

***Abstract*—Robots require a close coupling of perception and action. Cognitive robots go beyond this to require a further coupling with cognition. From the perspective of robotics, this coupling generally emphasizes a tightly integrated perceptuomotor system, which is then loosely connected to some limited form of cognitive system such as a planner. At the other end, from the perspective of automated planning, the emphasis is on a highly functional system that, taken to its extreme, calls perceptual and motor modules as independent functions. This paper proposes to join both perspectives through a unique representation where the responses of all modules on the software architecture (percepts or actions) are grounded using the same set of symbols. This allows to generalize the signal-to-symbol divide that separates classic perceptuomotor and automated planning systems, being the result a software architecture where all software modules interact using the same tokens.**

***Index Terms*—cognitive robotics, inner representations, symbol grounding**

I. Introduction

ABSTRACT reasoning about phenomena from the outer world is intimately tied with the existence of an internal representation of this external reality. From a robotic perspective, this implies the establishment and maintenance of a connection between what the robot reasons about and what it can sense [5]. The Physical Symbol Grounding is defined as the problem of how to ground symbol tokens to real world entities, i.e. to percepts that can have a high dimensionality and, unfortunately, that can vary under different conditions. Furthermore, the dynamic essence of the outer reality can impose that these percepts continuously change over time. Of course, this is a challenging problem, approached from very different points of view by many researchers in recent decades (e.g. see some examples on the brief survey by Coradeshi et al. [5]). Among these proposals, recent contributions [2] are pointing towards the use of a shared, unique internal representation. This representation is fed with the symbol tokens generated by all the software components in charge of solving the grounding problem.

R. Marfil, J.P. Bandera, A. Romero-Garcés and A. Bandera are with University of Málaga. E-mail: {rebeca, jpbandera, argarces, ajbandera}@uma.es
L.J. Manso, P. Bustos, and L.V. Calderita are with University of Extremadura. E-mail: {lmanso, pbustos, lvcalderita}@unex.es
J.C. González, A. García-Olaya, R. Fuentetaja and F. Fernández are with University Carlos III of Madrid. E-mail: {josgonza, agolaya, rfuentet, ffernand}@inf.uc3m.es

Figure 1(left) shows a schematic view of RoboCog [4], [3], [11], a software architecture where this premise of maintaining a shared representation is hold. The figure depicts how two different agents interact through this representation for unfolding a 'go to person' behavior provided by the deliberative agent (in this specific case, the PELEA module [1]). Domain-dependent agents, in charge of performing the necessary physical and perceptual actions, use this plan and the shared world model to perform their activities. In this case, the Person agent detects the pose of the person and provides these data to the representation, and the Navigation agent takes these data and moves the robot. The shared representation is cooperatively built and kept updated by all the modules.

The use of the state of the world as the best mechanism to communicate software components was pointed out by Flynn and Brooks [6], as a way for reducing the large and close dependence of the components within the subsumption architecture. Paradoxically, this was considered more a problem than an advantage by Hartley [8], as *similar states of the world could mean different things depending on the context*. Thus, *this would result in a behavior being activated when another behavior accidentally allowed the world to satisfy its preconditions*. Substituting the real world by this inner representation, the problem can be minimized as symbolic information can disambiguate these situations. In fact, in Figure 1(left) the task to accomplish by all agents is clear as it is commanded by the deliberative agent.

As Fig. 1(left) depicts, the execution of the high-level action emanated from PELEA is controlled by the Executive module, a component that also provides the interface to the representation. The Executive partially assumes the functionality of the Sequencer module of classical three-layer architectures [7]. It interfaces PELEA, from which it receives the plan to execute and to which it reports changes on the representation, through asynchronous events. The Executive module publishes the representation to all agents (blank arrows in Figure 1(left)) and it is also the only module in charge of checking if the changes on the representation, coming from the agents, are valid or not. More details can be read in [10], but here this property is reflected by connecting the Executive core with a Grammar data set. Significantly, this scheme implies that the agents will execute the required subtask as they receive a direct order from the Executive. Hence, the internalized state of the world does not guide its behavior and it will be only used, as described before for the simple 'go to

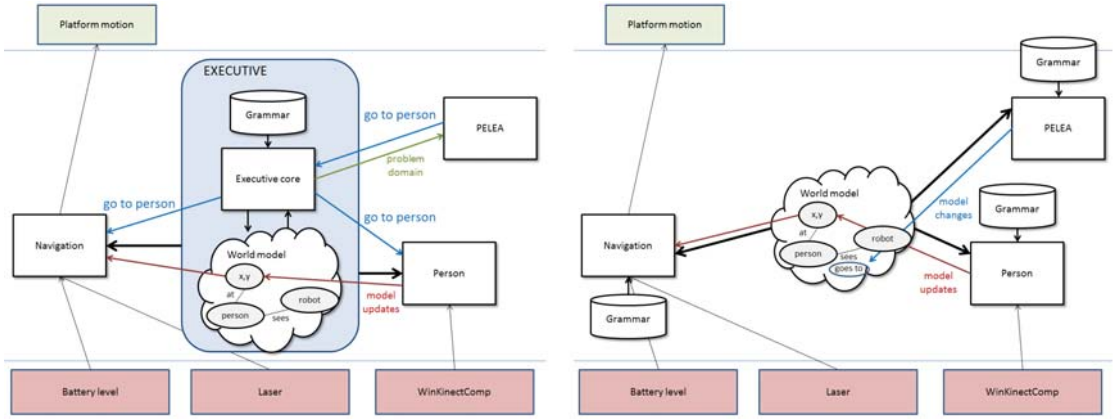


Fig. 1. (left) A brief scheme of the RoboCog architecture, showing its three main components: PELEA (a high level module for planning, monitoring, and learning), an Executive in charge of redirecting the plans from the planning module to the corresponding low-level modules and managing the representation of the world, and a set of domain-dependent agents (in this case represented by Navigation and Person). Modules with red background (Battery level, Laser...) provide inputs to the agents, meanwhile those with green background (Platform motion) receive the results from the agents. Both sets constitute the Hardware Abstraction Layer (HAL) of the system; and (right) the scheme of the new proposal for dealing with this same task. There is not an Executive module and PELEA needs to change the representation to achieve the correct response from the Person and Navigation agents.

person' example, to intercommunicate the agents. This paper proposes to change this scheme by removing the Executive module and forcing the agents to encode the action using the same set of symbols. Figure 1(right) schematizes how PELEA should ask the agents to perform the 'go to person' behavior. Briefly, as it occurs with perceptions, actions will also be thought of as changes to the world. It is in this new proposal where the shared representation truly becomes the core of the architecture, storing all data that is required for the agents to perform their activities. This simplifies the architecture as a whole, as no further modules are required to understand the state of the robot and its context. Furthermore, fewer connections eases intercommunication and generalization. As a major disadvantage, this scheme implies that the modules use a more complex logic to infer their activities from the state, without specific action commands. However, they are also more easily modified, added or removed without affecting the rest of the architecture. This approach also eases the deployment of different behavioral schemes such as stigmergic collaboration or competitive approaches (e.g. using more than one planner).

The rest of the paper is organized as follows: Section II briefly presents the unified framework for representing geometric and symbolic information. The proposal has been currently endowing within CLARC[1], a robot in charge of performing different tests to geriatric patients. Sections III and IV show the unfolding of the proposal for performing the Barthel test and the preliminary results from this work. An open discussion about the pros or cons of this new scheme is sketched at Section V.

[1]http://echord.eu/essential_grid/clark/

II. The Deep State Representation

The Deep State Representation (DSR) is a multi-labeled directed graph which holds symbolic and geometric information within the same structure. Symbolic tokens are stated as logic attributes related by predicates that, within the graph, are stored in nodes and edges respectively. Geometric information is stored as predefined object types linked by 4×4 homogeneous matrices. Again, they are respectively stored as nodes and edges of the graph. Figure 2 shows one simple example. The **person** and **robot** nodes are geometrical entities, both linked to the **room** (a specific anchor providing the origin of coordinates) by a rigid transformation. But, at the same time that we can compute the geometrical relationship between both nodes ($RT^{-1} \times RT'$), the **person** can be located (is_with) close to the **robot**. Furthermore, an agent can annotate that currently the **robot** is_not **speaking**.

A. Data structure

As a hybrid representation that stores information at both geometric and symbolic levels, the nodes of the DSR store concepts that can be symbolic, geometric or a mix of them. Metric concepts describe numeric quantities of objects in the world that can be structures like a three-dimensional mesh, scalars like the mass of a link, or lists like revision dates. Edges represent relationships among symbols. Two symbols may have several kinds of relationships but only one of them can be geometric. The geometric relationship is expressed with a fixed label called RT. This label stores the transformation matrix (expressed as a Rotation-Translation) between them.

Then, the DSR can be described as the union of two *quivers*: the one associated to the symbolic part of the representation, $\Gamma_s = (V, E_s, s_s, r_s)$, and the one related to the geometric part, $\Gamma_g = (V_g, E_g, s_g, r_g)$. A quiver is a quadruple, consisting of a set V of nodes, a set E of edges, and two maps $s, r : E \to V$.

These maps associate with each edge $e \in E$ its starting node $\mathbf{u} = s(e)$ and ending node $\mathbf{v} = r(e)$. Sometimes we denote by $e = \mathbf{uv} : \mathbf{u} \rightarrow \mathbf{v}$ an edge with $\mathbf{u} = s(e)$ and $\mathbf{v} = r(e)$. Within the DSR, both quivers will be finite, as both sets of nodes and edges are finite sets. A *path* of length m is a finite sequence $\{e_1, ...e_m\}$ of edges such that $r(e_k) = s(e_{k+1})$ for $k = 1...m-1$. A path of length $m \geq 1$ is called a *cycle* if $s(e_1)$ and $r(e_m)$ coincide.

According to its nature, the properties of the symbolic quiver Γ_s are:

1) The set of symbolic nodes V contains the geometric set V_g (i.e. $V_g \in V$)
2) Within Γ_s there are no cycles of length one. That is, there are no *loops*
3) Given a symbolic edge $e = \mathbf{uv} \in E_s$, we cannot infer the inverse $e^{-1} = \mathbf{vu}$
4) The symbolic edge $e = \mathbf{uv}$ can store multiple values

On the other hand, according to its geometric nature and the properties of the transformation matrix RT, the characteristics of the geometric quiver Γ_g are:

1) Within Γ_g there are no cycles (acyclic quiver)
2) For each pair of geometric nodes $\mathbf{u}$ and $\mathbf{v}$, the geometric edge $e = \mathbf{uv} \in E_g$ is unique
3) Any two nodes $\mathbf{u}, \mathbf{v} \in V_g$ can be connected by a unique simple path
4) For each geometric edge $e = \mathbf{uv} = RT$, we can define the inverse of e as $e^{-1} = \mathbf{vu} = RT^{-1}$

Thus, the quiver Γ_g defines a directed rooted tree or rooted tree quiver [9]. The *kinematic chain* $C(\mathbf{u}, \mathbf{v})$ is defined as the path between the nodes $\mathbf{u}$ and $\mathbf{v}$. The equivalent transformation RT of $C(\mathbf{u}, \mathbf{v})$ can be computed by multiplying all RT transformations associated to the edges on the paths from nodes $\mathbf{u}$ and $\mathbf{v}$ to their closest common ancestor $\mathbf{w}$. Note that the values from $\mathbf{u}$ to the common ancestor $\mathbf{w}$ will be obtained multiplying the inverse transformations. One example of computing a kinematic chain is shown in Figure 2.

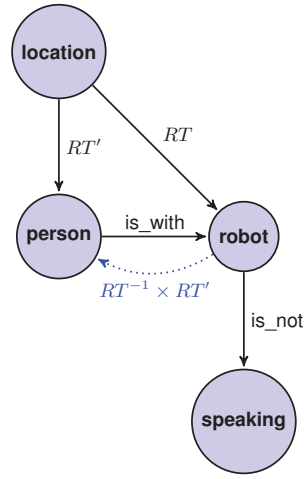


Fig. 2. Unified representation as a multi-labeled directed graph. Edges labeled as is_with and is_not denote logic predicates between nodes and they belong to Γ_s. Edges starting at **room** and ending at **person** and **robot** are geometric and they encodes a rigid transformation (RT' and RT respectively) between them. Geometric transformations can be chained or inverted to compute changes in coordinate systems.

Input: *action* from the Executive core
while *(1)* **do**
  subscribe to DSR updates;
  process { *action* };
  if *DSR_changes* **then**
    publish new DSR;
  end
end

Algorithm 1: Procedure of an agent within RoboCog

B. Internalizing the outer world within the DSR

The complexity of the domain-dependent modules typically implies that they will be internally organized as networks of software components (*compoNets*). Within each compoNet, the connection with the DSR is achieved through a specific component, the so-called *agent*. These agents are present in the two schemes drawn at Figure 1, but its degree of complexity has dramatically changed when we move from one scheme to the other. Within RoboCog (Figure 1(left)), the internal execution of an agent can be summarized by the Algorithm 1. With the removing of the Executive, the agents need to search for those changes on the DSR that launch the specific problem-solving skills of the compoNets they represent (e.g. detect the pose of a person). The new agents should then modify its internal data flow, as it is briefly outlined at Algorithm 2.

The **search_for_changes** skill depends on each agent and the behaviors that the compoNet can solve. Within the algorithm, it is stated that this function returns the *action* to perform. This is the most significant difference between Algorithms 1 and 2: in the first case the action is imposed by an external module, but in the second one, the action is determined by the agent. As we will briefly discuss at Section V this opens new ways for dealing with the top-down and bottom-up mechanisms for determining what the next action to perform will be or for implementing reflexive behaviors. The whole execution of the compoNet is conditioned by its inherent Grammar, i.e. the database storing triplets with the states of the DSR after, during and before the compoNet executes a specific action. Figure 3 shows one example, stored at the Grammar of the Speech agent (see Section III). Figure 3(left) shows the state of the DSR before PELEA states that the robot should say the sentence **yyy**. When PELEA changes the DSR (changing the attribute **xxx** to **yyy**, between **test** and **test_part**), and the agent Speech receives the new state, Speech uploads the DSR to inform all agents that the **robot** is **speaking**. When the sentence ends, Speech changes the DSR to **robot** finish **speaking**. Contrary to the way we work within RoboCog, where the Grammars were only defined by a initial state of the DSR (before an agent executes the action) and an ending state (after an agent executes the action), the agents must now to inform that they *are executing* the action. This constitutes the current way to avoid that other agent on the architecture

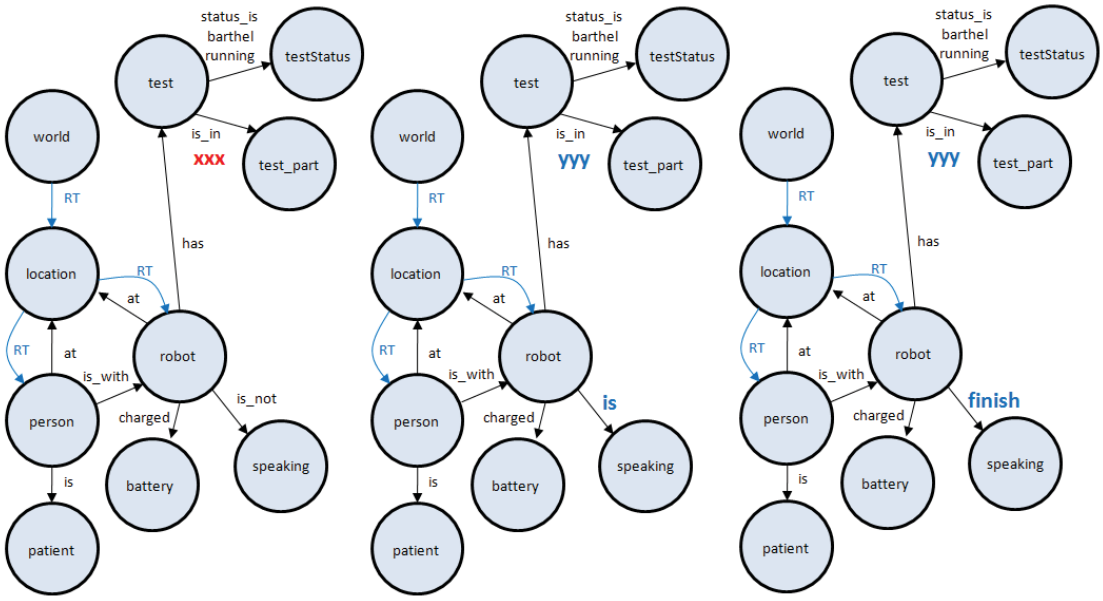


Fig. 3. (Left) The state of the DSR before PELEA states that the robot should say a sentence **yyy**; (center) PELEA changes the text to speech (from **xxx** to **yyy**) and then, when the Speech agent reads the DSR, the robot starts to speech (**robot** is **speaking**); and (right) the sentence has been said and the Speech agent informs to the rest of the agents through the DSR (**robot** finish **speaking**).

while *(1)* **do**
  subscribe to DSR updates;
  search_for_changes { output: *action* };
  process { *action* };
  if *DSR_changes* **then**
    update DSR;
  end
end

Algorithm 2: Procedure of an agent within the new proposal

modifies that part of the DSR meanwhile an action is being executed.

III. Building a whole architecture around the DSR

A. Our use case: the Barthel test

CLARC is waiting on Room 1 for its first patient. When Dr. Cesar presses the Start button on his mobile phone, CLARC wakes up and looks for Carlos, his patient, who should be sitting in front of it. When CLARC sees him, he greets him and presents itself as the responsible for conducting a small test, which will help the doctor to know how he is. It also briefly describes him what the test will be: basically a collection of questions that must be answered by selecting one of the 3-4 options described. And then the test starts. Sometimes, CLARC hear words that it does not understand. Sometimes, it just hears nothing. However, these situations are expected... and planned!. It is patient and can repeat the phrase several times, also suggest to leave it and go to the next question, and also always offer the option to answer using the touch screen on its chest. After 10-15 minutes, the test ends. It is time to say goodbye to Carlos and to send an internal message to Dr. Cesar indicating that the result of the test is stored on the CGAmed server for being validated.

This brief summary describes how the CLARC robot should address the Barthel Test. For performing the required actions, it is endowed with a software architecture that is showed at Figure 4. Out of the figure is the CGAmed server, where the application that Dr. Cesar used for launching the test is set. Once the Deliberative module (PELEA) receives this command, it wakes up the activity of the system, translates the high level action into a set of low level commands, and introduces the first of these commands within the DSR as a structural change on the DSR. Each of these changes provokes the response of one or several agents, which will try to modify the DSR towards a new state. Certain commands are single questions, that the agent of PELEA can ask examining the DSR. For instance, the command SearchPatient is answered as Yes, if the robot is seeing the patient seated in front of it, or as No, otherwise.

B. Overview of the architecture

Figure 4 shows an overview of the whole architecture in charge of performing the Barthel test within the CLARK

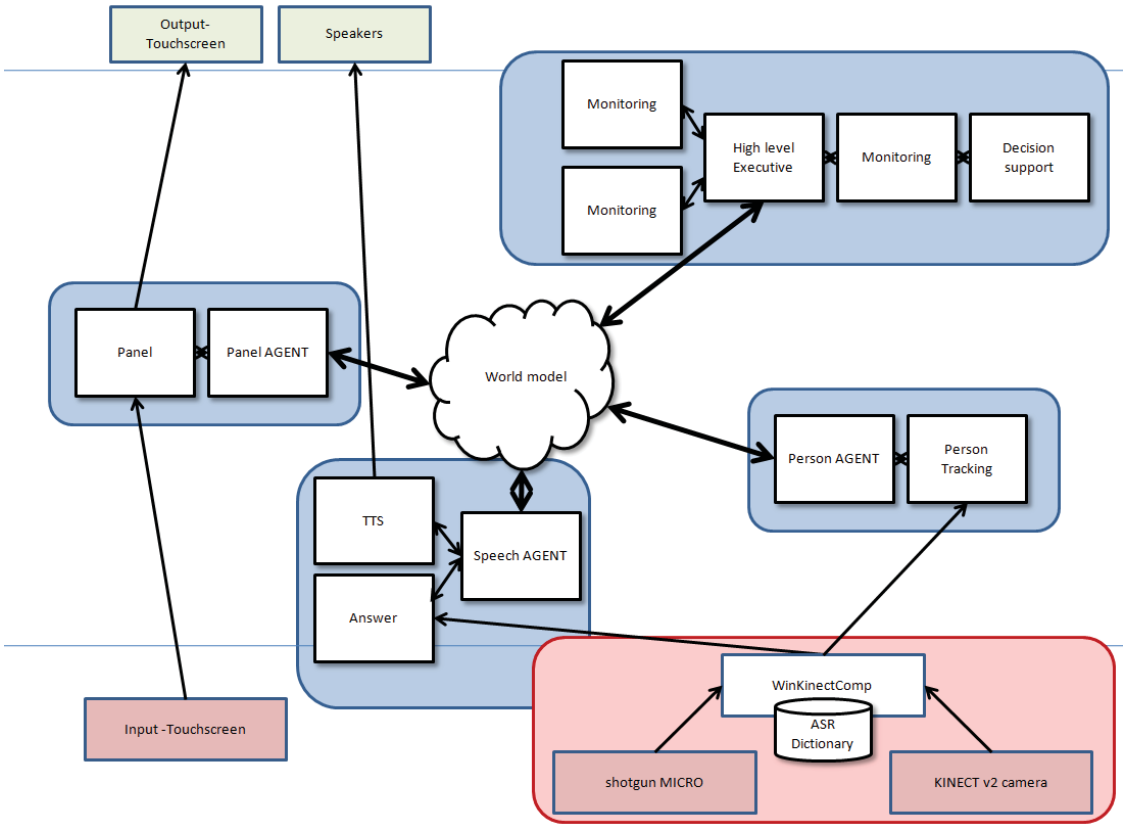


Fig. 4. Overview of the architecture within the CLARC robot. There are currently four compoNets: PELEA, Person, Panel and Speech. The Grammars that drive the behavior of these agents is encoded within the agents.

project. Surrounding the World model provided by the DSR there are four agents: PELEA, Speech, Panel and Person. The first one is in charge of providing the deliberative skills to the architecture, but it interacts with the rest of agents using the same procedure (i.e. changing the DSR). The Speech agent is the responsible of understanding the answers of the patient or guardian and of translating the text into speech, generating the voice of CLARC. This agent manages a specific grammar for dealing with the answers. The Panel agent manages the touch screen, which provides a non-verbal channel for interacting with the patient that complements the verbal one. Finally, the Person agent is the responsible of detecting and tracking the face of the interviewed person. It should be noted that, for this stage of the project, it is not needed that the robot moves. A fifth module is the WinKinectComp. It runs on a second PC and is in charge of capturing the preprocessed data provided by a rgbd Kinect sensor (i.e. joints of the person) and a shotgun microphone (automatic speech recognition). This data is provided to the Person and Speech compoNets.

PELEA is the most complex agent within the architecture. Within this project, it includes the following components

- The *High Level Executive* (HLE) module manages the whole compoNet. It receives the global goals and invokes the Monitoring module to get a plan achieving them. Then it takes the first action of the plan and invokes the HighToLow module to decompose it into low-level actions. These actions are then inserted into the DSR as changes on the model. The HLE looks at the changes in the DSR and, after a conversion to high level knowledge performed by LowToHigh, sends them to Monitoring that checks whether the plan is executing conveniently.
- The *Monitoring* module is in charge of maintaining a high level model of the environment and of invoking the Decision Support module when any deviation in the execution of the plan arises. It detects for example that the user has not answered a question or is not facing the robot and tries to find alternate plans to solve the problem found.
- The *Decision Support* module creates a plan starting from the current state, the goals to be achieved, the possible states of the world and the description of the changes the actions produce in the world state. To create the plan it

invokes an automated planner that returns the sequence of actions achieving the goals.
- The *HighToLow* module converts the high level actions of the plan created by the Decision support module into low level actions that can be included into the Inner Model.
- The *LowToHigh* module converts the information contained in the Inner Model, which represents knowledge in the form of binary predicates into n-ary predicates that the Monitoring module uses to reason about the correctness of the execution of the plan.

C. Encoding the grammar rules within the agents

With the removal of the central Executive module, we can consider that its role is now encoded within the agents on the architecture (see Figure 4). Thus, as Figure 1(right) shows, each compoNet has now its own Grammar. This Grammar is local to the compoNet and is currently encoded within the code of the agent. In an architecture that is mainly driven by how the internalized world changes, the coherence on the encoding of each change and its global synchronization are basic aspects. Although we have briefly described how the world is internalized at the DSR at Section II-B, we will provide here a more detailed description of how a Grammar is encoded within an agent.

Algorithm 3 illustrates how the specific grammar of the Speech compoNet is encoded in the agent. The Speech compoNet is in charge of translating the chosen sentence from text to speech and of receiving the responses from the patient (via the Automatic Speech Recognition (ASR) set on the WinKinectComp). There are three main modules within the Algorithm 3. The finishSpeaking is launched by the TTS module to the Speech agent to inform this that a sentence has been said. In the DSR, this implies that the **robot** is **speaking** must change to **robot** finish **speaking**. On the other hand, the setAnswer is launched by the Answer module to the agent to inform that a response has been captured. The DSR is changed from **person** waiting **answer** to **person** got **answer**. It also provides the specific $answer$. Thus, these functions encode the final state of the two rules driven the responsibilities of the Speech compoNet (e.g. Figure 3(right) shows the result of launching finishSpeaking). Within the main loop of the agent (compute), it is encoded the searching for changes aforementioned at algorithm 2. In this case, we document the two situations that launch the waiting of a new response from the patient (waitingAnswer) and the saying of a new sentence (startSpeaking). In the first case, the change on the DSR is done without evaluating any additional constraint. On the second case, we will evaluate if the $label$, i.e. the sentence to say, has been changed (see Figure 3). The procedures to address (i.e. the **process** { $action$ } of algorithm 2) are also launched (**canAnswer**() and **setText**($label$), respectively). But, as described at Section II-B, and just before launch one of these procedures, the agent notifies to the rest of agents that the process is under execution (publishing the new DSR model).

```
finishSpeaking
if getEdge(robot,speaking) == is then
    removeEdge(robot,speaking, is);
    addEdge(robot,speaking, finish);
    publishModification();
end
setAnswer
if getEdge(person,answer) == waiting then
    removeEdge(person,answer, waiting);
    addEdge(person,answer, got [answer]);
    publishModification();
end
compute
if worldChanged then
    waitingAnswer
    if getEdge(person,answer) == can then
        removeEdge(person,answer, can);
        addEdge(person,answer, waiting);
        canAnswer();
        model_modified = true;
    end
    startSpeaking
    if getEdge(test,test_part) == is_in then
        {q, label} = getEdgeAttribute(test,test_part);
        if label != label_back then
            removeEdge(robot,speaking, is_not);
            addEdge(robot,speaking, is);
            setText(label);
            model_modified = true;
        end
    end
    changeConfig
    if ... then
        ...
    end
    if model_modified then
        publishModification();
    end
end
```

Algorithm 3: Example of the Grammar encoded within the Speech agent

IV. Experimental results

The new scheme has been applied for implementing the Barthel test within the aforementioned CLARK project. The Barthel test consists of ten items that measure a person's daily functioning; specifically the activities of daily living and mobility: Bladder and bowel function, transfers and mobility, grooming and dressing, bathing and toilet use, and feeding and stairs use. Being actions and perceptions internalized within the DSR, the monitoring of the evolution of this state representation allows to track the whole execution of the use case. Fortunately, we have at our disposal in RoboComp the graphical tools for addressing this monitoring (see Figure 5).

The Barthel test needs that the robot can speech and show on a touchscreen specific sentences, for asking or helping. The robot must also hear or capture from the touchscreen the answers from the patient or relative. Finally, it is needed to track the presence of the person. The use case includes a first introduction, where the test is explained to the user, and then the robot asks the user for ten items. Each item implies that

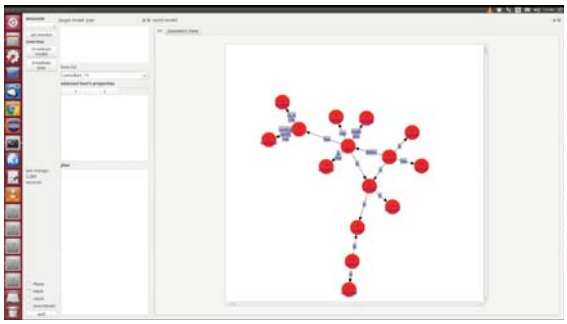


Fig. 5. Monitoring the evolution of the DSR.

the user chooses the option that best fits its state from a list. The presentation of each item follows the same scheme:

1) The robot introduces the item through voice and a message on the touchscreen
2) The robot describes each possible option (voice and message)
3) The robot asks the user to give an answer (voice or tactile on the touchscreen)
4) If after an specific time there is no response
   a) The robot describes each possible option via voice
   b) The robot shows all options on the touchscreen

If after this second try the patient does not answer, the robot will go to the next item on the test. Two non-answered items will provoke that the robot asks the clinician to attend. Figure 6 shows the evolution of the DSR during the Barthel test. It shows how the state evolves when the person is lost and then is detected again. It can be noted that the DSR practically provides a semantically annotated view of the scene. We have not measured response times but currently all the test can be run online and without remarkable latencies. The people from the Hospital Universitario Virgen del Rocío has checked the test and it is now ready for be evaluated by real users.

V. Open Discussion

It is really hard to draw a Conclusion Section as this is a preliminary work. In any case, we are currently able to run a complete Barthel test, following the correct course of actions but also acting against exogenous events such as the suddenly absence of the interviewed person (as Figure 6 shows). This allows us to open this Section and sketch what the main pros or cons of this proposal could be.

Pros. On one hand, we have been able to encode using the same scheme and collection of tokens the commands originated from a deliberative module and the perceptions, more or less elaborated by the domain-dependent modules, but always coming from the sensors. It is interesting to note that this provides a natural mechanism for merging top-down and bottom-up procedures for determining the course of action. In this proposal, the domain-dependent modules can respond in the same way to both kinds of processes. For instance, when the robot looses the person, the Speech component can immediately stop talking and PELEA can change the global course of action. The local behavior of the Speech module does not need to wait for receiving a specific command from PELEA. Thus, these modules can learn how to react to specific stimulus.

Cons. It is clear that the complexity of those software components in charge of monitoring the evolution of the DSR (our agents) is now very more complex. Although the architecture could provide the impression of being modular, this is not a reality in its current version: the definition of the agents demands a high degree of coupling among all researchers on charge of the implementation of each one of them. The reason of this dependence can be on the need of working with the same collection of symbols/tokens, which is not currently defined in advance. Thus, each change requires that all agents know how it will be 'written' on the DSR. Agents will also manage with care how they advise to the rest of agents that they are working over a specific part of the DSR. The current messages on the DSR (such as the **robot** is **speaking**) must be correctly interpreted by other agents that could be interested on using the speakers. We need again a very close interaction among the whole team of programmers.

Obviously, future work will focus on dealing with the main problems detected after these trials. We need to develop a mechanism that allows the agents to access to specific parts of the DSR (e.g. the person) and that sets and manages priority levels for using the resources. The aim should be to mimic the main guidelines of active vision perception, avoiding that other agents, with lower priority levels, can change this part of the DSR meanwhile an action is being executed. It is also mandatory to change the current way of encoding the agents, providing a mechanism that can allow a easier encoding. Finally, the collection of symbols must be *generalized*, as a way for achieving the desired modularity of the compoNets. The current encoding, which allows to *read* the state of the outer world by a simple observation of the DSR, can be a good option for achieving this generalization. However, we must also be open to the possibility that this encoding can ease our monitoring of the evolution of the DSR, but could not be the best option for achieving autonomous learning of the task-dependent modules. Other encodings (e.g. low-level features or outcomes generated/employed by these modules) could be employed at the graph items that are closer to the sensors. This will increase the data volume represented on the DSR but also open the architecture for techniques such as deep learning, which could be applied to generate the current symbols encoded using natural language. We are currently working on other tests, such as the Minimental or the GetUp & Go ones. This last one requires a high coupling of the DSR with the geometrical reality of the motion of the patient's joints during a short sequence of movements. It will be then required that the annotating of low-level features on the DSR is also increased.

Acknowledgments

This paper has been partially supported by the Spanish Ministerio de Economía y Competitividad TIN2015-65686-C5 and FEDER funds and by the FP7 EU project ECHORD++ grant 601116 (CLARK project).

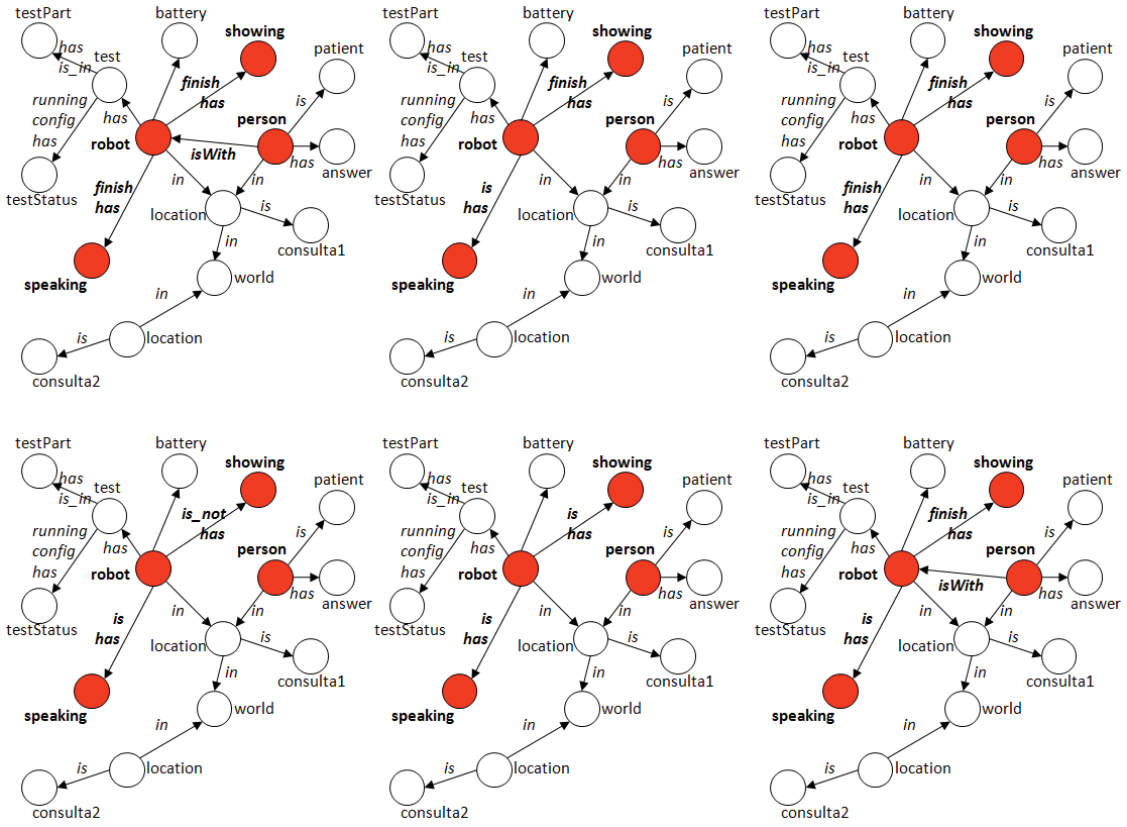


Fig. 6. Evolution of the DSR: During the execution of the Barthel test, the robot loses the person for several frames and then asks the patient to sit again in front of it. More related nodes and edges are colored and in bold, respectively. The figure shows a sequence of views of the DSR: (top-left) before losing the person, the robot has just finished on asking a question and showing a message on the touchscreen; (top-center) the **robot** is **speaking** a new sentence but then it loses the person. The situation is reported to PELEA; (top-right) but the Speech agent immediately stops speaking; (down-left) the Speech and Panel returns the DSR to its original state (**robot** is_not **speaking** and **robot** is_not **showing**) and PELEA changes the world proposing to speech a specific sentence ('Please, sit down in front of me'). This sentence is encoded as a label on the is_in attribute between **test** and **testPart**; (down-center) the Panel also shows this sentence on the touchscreen; and (down-right) the **person** isWith **robot** again and PELEA determines that the test can continue

REFERENCES

[1] V. Alcázar, C. Guzmán, D. Prior, D. Borrajo, L. Castillo and E. Onaindia, Pelea: Planning, learning and execution architecture, *PlanSIG10*, 17-24, 2010

[2] S. Blumenthal, H. Bruyninckx, W. Nowak and E. Prassler, A scene graph based shared 3d world model for robotic applications. *IEEE International Conference on Robotics and Automation*, 453-460, 2013

[3] L.V. Calderita, L.J. Manso, P. Bustos. C. Suárez-Mejías, F. Fernández, A. Bandera, THERAPIST: Towards an Autonomous Socially Interactive Robot for Motor and Neurorehabilitation Therapies for Children, *JMIR Rehabil Assist Technol.* **1**(1), DOI: 10.2196/rehab.3151, 2014

[4] L.J. Manso, P. Bustos, P. Bachiller and P. Núñez. A Perception-aware Architecture for Autonomous Robots. *International Journal of Advanced Robotic Systems* 12(174), pp.13. DOI: 10.5772/61742. 2015.

[5] S. Coradeschi, A. Loutfi and B. Wrede, A Short Review of Symbol Grounding in Robotic and Intelligent Systems, *Künstliche Intelligenz* **27** (2), 129–136, 2013

[6] A.M. Flynn and R.A. Brooks, Battling Reality, *MIT AI Memo* 1148, 1989

[7] E. Gat, On Three-Layer Architectures, *Artificial Intelligence and Mobile Robots*, 195-210, Cambridge: MIT Press, 1998

[8] R. Hartley and F. Pipitone, Experiments with the subsumption architecture. *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 1991

[9] V. Katter and N. Mahrt. Reduced representations of rooted trees. *Journal of Algebra* 412, 41-49, 2014

[10] L.J. Manso, L.V. Calderita, P. Bustos, J. García, M. Martínez, F. Fernández, A. Romero-Garcés and A. Bandera, A genera-purpose architecture for control mobile robots. *Workshop on Physical Agents (WAF)*, 105-116, 2014

[11] A. Romero-Garcés, L.V. Calderita, J. Martínez-Gómez, J.P. Bandera, R. Marfil, L.J. Manso, A. Bandera and P. Bustos, Testing a fully autonomous robotic salesman in real scenarios. *IEEE International Conference on Autonomous Robots Systems and Competitions*, 2015

The *Welcoming visitors* Task in the APEDROS Project

Daniel González-Medina, Álvaro Villena, Cristina Romero-González, Jesus Martínez-Gómez, Luis Rodríguez-Ruiz, Ismael García-Varea

***Abstract*—*Welcoming visitors* is expected to be an important task for future social robots, especially for those assisting dependent people, who may not be able to deal with simple, daily tasks. In this paper, we present a solution for the welcoming visitor task in the APEDROS research project, that it is based on the RoCKIn@Home challenge. Our solution relies on the use of a mobile robot in conjunction with a large range depth sensor, which improves the applicability of the proposal. The proposal has been empirically evaluated within a research lab where visitors may require, by means of voice commands, meeting some of the lab staff or finding objects that are suitable to be placed in the environment. The results obtained validate the effectiveness of the proposal.**

***Index Terms*—human-robot interaction, navigation, speech recognition**

I. Introduction

HUMAN-robot interaction (HRI) has become one of the most promising research fields with a wide range of real-world nowadays applications [3]. Current technologies allow robots to engage in conversation with humans while automatically perceiving the users' emotional state, recognizing their gestures, or even understanding their body language [20].

At the same time, one of the biggest challenges that society faces today is the provision of assistance to dependent people. Dependence is a phenomenon which affects all ages, not only the elderly, as many people have disabilities caused by disease, congenital defects or accidents. The impairment of cognitive and physical abilities limits a person's autonomy, and this is considered a major problem which developed countries have decided to tackle. In the case of Spain, demographic estimations indicate that, by the year 2020, there will be around 1.5 million dependent people, and this tendency will continue to rise due to its aging population [2].

The intersection of these two fields seems almost natural: social robots with enhanced HRI capabilities can assume a predominant role as caretakers of these dependent people [8]. Based on this premise, the APEDROS (*Asistencia a PErsonas con Dependencia mediante RObots Sociales*) research project aims at closing the gap between the existing technologies in social robotics and the needs of dependent people. Among others, the robot must be able to identify the key elements in its environment, recognize known people, respond to different commands, and understand natural language.

*University of Castilla-La Mancha, Spain.

E-mails: {daniel.gonzalez, alvaro.villena, cristina.rgonzalez, jesus.martinez, luis.rruiz, ismael.garcia}@uclm.es

For the evaluation of the APEDROS research project, several use cases have been designed to test the response of the robot in some common scenarios that could be found when assisting dependent people. One of them is the *Welcoming visitors* task, where the robot must identify a person in a known indoor place, and perform a different activity based on who that person is. This use case involves user detection and recognition, localization and mapping, as well as speech recognition and synthesis. It is based on the *Welcoming visitor* task of the RoCKIn@Home challenge [18], which has been adapted to better fit the evaluation process of the APEDROS project goals. Namely, the robot has to actively identify those users requiring interaction, and to translate the user commands to robot actions. These robot actions are expected to allow the user to locate and meet other people in the environment (laboratory staff), or to find some specific objects suitable for manipulation. This last feature involves a previous scene understanding stage [16] to estimate the most plausible location for the objects to be found, which has been assumed in this work.

Fig. 1 describes the overall scheme for the use case that has been used to validate the proposal. This use case has been evaluated in a real-world scenario and involving different people playing the visitor role. The mobile robot is capable of detecting visitors, interacting with them, meeting their requirements, and then coming back to its charging area. The evaluation also shows great performance in specific tasks like user classification and speech recognition.

The rest of the article is organized as follows. A brief review of the APEDROS research project and the details of the *Welcoming visitors* task are depicted in Section II. Section III describes the development of the complete use case, including the robotic platform, the modules and their role in the solution to the problem. Finally, the results obtained and the obtained conclusions are shown in Sections IV and V.

II. APEDROS Research Project

In order for a social robot to interact properly, it needs to be able to communicate with people holding high-level dialogues. To this end, certain requirements must be met. First, the robot must be able to visually track the movements of the interlocutor. Also, the robot must be able to recognize and interpret human speech, including affective speech, discrete commands and natural language [13]. In addition, the social robot must have the ability to recognize facial expressions, gestures and human actions and to interpret the social behavior

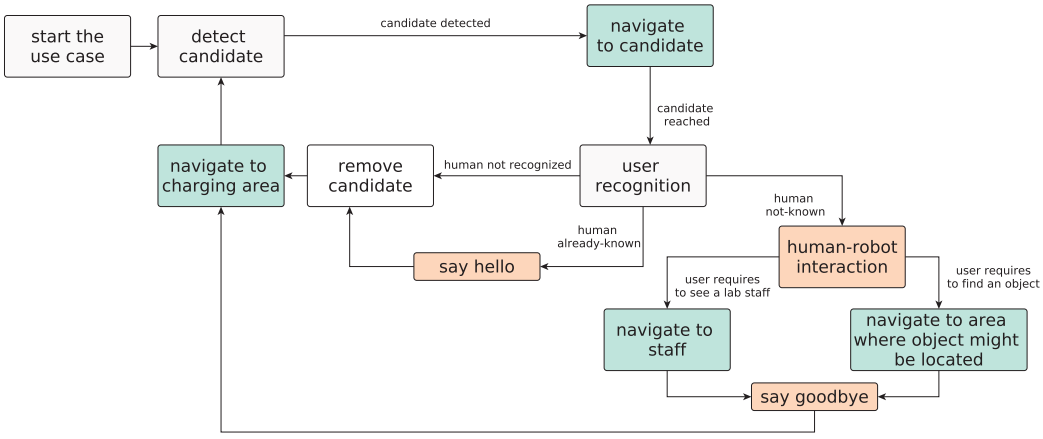


Figure 1. Overall scheme of the use case for the proposal evaluation

of people through the construction of elaborate cognitive-affective models [9].

All these multimodal HRI tasks are addressed in the APEDROS research project [10], where these are some of the key features for the development of social robots for assisting dependent people. Specifically, the main goal of this project is the integration of existing technologies with novel approaches to be developed within the project. This proposal pursues the generation of innovative and original knowledge, taking the current state of the art in this area as a starting point. The fundamental hypothesis that this project, APEDROS, puts forward is that social robotics can improve social welfare, as well as the quality of life for people with limited autonomy caused by their disabilities.

With the aim of using this technology in the most effective way, the impact of its use on the population will have to be studied and evaluated. For this reason, this project highlights three new scientific sub-objectives in the area of social and service robotics:

1) Study and development of new architectures of cognitive control and its implementation in the available hardware architecture so that they enable the robot to obtain information from the environment via its sensors, thus endowing it with autonomy.
2) Design of mechanisms for multimodal HRI, which has to be suitable for each one of the defined scenarios according to the user profile.
3) Evaluation, validation and assessment of the impact of using social robots, especially regarding the quality of life for people.

The evaluation of human-robot interaction systems in not trivial as it involves several external factors like end-user behaviors or the appearance in the environment of external agents [17]. Moreover, any real-world environment incorporates different issues related to heterogeneous points like power supplies, lighting conditions or background noise. Current related approaches include tour-guide robots [1], which may include specific modules for human detection and tracking, gesture recognition and voice recognition, or general purpose robot used to empower people with disabilities [5].

A. Welcoming visitors Task

The main goal of the proposal is to successfully meet and interact with people arriving at any kind of indoor environment by means of a mobile robot. Specifically, we setup our working scenario in the SIMD lab of the Albacete Research Institute of Informatics, University of Castilla-La Mancha, Spain. The lab consists of three different rooms, and their dimensions are 6x4.5 m, 6x2 m and 9.5x6.5 m. The overall size of the working scenario is then over 100 m^2 (an example of the robot meeting a visitor in the environment is shown in Fig. 2). This development involves: detecting people requiring interaction, going towards candidates and then, interacting with them to solve a predefined task. This task consists in taking the user to a specific place to either meet someone or reaching an object. In both cases, the robot would need to firstly find and determine the position of the person/object, but it is assumed as prior knowledge in this proposal.

Users should explicitly provide the robot with an spoken description of the task to be solved, similarly to the working scenario proposed in the RoCKIn@Home challenge[1]. In order to improve the user experience, the robot is expected to be able to recognize and find the people who works in the environment. This way, the robot can distinguish between the lab staff and visitors.

The environment is equipped with a velodyne sensor, which should be deployed for covering the maximum area. This sensor copes with the detection of potential users, in such a way that the robot does not need to continuously move to discover new users. This allows, as well, that new users can be detected even if the robot is still interacting with other users.

[1] http://rockinrobotchallenge.eu/

Figure 2. Robot welcoming a visitor within the SIMD environment.

Furthermore, the robot can initially be in any position in the environment (although it will usually be in the charging area).

III. Use Case Development

A. Robotic Platform

The robot employed is a differential-drive PeopleBot robotic platform fitted with additional sensors and computing capabilities (see Fig. 3). More specifically, the robot has been equipped with an RGB-D sensor, namely an ASUS Xtion PRO Live, and a directional microphone that improves the speech recognition performance. The initial PeopleBot platform included the navigation Hardware package, whose main device is a SICK LMS200 2D laser with an effective operation range of 10 meters. It has also been upgraded with an Intel NUC processor.

Figure 3. Robotic platform used for the experimentation.

Regarding the software architecture, we have relied on the RoboComp [15] platform, an open-source, component-oriented framework designed for robotics programming. In addition, the navigation and localization capabilities provided by the ARIA/ARNL libraries have also been used.

B. Modules

The following modules/tasks have been adopted for accomplishing the previously described task: user detection, user recognition, localization, mapping and navigation, speech recognition and generation, planning and scene understanding.

1) User detection: This module deals with detecting potential users in the environment, using the velodyne sensor as a source of information. To this end, we have developed a system that detects candidate users by following a background subtraction approach, once the static elements of the environment have been previously imaged. This is achieved from the shape and size of the acquired point clusters not belonging to structural elements in the environment, namely walls and furniture. Those clusters whose height and size match the size of a person are marked as user candidates. In order to prevent the system from continuous candidate detections which, in fact, correspond with the robot location, the system dynamically integrates the robot position into the background for its subtraction. An example of user detection is presented in Fig. 4, where we can observe how a user is successfully detected using this background subtraction approach.

2) User recognition: After detecting a potential user, the robot should firstly check that the candidate is actually a human being, and not a false positive from the user detection module. For this purpose, the robot needs to navigate to the position of the potential candidate, which involves the use of a common reference system. Candidate users are discarded if no skeleton are detected around the position of interest. This process is carried out by processing the input RGB-D image through OpenNI and NITE libraries. Once a skeleton has been detected, the user's face is found using the Viola-Jones face detector [19]. The acquired face is then classified by a Haar Cascade classifier [12] previously trained with a set of samples from the staff faces. The potential user is then classified as either a member of the staff or a visitor requiring interaction. This is conducted by processing the last n decisions of the classifier, as shown in Fig. 5. Such classifier decisions include the class and distance to the nearest staff instance in the training set. After processing twenty frames, the user is classified as the predominant class (laboratory staff) or marked as a user requiring interaction otherwise.

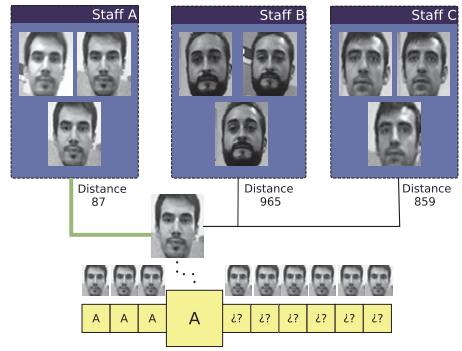
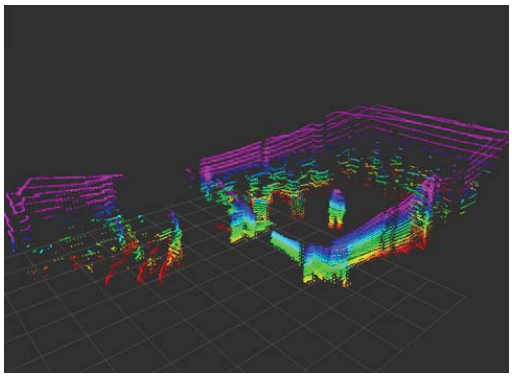
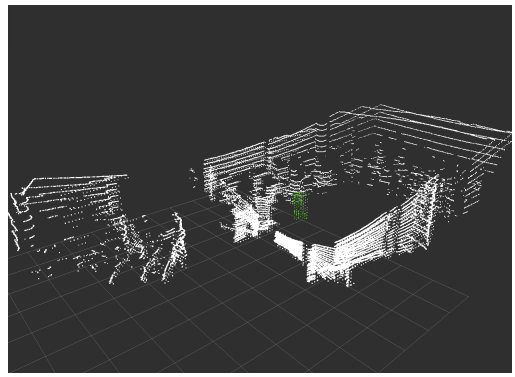


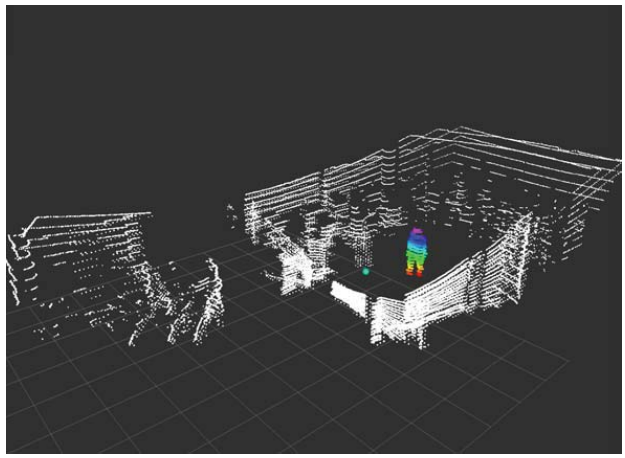
Figure 5. User recognition based on the Haar Cascade classifier.



Point cloud perceived



Background including robot occupancy (green)



People detection and robot position (green point)

Figure 4. Candidate detection process. The process starts with the acquisition of a point cloud (top-left). From this cloud we remove both the static background and the robot dynamic occupancy (top-right). The result would correspond to the candidate position of the visitor (bottom).

3) Localization, Mapping and Navigation: The localization and navigation capabilities are directly provided by the ARIA and ARNL libraries[2]. The working scenario should be previously mapped using the scanning tool provided by ARNL and the laser data of the robot, namely the SICK LMS200 device. The map is generated with the Mapper3 tool, which creates a map file given the collected laser data and some configuration parameters as, for example, the usage of raw laser data or corrected data using a loop closure technique. Once the map is available, the robot can navigate to any goal while it is self localized using the Monte Carlo method [7].

4) Speech Recognition and Generation: In order for the robot to naturally interact with the users, it has been provided with speech recognition and understanding capabilities. This recognition/understanding process is performed in two separate stages. First an automatic, continuous speech recognizer is employed to obtain a transcription to the user speech. For this task, we have relied on the *Google* speech recognition system that can accurately transcribe speech in a real environment. Once the transcription is available, the next stage deals with the interpretation of the words uttered by the user. Here, first, a parsing of the text transcription is performed in order to obtain elements like part of speech, name entities or semantic role labeling, using the SENNA [6] toolkit. Then, the output of SENNA is processed following a rule-based approach to obtain a semantic representation of the transcribed sentence, known as CFR (Command Frame Representation).

As a result of the recognition and understanding task, a suitable representation of the user requirement is obtained. This representation is then sent to the planning module. The complete pipeline for the speech recognition is shown in Fig. 6.

5) Planning: The cognitive architecture of the robot relies on RoboCog [4], which follows the Hayes-Roth guidelines [11] based on human abstraction when making plans. In this architecture, planning is performed using the active grammar model (AGM [14]) as a symbolic representation

[2]http://robots.mobilerobots.com

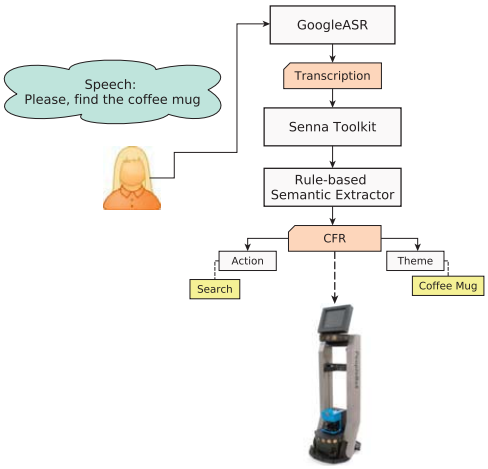


Figure 6. Speech recognition procedure.

of the environment. The geometric level models the world and their relevant elements, like other agents or objects, in the form of a graph where nodes are linked to a kinematic tree. The complete plan for the use case considers up to seventeen relevant transitions where changes are devoted to key domain elements like candidates or the robot. An example of a planning transition is shown in Fig. 7, where world representation is modified due to the detection of a candidate position. In addition to the planning, the AGM architecture is also exploited to provide the robot with prior knowledge related to the most feasible position of laboratory staff and objects.

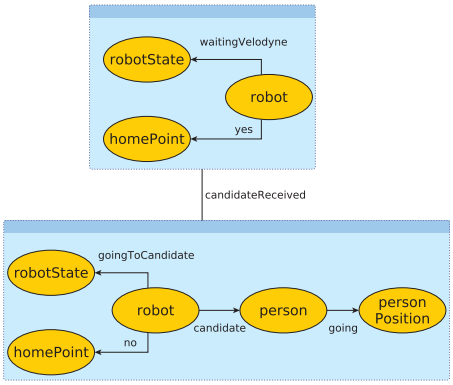


Figure 7. Snapshot of a planning transition.

IV. Experiments and Results

Experimentation on this paper was performed during several days and considering different end users to better generalize the results obtained. Specifically, the experiments consisted of twelve different trials, where a trial covered the complete use case including the different sub-stages like candidate detection, navigation and mapping or human-robot interaction among others. Nine of those trials corresponded to known staff entering the lab, and the other three where visitors that required to meet someone from the lab.

The development of the use case was based on the RoCKIn@Home scenario and involved several human agents. Concretely, each trial starts with a person coming into the lab. Next, the robot has to determine whether or not that person is a staff member. If so, the use case will be finished by greeting the person and making the robot come back to to the charging area. If the newcomer is identified as an unknown person, the robot will interact with him/her by enumerating the actions it is able to perform. In this use case, such actions are limited to take the user to the place where a specific staff member or object can be found.

Despite each experiment involved the complete development of the use case, we also focused the experimentation on the evaluation of specific tasks or capabilities. Concretely, we validated the detection accuracy from large range sensors, as well as its performance on the user recognition and dialog generation.

A. Candidate Detection

Two different parameters concerning candidate detection were evaluated: speed and precision. That is, we measured the elapsed time between the user entrance and the corresponding detection. We also computed the distance between the user position, as detected by the range sensor, and their ground truth true position. The results obtained are included in Table I and graphically presented in Fig. 8. The position of candidate users was detected a few seconds after their entrance, and the position detection was accurate enough for further human-robot interactions.

Table I
Error on user detection and elapsed time between the user entrance and their detection.

	Average	Standard Deviation
Error on the detection (mm)	656.20	190.55
Time to detect (s)	3.78	1.17

B. User Recognition

With regards to the user recognition, we decided to estimate the accuracy for those cases where the candidate was a known laboratory staff. The face recognizer is active since the candidate is labeled as a person by the skeleton tracker. This system processes every visual image acquired with the camera to produce: a.- the most probable class label for the current input, and b.- a confidence measure on the classified image obtained from the distance measurement.

To this end, we trained the user recognizer with three different staff members (same gender and age to increase challenging). From these people, we recorded and annotated 200 different face perceptions acquired under different lighting

Table II
SPEECH RECOGNITION TRANSCRIPTIONS, COMMAND REPRESENTATIONS AND ELAPSED TIME SINCE THE USER STARTS THE CONVERSATION UNTIL THE ROBOT FULLY UNDERSTANDS THE USER REQUIREMENTS.

Original sentence	Transcript	CFR	Time (s)
take me to Daniel	take me to the near take me to Daniel	TAKING(theme:"me",source:"to the near") TAKING(theme:"me",source:"to Daniel")	10.85
take me to Cristina	take me to Christina	TAKING(theme:"me",source:"to Christina")	7.19
take me to Jesus	take me to this ??? take me to Jesus	TAKING(theme:"me",source:"to this") CFR Error! TAKING(theme:"me",source:"to Jesus")	17.55
take me to Daniel	??? take me to Danny ??? take me to Daniel	CFR Error! TAKING(theme:"me",source:"to Danny") CFR Error! TAKING(theme:"me",source:"to Daniel")	15.32
take me to Cristina	??? take me to Christina	CFR Error! TAKING(theme:"me",source:"to Christina")	10.84
take me to Jesus	take me to Jesus	TAKING(theme:"me",source:"to Jesus")	7.88
		Average time (s)	11.02 $\pm$ 4.09

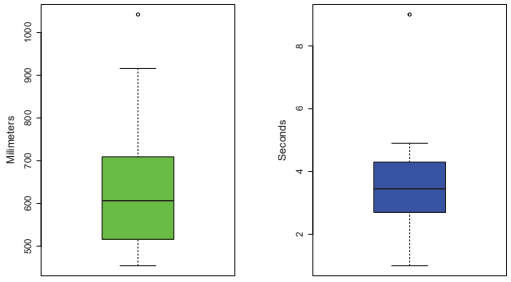


Figure 8. Error on the candidate detection (mm) and elapsed time (s).

conditions. The recording was performed while the user was 2 meters in front of the robot similar to Fig. 2. The training set then consisted of 600 different faces belonging to three classes: Staff A, Staff B and Staff C. Any single face perception was initially classified using the class of the most similar training instance. If its distance was above a pre-set threshold (250 in the experiments) the classification decision was changed to unknown visitor. A potential user was categorized as the staff member SM_i if the last 20 face perceptions included at least 15 SM_i classifications. From these data, we obtained an accuracy over 99.16 %.

C. Human-Robot Interaction

Finally, the human-robot dialog-based interaction task was also assessed. Here, the elapsed time since the user starts the conversation until the robot fully understands the user requirements was measured. This time may be adversely influenced by errors either in the transcription or understanding processes. We also recorded the original user sentence, the transcription generated and the CFR output. All these results are shown in Table II. On average, the robot needed 11 seconds to successfully understand the visitor requirements. However, the minimum average time is 7.54 s, which is a valid time for human-robot interaction. This time difference is due exclusively to errors in the transcription, while SENNA with the rule-based semantic extractor was exposed as a very useful tool for obtaining always valid CFRs.

V. CONCLUSION AND FUTURE WORK

We have presented the design and evaluation of a complete use case where a social robot welcome visitors. The use case comprises several sub-tasks like user detection and recognition, navigation, mapping, and human-robot interaction, which have been described.

In view of the results obtained, we can initially conclude that range sensors can be successfully exploited to increase the operative range of mobile robots. This has been carried out by implementing a two-stage people recognizer where a initial stage detects distant candidates. The potential candidates are then verified in a second stage involving skeleton detection from depth images. We can also conclude that the use of environment lexical annotations can facilitate the human-robot interaction, as humans may directly refer to objects or people without explicit indications about their locations. Finally, the speech recognition system shows promising results, with still some room to improve the transcription procedure.

As future work, we plan to extend the use case to dynamically discover knowledge which could be useful for visitors. In addition, a verification of the successfulness of the task in terms of checking that the user request was correctly addressed, that is the lab staff or object found actually corresponds to the user requirement.

ACKNOWLEDGMENTS

This work has been partially funded by FEDER funds and the Spanish Government (MICINN) through projects TIN2013-46638-C3-3-P, TIN2015-66972-C5-2-R, and DPI2013-40534-R and by Consejería de Educación, Cultura y Deportes of the JCCM regional government through project PPII-2014-015-P. Cristina Romero-González is funded by the MECD grant FPU12/04387.

REFERENCES

[1] V. Alvarez-Santos, R. Iglesias, X.M. Pardo, C.V. Regueiro, and A. Canedo-Rodriguez. Gesture-based interaction with voice feedback for a tour-guide robot. *Journal of Visual Communication and Image Representation*, 25(2):499 – 509, 2014.

[2] A. Börsch-Supan, M. Brandt, C. Hunkler, T. Kneip, J. Korbmacher, F. Malter, B. Schaan, S. Stuck, and S. Zuber. Data resource profile: the survey of health, ageing and retirement in europe (share). *International journal of epidemiology*, page dyt088, 2013.

[3] C. Breazeal, N. DePalma, J. Orkin, S. Chernova, and M. Jung. Crowdsourcing human-robot interaction: New methods and system evaluation in a public en-vironment. *Journal of Human-Robot Interaction*, 2(1):82–111, 2013.

[4] P. Bustos, J. Martínez-Gómez, I. García-Varea, L. Rodríguez-Ruiz, P. Bachiller, L. Calderita, L. Manso, A. Sánchez, A. Bandera, and J.P. Bandera. Multimodal interaction with loki. In *Workshop de Agentes Físicos, Madrid-Spain*, 2013.

[5] T. L. Chen, M. Ciocarlie, S. Cousins, P. M. Grice, K. Hawkins, K. Hsiao, C. C. Kemp, C. H. King, D. A. Lazewatsky, A. E. Leeper, H. Nguyen, A. Paepcke, C. Pantofaru, W. D. Smart, and L. Takayama. Robots for humanity: using assistive robotics to empower people with disabilities. *IEEE Robotics Automation Magazine*, 20(1):30–39, 2013.

[6] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November 2011.

[7] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Monte carlo localization for mobile robots. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 1322–1328. IEEE, 1999.

[8] J. Fasola and M. J. Matarić. Using socially assistive human–robot interaction to motivate physical exercise for older adults. *Proceedings of the IEEE*, 100(8):2512–2526, 2012.

[9] T. Fong, I. Nourbakhsh, and K. Dautenhahn. A survey of socially interactive robots. *Robotics and Autonomous Systems*, 42(3–4):143 – 166, 2003.

[10] I. García-Varea, A. Jiménez-Picazo, J. Martínez-Gómez, A. Revuelta-Martínez, L. Rodríguez-Ruiz, P. Bustos, and P. Nuñez. Apedros: Asistencia a personas con discapacidad mediante robots sociales. In *VI Congreso de Tecnologías de Apoyo a la Discapacidad IBERDISCAP 2011*, 2011.

[11] B. Hayes-Roth and F. Hayes-Roth. A cognitive model of planning. *Cognitive science*, 3(4):275–310, 1979.

[12] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, pages I–900–I–903 vol.1, 2002.

[13] G. Littlewort, M. Stewart Bartlett, I. R. Fasel, J. Chenu, T. Kanda, H. Ishiguro, and J. R. Movellan. Towards social robots: Automatic evaluation of human-robot interaction by face detection and expression classification. In *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada]*, pages 1563–1570, 2003.

[14] L. Manso. *Perception as Stochastic Sampling on Dynamic Graph Spaces*. PhD thesis, Cáceres Polytechnic School, University of Extremadura, 2013.

[15] L. Manso, P. Bachiller, P. Bustos, P. Núñez, R. Cintas, and L. Calderita. Robocomp: a tool-based robotics framework. In *Simulation, Modeling, and Programming for Autonomous Robots*, pages 251–262. Springer, 2010.

[16] J. C. Rangel, M. Cazorla, I. García-Varea, J. Martínez-Gómez, E. Fromont, and M. Sebban. Scene classification based on semantic labeling. *Advanced Robotics*, pages 1–12, 2016.

[17] A. Romero-Garcés, L. V. Calderita, J. Martínez-Gómez, J. P. Bandera, R. Marfil, L. J. Manso, A. Bandera, and P. Bustos. Testing a fully autonomous robotic salesman in real scenarios. In *International Conference on Autonomous Robot Systems and Competitions*, pages 124–130, 2015.

[18] S. Schneider, F. Hegger, A. Ahmad, I. Awaad, F. Amigoni, J. Berghofer, R. Bischoff, A. Bonarini, R. Dwiputra, G. Fontana, et al. The rockin@ home challenge. In *ISR/Robotik 2014; 41st International Symposium on Robotics; Proceedings of*, pages 1–7. VDE, 2014.

[19] P. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[20] Z. Zeng, M. Pantic, G. I. Roisman, and T. S. Huang. A survey of affect recognition methods: Audio, visual, and spontaneous expressions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(1):39–58, 2009.

Use and advances in the Active Grammar-based Modeling architecture

L.J. Manso, L.V. Calderita, P. Bustos, A. Bandera

***Abstract*—The choice of using a robotic architecture and one of its possible implementations is one of the most crucial design decisions when developing robots. Such decision affects the whole development process, the limitations of the robot, and changing minds can be prohibitively time consuming. This paper presents the reviewed design and the most relevant implementation issues of the Active Grammar-based Modeling architecture (AGM), as well as the latest developments thereof. AGM is flexible, modular and designed with computation distribution in mind. In addition to a continuous refactoring of the API library and planner, the most relevant improvements are an enhanced mission specification syntax, support for representations combining symbolic and metric properties, redesigned communication patterns, and extended middleware support. A few use examples are presented to demonstrate successful application of the architecture and why some of its features were needed.**

***Index Terms*—robotic architectures, artificial intelligence**

I. Introduction

ROBOTIC architectures aim to define how the different software modules of a robot should interact. They are of crucial interest because multiple properties of the robots are affected by the architecture used. Most implementations of advanced architectures provide users with reusable domain-independent modules, such as executives or planners, to avoid forcing users reinventing the wheel for every robot. These implementations are important for the roboticists because they influence the development process, the range of middleware and programming languages supported. Modularity and scalability are also affected by the concrete implementation of the architecture.

Almost all advanced autonomous robots rely on some form of a three-tiered robotics architecture (see [4]) in which one can find a reactive layer, a plan execution layer and a deliberative layer that monitors the state of the robot's internal world model to update the plan. The main differences come from implementation issues such as the middleware used, how intermediate-level modules and high-level modules such as the planner and the executive communicate, or how is the robots' internal model represented. The Active Grammar-based architecture (AGM) is no exception, proposing a detailed description of how the software modules of the robots' can interact and proposing a particular world model structure.

In particular, AGM world models are multi-graph structures with typed nodes (symbols) and edges (predicates providing relationships between symbols), where the nodes can be attributed with geometric properties. These attributes are supposed not to affect the plan, since the planner do not take them into account. Formally, these graph models can be defined as tuples $G = (V, E)$ where V is the set of nodes and E the set of edges. Nodes are tuples $n = (i_n, t_n, a_n)$, where i_n is a string identifier, t_n is represents the type of the node and a_n is a string to string mapping used to store an arbitrary number of attributes for the symbols. Edges are tuples $e = (s_e, d_e, t_e)$, where s_e and d_e are the identifiers of the source and destination nodes of the edge and t_e is a string used to define a type for the edges. This kind of graph is used to represent the robots' knowledge, to describe how the world models can change and to specify missions. See figure 2.

The architecture is built around the representation. There is a deliberative module that, based on the robot's domain and goal, proposes a plan. The rest of the architecture is composed of a pool of semi-autonomous software modules –named *agents* in this context, as in M. Minsky's *Society of Mind* [10]– which are given the plan, can read and modify the representation, and are in charge of performing a subset of actions. In the simplest scenario each action is executed by a single agent, but actions can also be executed by multiple agents in collaboration. Agents can in turn be arbitrarily modularized, controlling their own software modules. The deliberative module is also in charge of managing the representation, accepting or rejecting the update proposals made by the agents. The diagram of the current architecture is shown in figure 1.

The behavior of these agents range from blindly contributing to the execution of the first action of the plan to a pure reactive behavior[1]. Regardless of this, they can modify the model or read it in order to make the robot reach its goal. For example, the behavior of some perceptual agents can be purely reactive, behaving independently of the current plan (*e.g.*, an agent in charge of passively detecting persons). On the other hand, there can be agents in charge of modifying the model when necessary, not necessarily performing any physical action.

Domain descriptions are sets of graph-rewriting rules describing how an action or percept can modify the robots' world models. Each rule in these sets is composed of a left-hand side pattern (LHS) and a right-hand side one (RHS), and states that, starting from any valid world model, the substitution of the pattern in the LHS by the pattern in the RHS yields a valid world model. This information is used by the executive to compute the sequence of actions that constitutes each plan and to verify that the world models held by the robots are valid. See [6] for a deeper description of the used formalism.

L.J. Manso, L.V. Calderita and P. Bustos are with the Computer and Communication Technology Department of Universidad de Extremadura.
e-mail: {lmanso,lvcalderita,pbustos}@unex.es

A. Bandera is with the Electronic Technology Department of Universidad de Málaga.
e-mail: ajbandera@uma.es

[1]Agents are given the full plan, not just the first action, so they can anticipate further actions (*e.g.*, robots can lift their arms to prepare for grasping tasks when walking towards the corresponding object).

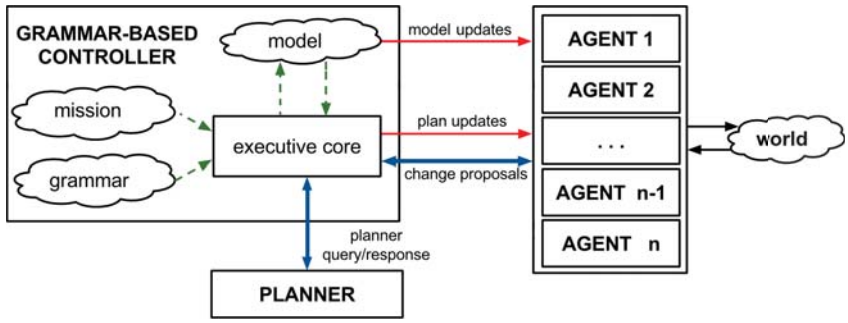


Fig. 1: Redesigned diagram of AGM. The grammar-based controller is composed of the *executive*, the *mission* specification, the *world grammar* (*i.e.*, domain description) and the *world model*. The *planner* is used by the *executive* to find plans and verify change proposals. The *agents* interact with the world perceiving or acting according to the plan, and propose to the executive model updates to acknowledge new information gathered from the environment or their actions. The executive then broadcasts to the agents those change-proposals that are found to be valid. The style of the arrow represents the nature of the information flow: dashed, thick and thin lines, mean direct access, RPC-like and publish/subscribe communication, respectively. The only modification in the diagram from the one in [8] is that change proposals are sent to the executive by RPC.

The limitations detected and the new features implemented are described in section II. To illustrate how AGM can be used, a selection of concrete examples of use is presented in section III. The conclusions and future work are presented in section IV.

II. New features

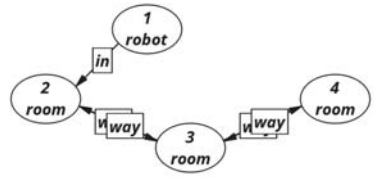
Since it was first presented in [8] the Active Grammar-based architecture (AGM) has been used in multiple scenarios and robots [11], [9], especially in the CORTEX architecture [1], which is built on top of AGM. This extensive use has raised several limitations and feature requests. This section describes how the AGM architecture has evolved over time as the result of the experience gained through its use.

A. Flexible mission definition

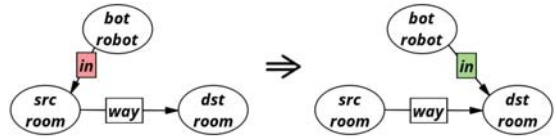
AGM targets, as proposed in [8], were defined as concrete patterns sought to be found in the robot's world model representation. These patterns were specified using a graphical model. Despite this approach is moderately easy to understand and is enough for representing most missions (*e.g.*, bringing objects, going to places, even serving coffee [7]) the missions could not work with any condition that was not explicitly represented. A simple example of this limitation can be seen in Table I, where a clean-the-table mission is specified using a) the initial approach and, b) the current approach. Using the former approach the "table clean" condition had to be explicitly marked, whereas the current approach allows avoiding such restriction.

Graphical models are easy to read and share with domain-experts and have been demonstrated to be faster and less error-prone. Increasing the expressive power of the visual language with disjunctive and quantified formulas would probably make things little less complicated than using a textual language. Therefore, it was decided to add a new optional textual section to the mission definition which is applied in conjunction with the graphic section. The syntax is similar to the one of PDDL [2].

The solution in Table I.a is clearly easier to read than the one of Table I.b by domain-experts. However, is was considered



(a) World model example describes three rooms connected in a row. The robot is in one of the extremes of the room sequence. The number in the top of each symbol is its identifier, the word in the bottom is its type.

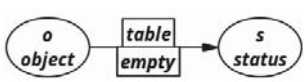


(b) Rule example (left and right hand side of the rule). It describes using a graph-grammar rule the preconditions and how the world model is modified when the robot moves from one room to another. Note that variable identifiers are used in rule definitions so that they can be used with any set of symbols satisfying the conditions.

(c) Mission example in which a robot is required to be located in the room identified with number 4. Note that mission definitions can combine concrete identifiers (with numbers) and variable ones (the robot x in this case).

Fig. 2: Examples of how the AGM architecture uses graphs to describe a) world models, b) domain descriptions, and c) missions.

a) initial proposal:



b) current solution:


```
1  (forall something:object
2     (not (in something o))
3  )
```

TABLE I: A clean-the-table mission specified using a) the initial approach, b) the current approach.

that the gap is not sufficiently big to justify a continuous monitoring and updating the world to include this kind of flags.

B. Native support for hybrid representations

The first robots using AGM used a single *world* reference frame for geometric information, so the pose of the objects could easily be specified in the attributes of their corresponding symbols[2]. However, as the architecture was being used in more robots, the need for a methodical way to represent kinematic trees with different reference frames was clear. Being able to extract this kinematic information with an easy-to-use API was also desirable.

Among all the possible approaches taken into account, the decision was to include special *transformation* edges describing the kinematic relationships. The rest of the options were discarded because they made kinematic structures harder to understand at first sight or because the impact of the model was bigger (regarding the number of additional symbols and edges to include). However, to implement these edges, which are identified with the *"RT"* label, it was necessary to enable edge attributes (a feature that the first AGM proposal missed). From a formal point of view, only the definition of edges is affected by this change. Edges are now defined as tuples $e = (s_e, d_e, t_e, a_e)$, where s_e and d_e are the source and destination nodes of the edge, respectively, t_e is a string used to define a type for the edges and a_e is a string to string map used to store an arbitrary number of attributes for the edges.

While the initial world models of the robots in AGM are defined using an XML format, their kinematic trees are usually defined using specialized file formats such as URDF [12] or InnerModel [5]. It would be time-consuming and error prone to manually include and update these kinematic structures in the robots' world models if desired. Therefore AGM provides two tools to automate the task:

- **agminner** is a tool provided to include the kinematic structure of the robot in it's internal model file.
- **libagm** is a C++ library which, among other purposes such as general access to the graph model (see section II-D), can be used to extract/update kinematics specific objects with geometry-oriented API from the model. Currently, libagm supports InnerModel kinematic definitions.

One of the advantages of using graph-like structures to represent the robots' knowledge is that they are easy to visualize. However, including their kinematic structures in the model makes these models too large to be easily visualized. To overcome this issue the AGM's model viewer was endowed with options that hide part of this information. See Figure 3.

C. Improvements on the communication strategy

Model modification proposals were initially published using one-way communication with the executive so agents did not get an answer when publishing new proposals. This made change proposals be overwritten if new proposals arrived at a fast pace. The issue has been tackled by substituting the one-way publishing mechanism which raises an exception in case the executive is busy or the modification is not correct.

Since the support for hybrid models the number of modifications of the model increased dramatically. Publishing the whole model with every change increased the cost of maintaining complex hybrid models. Maintaining human models involves updating a high number of joints and their reference frames per second. Therefore, the alternative proposed here is to allow publishing edges one at a time or –in order to decrease the network overhead– multiple edges per request.

D. libagm

Agents receive the actions they have to perform as part of the plans sent by the executive. To perform their corresponding tasks they have to access the symbols included in the current action, probably other additional symbols and, eventually modify the graph accordingly. Programmatically implementing some of these tasks is time-consuming and error-prone, so libagm has included in its API new methods to ease:

- accessing the symbols in the current action
- accessing edges given the ending symbols
- iterating over the symbols connected to a specific symbol
- publishing modified models, making transparent the middleware-dependent serialization
- printing and drawing the models

among other minor improvements and those commented in section II-B.

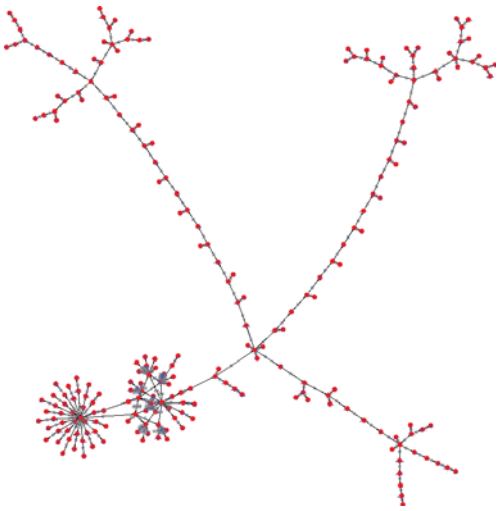
Additionally, a detailed description of the API and examples of its use have been written and published in the web[3].

E. Middleware support

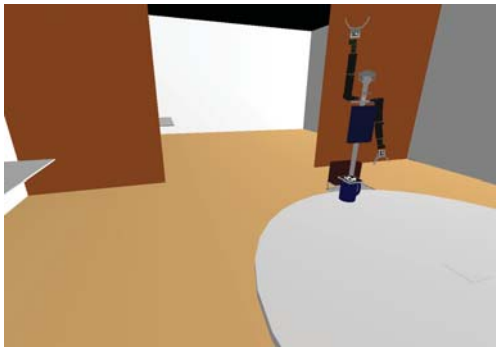
The first version of the architecture supported the RoboComp framework [3] which, despite of being a full-featured

[2]Keep in mind that these models were a directed multi-graphs, where nodes and edges were typed. Additionally, nodes could have optional attributes to store non-symbolic information which is not supposed to affect the plan but is used by some of the agents.

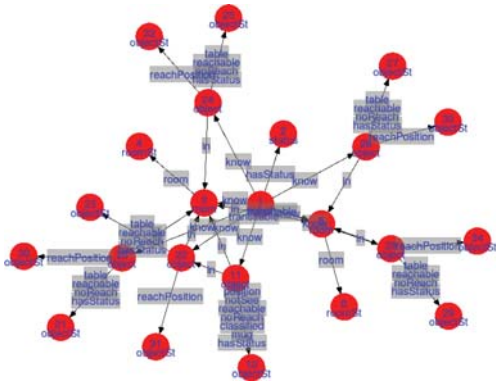
[3]http://www.grammarsandrobots.org



(a) Whole model.



(b) Geometric view of the model.



(c) Model after filtering geometry-only nodes.

Fig. 3: Example of a complex AGM model: a) as it is, b) its geometric view, c) filtering geometry-only nodes.

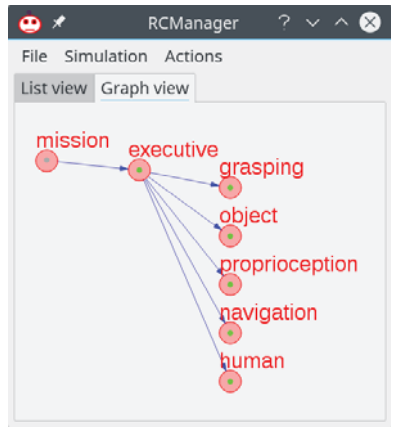


Fig. 4: Deployment network of the high level components used in these examples of use.

framework, its user base is quite limited. Learning to use a new framework or middleware is time-consuming, so roboticists may be not sufficiently motivated to change. To overcome this limitation, support for multiple framework is now underway.

The solution underway allows cooperation between agents implemented using different frameworks, specifically, RoboComp or ROS. The new executive implements services for both frameworks and, for each topic to subscribe to or publish, there is a RoboComp and a ROS version. Of course, since each of the agents can be programmed using these frameworks, they can in turn use other lower-level components implemented in the supported frameworks.

III. Use cases

Instead of describing a full-featured robot domain we will introduce several common robot tasks and how they were solved in a real robot using the new features of the architecture.

Figure 4 depicts the executive and the set of agents used in these examples along a mission viewer.

A. Changing rooms

For changing the room in which the robot is located the rule described in figure 2b was used. The robot and the rooms are explicitly represented using *robot* and *room* symbols, respectively. The current semantic location of the robot is represented using a link labeled as *in* from the robot to the corresponding room symbol. The rule depicted in figure 2b describes how, given two rooms *src* and *dst* so that the robot (*bot*) is located in *src* and there is a link labeled as *way* from *src* to *dst*, the robot can change rooms. The result of such action is that the link from *bot* to *src* is removed and a new link from *bot* to *dst* is created. Note that, as in the previous versions of AGM, removed elements are highlighted in red whereas those created as a result of the rule are highlighted in green.

From the point of view of the execution of the rule, there are two independent processes involved in the navigation agent.

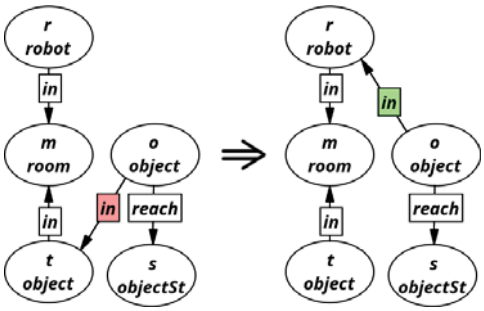


Fig. 5: Graph-grammar rule describing how object grasping affects the models.

First, when the executive requests the change room rule to be triggered, the agent sends the corresponding command to a lower level robot navigation component in charge of actually moving the platform. Concurrently, it continuously monitors the pose of the robot using another lower level localization component. The limits of the rooms are specified as an attribute of each room's symbol, so when the robot actually changes the room the *navigation* agent can update the model correspondingly, modifying the *in* edge as depicted in figure 2b.

B. Grasping

Grasping objects is slightly more complex than changing rooms. If there is a robot r and a table t containing an object o in a room m, if the o is reachable by the robot, such object can change from being in the table t to being in the robot.

For this task, we benefited from the hybrid models support combining symbols, relationships between the symbols, and geometric information. Since the object to grasp and the robot itself is represented in the model (as in figure 3a) we can extract an InnerModel object to compute the necessary geometry-related computations.

As described in section II-A, continuously monitoring the model to highlight logical conditions (based on symbols and/or edges) is not necessary anymore. However, geometric conditions are still necessary to be continuously monitored. It is the case of the *reach/noReach* edge that links each object and its status symbol. Depending on whether the robot can or cannot reach an object, the *grasping* agent, modifies the corresponding link. Only when it is reachable is that the robot can actually grasp an object. Therefore, before grasping an object an approaching action is requested by the executive if necessary (*i.e.*, if the object is not close to the area of the space where the robot can grasp objects).

If, for any reason, the robot platform starts moving, the grasping action is paused. To implement this the *grasping* agent needs to monitor the movement of the robot platform. However, access to the platform by the grasping agent is not recommended. It was already pointed out that the model holds the robot's pose as given by a localization component, however, localization may slightly vary even if the robot is still. In order to make the *grasping* agent able to monitor the raw odometry without providing it with access to the platform, the *navigation* agent is also in charge of including and updating the amount of movement of the robot in the last seconds as a "movedInLastSeconds" attribute of the *robot* symbol.

C. Human representation

As introduced in section II-C, maintaining geometric human models involves updating a high number of reference frames per second. In particular, assuming that for every human a total of 15 joints are tracked at $30Hz$, it would require 450 updates per second per person. If the agent *human*, the one in charge of this task would have to perform a remote call to the executive per joint, it would require 900 remote calls per second to track two humans in real time.

Thanks to the new interface of the AGM's executive, it can now be done with just one call. This made the frequency of the agent go from 2Hz to 30Hz. Instead of publishing the edges it would have also been possible to perform structural change proposals sending the whole world model, but this kind of behavior would slow the rest of the agents down because they would have to extract their corresponding extracted InnerModel objects much frequently.

IV. Conclusions and future work

The improvements of the AGM architecture and the software provided were presented. Section III described how can different tasks be implemented using AGM.

There are two current lines of work to improve AGM. First, support for hierarchical reasoning is underway. It will allow the planner to avoid taking details into account until necessary. Second, efforts are being made toward a decentralized representation. Currently the executive holds the *valid* world model, however, it would be interesting to distribute the issue. To this end, each agent proposing structural changes would have to perform the model-checking mechanisms that the executive currently performs.

Acknowledgment

This work has been partially supported by the Spanish Ministerio de Economa y Competitividad Project TIN2015-65686-C5-5-R, by the Extremaduran Goverment fund GR15120 "Ayudas a Grupos" and FEDER funds.

References

[1] Luis Vicente Calderita Estévez. *Deep State Representation: an unified internal representation for the robotics cognitive architecture CORTEX*. PhD thesis, Universidad de Extremadura, 2016.

[2] D. McDermott et al. PDDL: the planning domain definition language. Technical Report DCS TR 1165, Yale Center for Vision and Control, 1998.

[3] L.J. Manso et al. RoboComp: a Tool-based Robotics Framework. In *Simulation, Modeling and Programming for Autonomous Robots*, pages 251–262. Springer, 2010.

[4] E. Gat. On three-layer architectures. *Artificial intelligence and mobile robots*, pages 195–210, 1998.

[5] Marco Antonio Gutierrez Giraldo. Progress in robocomp. *Journal of Physical Agents*, 7(1):38–47, 2013.

[6] L.J. Manso. *Perception as Stochastic Sampling on Dynamic Graph Spaces", school=Escuela Politécnica de Cáceres, Universidad de Extremadura, year=2013*. PhD thesis.

[7] L.J. Manso, P. Bustos, R. Alami, G. Milliez, and P. Núñez. Planning human-robot interaction tasks using graph models. In *Proceedings of International Workshop on Recognition and Action for Scene Understanding (REACTS 2015)*, pages 15–27, 2015.

[8] L.J. Manso, P. Bustos, P. Bachiller, and P. Núñez. A perception-aware architecture for autonomous robots. *International Journal of Advanced Robotic Systems*, 12(174):13, 2015.

[9] Jesús Martınez-Gómez, Rebeca Marfil, Luis V. Calderita, Juan P. Bandera, Luis J. Manso, Antonio Bandera, Adrián Romero-Garcés, and Pablo Bustos. Toward social cognition in robotics: Extracting and internalizing meaning from perception. In *XV Workshop of Physical Agents (WAF 2014), León, Spain*, pages 93–104, 2014.

[10] Marvin Minsky. *Society of mind*. Simon and Schuster, 1988.

[11] Adrián Romero-Garcés, Luis Vicente Calderita, Jesús Martínez-Gómez, Juan Pedro Bandera, Rebeca Marfil, Luis J. Manso, Pablo Bustos, and Antonio Bandera. The cognitive architecture of a robotic salesman. *Conference of the Spanish Association for Artificial Intelligence, CAEPIA'15*, 15(6):16, 2015.

[12] Martin Theobald, Mauro Sozio, Fabian Suchanek, and Ndapandula Nakashole. Urdf: Efficient reasoning in uncertain rdf knowledge bases with soft and hard rules. *MPII20105-002, Max-Planck-Institut für Informatik*, 2010.

Strategies for Haptic-Robotic Teleoperation in Board Games: Playing checkers with Baxter

Francisco J. Rodríguez–Sedano, Gonzalo Esteban, Laura Inyesto, Pablo Blanco, Francisco J. Rodríguez–Lera

***Abstract*—Teleoperating robots is quite a common practice in fields such as surgery, defence or rescue. The main source of information in this kind of environments is the sense of sight. The user can see on a display what the robot is watching in real time, and maybe also a visual representation of the robot's surroundings. Our proposal involves the use of haptic devices to teleoperate a robot, Baxter, in order for the user to obtain haptic feedback together with visual information. As a proof of concept, the proposed environment is playing checkers. Our goal is to test if the inclusion of the sense of touch improves the user experience or not.**

***Index Terms*—Teleoperation, Baxter robot, haptics.**

I. Introduction

TELEOPERATING robots is quite a common practice in many fields such as surgery, defence or rescue [1]. The reason is simple: assisting a person to perform and accomplish complex or uncertain tasks in some environments may mean the difference between failure or success.

Enhancing the user experience is a key aspect in teleoperation. These systems use different interfaces (e.g. cameras, microphones or input devices) to provide sensory information to the operator, thus, improving the user experience. Traditionally, video feedback from an on-board or front-mounted camera is limited by technical constraints [2], [3] like a restricted field of view or poor resolution. In some scenarios, these constraints make it difficult for the operator to be aware of the robot's proximity to objects [4], causing a decrease in performance. To alleviate such limitations, at least partially, haptic cues (either by force or tactile feedback) have been shown to be useful in some applications [5], [6], [7], especially when the operator performs manipulation tasks [8], [9].

The motivation behind this paper is to test whether or not the sense of touch improves the user experience in teleoperation. To achieve this, we propose an experiment: teleoperate a robot to play a board game. The scenario will have two players and one game. One player is located at the game's place while the other is away, teleoperating a robot which is "physically" located at that same place. The teleoperation system consists of a Geomagic Touch haptic interface (that acts as a master device) and a Baxter robot (which acts as the slave device). The chosen board game is "checkers", a strategy game chosen because of its simplicity: all pieces are equal, except for their color, and they can only move diagonally forward. With this setup, the user experience is evaluated by defining and measuring some metrics with a group of expert evaluators.

The paper is organized as follows. Section II shows the evaluation to be performed on the manipulation strategies described in section III. The environment described in section IV is used for the experiment presented together with the results obtained in section V. Finally, the paper ends with our conclusions.

Robotics Group. University of León. Campus de Vegazana s/n, León 24071 (Spain)

E-mail: francisco.sedano@unileon.es, gestc@unileon.es, linyea00@estudiantes.unileon.es, pblanm02@estudiantes.unileon.es, fjrodl@unileon.es

II. Evaluation methodology

In the following sections, the environment and the experiment will be described in detail. But first of all, the main goal of this paper will be presented. Our main goal is to test whether or not the sense of touch improves the user experience in teleoperation, a task usually commanded by sight.

For evaluation of our experiment we used the Guideline for Ergonomic Haptic Interaction Design (GEHID) developed by L. M. Muñoz, P. Ponsa, and A. Casals [12]. The guide provides an approach that relates human factors to robotics technology and is based on measures that characterize the haptic interfaces, users capabilities and the objects to manipulate. We chose this guide as it is a method that aims to cover aspects of haptic interface design and human-robot interaction in order to improve the design and use of human-robot haptic interfaces in telerobotics applications.

The method to be followed for using the GEDIH guide consists in forming a focus group composed of experts and designers in order to follow these steps for every indicator defined: analyze the indicator, measure the indicator, obtain the GEDIH global evaluation index and, finally, offer improvement recommendations in the interface design. After the GEDIH validation, a users experience test can be prepared in order to measure human-robot metrics (task effectiveness, efficiency and satisfaction) [13].

As our first step, we detailed a set of selected indicators that provide a quantitative and/or qualitative measure of the information perceived by the user from the teleoperated environment. The selected indicators are the following ones:

- **ReactionForce/Moment**, this indicator measures the variation in the force or moment perceived when making contact with an object or exerting a force over it.
- **Pressure**, in this case the variation in the force perceived under contact with a surface unit is measured.
- **Rigidity**, measures the absence of displacement perceived when a force is exerted.

- **Weight/Inertia**, this indicator measures the resistance that is perceived by the user when an object is held statically or moved from one place to another.
- **Impulse/Collision**, in this case the indicator measures the variation of the linear momentum that happens when colliding with objects in the teleoperated environment.
- **Vibration**, this indicator measures the variation in the position perceived when an object is manipulated by the user.
- **Geometric Properties**, in this case, the perception of the size and shape of the manipulated objects in the teleoperated environment is needed.
- **Disposition**, it is also necessary to measure the perception of the position and orientation of objects.

These indicators have to be related to the basic tasks to be performed during the experiment in order to establish how to obtain specific results for each of them from the user experience.

III. Manipulation Strategies in Board Games

Most board games contain elements with different sizes and shapes to play: chips, dices, dice cup, pawns. When we think of a human-robot game, we have to think of two ways of playing, by using virtual and real interaction. In this research, virtual interaction is our starting point. That is, games in which a robot performs the action, but the moves are commanded by a human. We are not considering real interaction games, in which the robot takes the decisions and is able to perform the action by itself [15].

In order to detail the manipulation strategies that are going to be used in the rest of the experiment, first we will set the basic actions to be performed by the teleoperated robot. Second, we will establish the tasks involved in each of the previous actions, and the relationship between these tasks, the robot sensors, the robot actuators, the haptic device and the operator. And third, we will relate these tasks to the indicators chosen for the evaluation of the experiment.

Fist, from the point of view of the basic actions that the robot can perform, two different ways of moving the game pieces can be proposed: one is to grasp and drag the pawns/chips along the board to a target position; and the other one is to grasp and raise the piece and move it to the target position.

Second, considering just chip games, these two basic actions can be divided into the following tasks:

- Grasp and drag:
  1) Chip presence: Determine if the chip is near the manipulator jaw.
  2) Calculate start and end position: Determine the plan to move the chip to the desired position.
  3) Chip grasping. Is the task that allows the manipulator to grab an object, or to lay down the jaws on top of the chip.
  4) Chip pushing. Move a game piece by applying a force to it and move it along the board.
  5) Chip presence: Determine if the chip has reached the desired position.
  6) Chip releasing. Release the chip in the desired position.
  7) Chip presence: Determine the final chip position.
- Grasp and raise:
  1) Chip presence: Determine if the chip is near the manipulator jaw.
  2) Calculate start and end position: Determine the plan to move the chip to the desired position.
  3) Chip grasping. Is the task that allows the manipulator to grab an object.
  4) Chip raising. Move the object upwards the target place.
  5) Chip moving. Move the chip to the estimated end position.
  6) Chip releasing. Release the chip in the desired position.
  7) Chip presence: Determine final chip position.

In a virtual interaction game, tasks from 3 to 6 can be substituted by human movements. But, what happens if we want to teleoperate the robot to perform these tasks or even what happens when the six tasks are replaced by a human that teleoperates a robot? This last scenario is the one that we are going to apply in this paper. We also propose the application of haptic devices to provide more information than simple visual perception.

In this situation, we have established the following relationships between previous tasks, the robot sensors, the robot actuators, the haptic device and the operator:

1) Chip presence: Exteroception sensors
2) Chip grasping: Exteroception sensors and haptic devices
3) Chip raising/dragging: Haptic device
4) Chip moving. Exteroception sensors and haptic devices
5) Chip releasing. Exteroception sensors and haptic devices

TABLE I
Basic tasks and GEHID indicators.

Basic task	Indicator
Presence	Collision on 3D movement
Grasping	Rigidity on the the grasping tool
Push	Vibration on 3D movement
Translating	Weight on 3D movement
Assembling	Collision on 3D movement

The third step is to establish a clear relationship between selected indicators and basic haptic tasks. Table I shows this relationship. The first column shows the basic tasks involved in our experiment and the specific indicators related to each of them are placed in the second column. But we have also considered for the design of our haptic interface that different tasks have different requirements, both the haptic device and the robot. For example, in the moving task, we need the game piece weight perception for the haptic force sensor, and also the grasping force provided by the corresponding robot sensor. Moreover, in our experiment, we get visual feedback provided by two cameras in the robot to determine the position of the game pieces so the operator can determine the necessary directions and trajectories to perform the desired task.

Figure 1 shows a summary of this whole process.

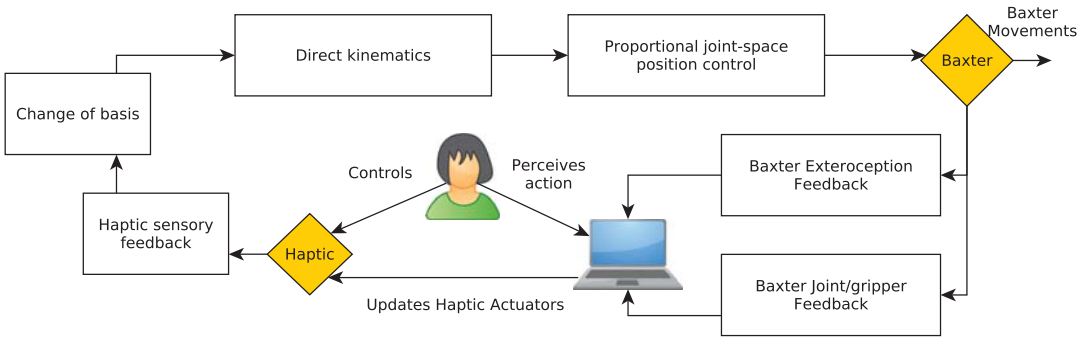


Fig. 1. Haptic-Baxter-Haptic control with human interaction.

IV. Environment description

Any telemanipulation system is comprised of two robots, master and slave, and a communication channel that physically links them. The master allows an operator to send commands to control the slave's motion, while the slave robot interacts with the environment by executing such commands. Since the goal of this paper is to test if haptic feedback improves the user experience, the master robot must be a haptic device. To play a board game, the slave robot must be placed in a fixed location and be able to grasp and manipulate objects with precision. Our system is comprised of a Geomagic Touch haptic device and a Baxter robot acting as master and slave, respectively.

To perform the experiment, the environment needs to be placed where a board game can be played. The game mechanics must be as simple as possible, so that the user focuses on the manipulation aspects, i.e. grasping the game pieces and moving them along the board. Because Baxter needs a large workspace, the game board needs to be big enough; as well as the game pieces, in order to be easily grasped by the robot's end effector. After considering several options checker's game was selected.

The chosen environment is a room containing two separate tables: one for the Checkers board and one for the operator. The game's table contains the Checkers board and pieces. The operator's table contains the master device and a workstation. To ignore the communication channel time delays, both master and slave are connected to the workstation through Ethernet cables.

A. Master: Geomagic Touch

The master device needs to be able to send the operator's movements and commands to Baxter, but at the same time, provide some kinesthetic information to the operator, i.e. reflect the forces or vibrations sensed by Baxter. In the experiment's context, haptic devices capable of force feedback perfectly fits as a master device; particularly, we used a Geomagic Touch haptic device.

The Geomagic Touch is a 6 DOF device, that has 3 actuated DOF associated to the armature which provides the translational movements (X, Y, and Z Cartesian coordinates) and other 3 non-actuated DOF associated to the gimball that gives the orientation (pitch, roll and yaw rotational movements). Also, the device is able to perform a force up to a maximum of 3.3 N within a workspace of 160 (width) $\times$ 120 (height) $\times$ 70 (depth) mm. These characteristics are more than enough to both, translate the operator's movement using the device's stylus and feel the environment perceived by Baxter using the device's actuators.

The device is connected to a workstation running Ubuntu 14.04 (Trusty) with ROS Indigo. To communicate the haptic device with ROS, we used the package "*phantom-omni*" developed by Suarez-Ruiz [10]. However, this package was developed for ROS Hydro, so we adapted it and developed a new teleoperator program to control Baxter.

B. Slave: Baxter

The slave device needs to mimic the operator's movement and be able to grab or translate the Checker's pieces. For the experiment we used a Baxter robot (see figure 2).

Baxter is an industrial robot produced by Rethink Robotics. It was developed to enable collaborative human-robot co-work and enhanced Human-Robot Interaction. It consists of two seven degree-of-freedom arms, which provide kinematic redundancy that allows to enhance object manipulation. It comes with Series Elastic Actuators at each joint, incorporating full position and force sensing. Attending specification its max payload is 2.2 kg (including end effector) and a gripping force of 35 N. Baxter has several mounted sensors, one camera on each griper along with an infrared sensor range (4–40 cm), one camera on top of the robot display which is situated as a head and range finding sensors integrated on top of the robot. Baxter deploys an intuitive Zero Force Gravity Compensation mode that allows users to move each degree of freedom of both arms without effort.

The robot is connected to the same workstation as the Geomagic Touch. To communicate with Baxter, we used the ROS package "*baxter_pykdl*" from [11] which supports Indigo.

Fig. 2. Baxter grabbing a Checkers piece from the board.

C. Controller

To use the haptic device for the six DOF robot, the robot was split into two sets of three degrees of freedom. The first system is determined by the Cartesian coordinates of the wrist joint. The second one, the rotation of the gimbal, is then used to correspond to the three rotational degrees of freedom (Rx, Ry, and Rz) of the forearm of the robot. By doing this, the problems associated with multiple solutions to larger order degree of freedom robotic systems are mitigated.

We have used as an approximation the fact that the three axes intersect at the wrist despite the length of the link between the lower arm and forearm. This is justified due to the relatively short length of this link, and the fact that the robot is controlled based on the vision of the operator, allowing for intuitive compensation of the operator to position the end effector. This visual compensation is also used to mitigate the effects of motor backlash propagation through the robot, although other solutions to reduce backlash are being attempted.

1) Movement mapping: In order to interact with the environment, the physical movement of the haptic device must be properly translated to Baxter. For this work, only one haptic device is used, so the movement will only be mapped to one arm.

Position and orientation represent, at any time, the movements of an object in a space. Robotic mechanisms use the kinematic of the joints to represent their geometry, and thus, their workspace and movements. Both Geomagic Touch and Baxter have different joints (see figure 3). So, to link the movement from one to another, first we need to map their joints properly. However, Baxter's arm has more joints than Geomagic Touch, so some of its joints must be ignored to reduce the complexity of the movement translation.

Fig. 3. Corresponding joints of Baxter (top) and the Geomagic Touch (bottom).

Table II shows the joint mapping established between both robots. The whole arm movement is achieved as follows: to move in the X-axis, the "waist" joint is used; Y-axis is achieved by moving the "shoulder" joint; and Z-axis is moved by the "elbow" joint. However, these mappings are not enough, because for some cases, only half of the arm needs to move in the Y-axis, for instance, when trying to pick or drop a Checkers chip from the board. We solved this issue by using the "wrist2" joint.

TABLE II
JOINT MAPPING BETWEEN BAXTER AND THE GEOMAGIC TOUCH.

Baxter	Geomagic Touch
S0	waist
S1	shoulder
E0	Not used
E1	elbow
W0	Not used
W1	wrist2
W2	Not used

Even so, this mapping translates the movement, it is not correctly scaled as both robots use different workspaces, i.e. a large move for the Geomagic Touch is translated as a small move in Baxter. To fix this problem, we need to represent the haptic movement and orientation inside Baxter's workspace.

This is achieved by computing the proper transformation matrices for each joint. To bound the movements between the two workspaces, the transformation matrices will only use the scaling part, thus simplifying the calculations. Because each joint moves along a single axis, the problem is reduced to compute a scaling factor for each one (see table III),

TABLE III
SCALING FACTORS FOR THE MOVEMENT OF EACH JOINT MAPPING.

	Workspace bounds		
Joint Mapping	**Baxter**	**Geomagic Touch**	**Scaling factor**
S0→"waist"	3.2	1.96	1.63
S1→"shoulder"	2.24	1.75	1.28
E1→"elbow"	2.63	2.1	1.25
W1→"wrist2"	3.66	4.63	0.79

V. EXPERIMENT AND RESULTS

In this section we present the experimental procedure to analyse operator skills to move the game chip from one position to another one. We plan to play checkers in a robot - human game, but using teleoperation to control the robot.

A. Game description

Checkers is a strategy game board for two players that play on opposite sides of board. It can be played on 8x8, 10x10 or 12x12 checker boards. There are two kind of pieces: the dark pieces and the light pieces. The game consists in moving a piece diagonally to an adjacent unoccupied square. If the adjacent square contains an opponent's piece, and the square immediately beyond it is vacant, the piece may be captured and taken away from the board by jumping over it. During the game, players alternate turns.

B. Board and game pieces adaptation

To accomplish our experiment, we modified the game board, game pieces and the gripper system of our robot. Baxter has two electric parallel grippers that allow our robot to pick up rigid and semi-rigid objects of many shapes and sizes. To play checkers we need these devices to be able to grab and move game pieces.

In real board games the shape of these pieces is cylindrical and they have a height of 3 or 4 and less of 25 mm of diameter. This is a big handicap because Baxter accuracy is only 5 mm.

For this reason, this first approach of the game proposes using 3D printer pieces. We have designed and fabricated cylindrical pieces of ten millimetres of height and 45 mm of diameter. Figure 4 shows the two elements manufactured by a 3D printer.

The chip has a slit in it to simplify the process of grasping, instead of pressing the piece from its external shape, the jaws are inserted into the slit and the gripper is opened instead of closed.

Finally, we have also modified the board and we made our own game board where the squares have a size of 55x55 mm, to adapt it to the pieces' size.

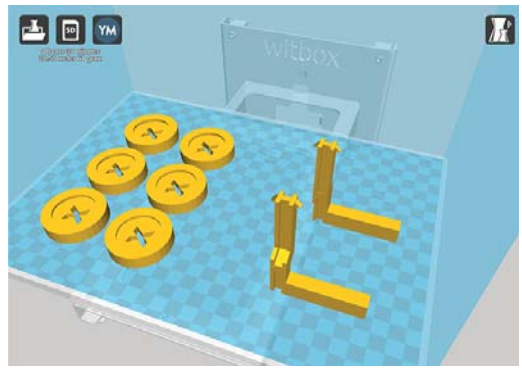


Fig. 4. Games pieces and grippers for print in 3D printer.

C. Experimental Workspace

An experimental workspace is defined in order to evaluate the users skills. The checkers game consists in repeating several times the actions of grasping a chip and moving it to another place on the board. Thus, we decided to generalize this task and perform an evaluation with a small group of users in order to understand their perception.

In this case, the workspace is a virtual orthogonal polyhedron situated in front of Baxter. The dimensions are a length of 75 cm and a width of 30 cm. The workspace is composed by two foam elements separated 15 cm from each other. It has two lateral panels to avoid visual contact from the operator. These elements are used to measure the collisions with the robot arm during the teleoperation task.

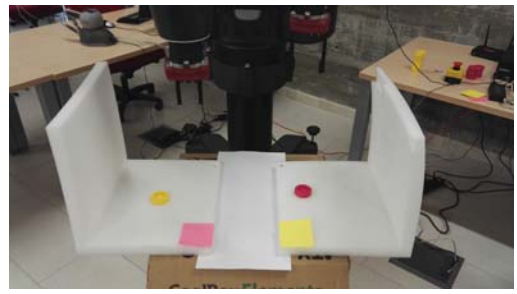


Fig. 5. Experimental workspace designed for the evaluation test.

Fig. 5 shows the workspace used during the experimental test. The user selects chip color before the experiment (yellow or pink) and she has to move one chip to the side marked with the same color.

The subjects of the experiment receive two sources of information: the haptic interface and the information from robot cameras. In this case, there was also an extra camera on top of the robot's head for perceiving Baxter's arm movements. Fig. 6 presents the information presented to each subject. Top-left image on the figure presents the camera available on Baxter's wrist. Top-right image on the figure shows the workspace from Baxter's head. IT provides a wide-view angle

of the workspace. The terminal available on the bottom offers the distance from the arm wrist to the table.

D. Evaluation metrics

From the point of view of the evaluation, the concepts of usability proposed in ISO 9241-11 [14] are the ones considered: effectiveness, efficiency and satisfaction. The effectiveness will evaluate if all subjects are able to accomplish the task with the expected result. The efficiency will evaluate if the subjects accomplish the task with the least waste of time and effort. In this case we are going to measure the time and the collisions with the lateral planners. Finally, satisfaction value has to consider both robot and haptic devices. In that manner, we measure aspects like collaborative metrics [12] in order to analyse human-robot interaction and human-haptic interaction. This feedback is given by a questionnaire performed by all the subjects of the experiment.

E. Evaluation results

A group of nine subjects was chosen. They were familiar with the use of haptic devices: seven subjects were technical people and two subjects were the developers of the application. Table V-E shows the quantitative results. There is a difference of 37 seconds between the fastest technical operator and the fastest developer.

TABLE IV
RESULTS OF THE EXPERIMENT

Subject	Time (sec.)	Collisions	Specification
1	57	2	Technician
2	60	0	Technician
3	161	3	Technician
4	50	0	Technician
5	70	0	Technician
6	55	0	Technician
7	90	0	Technician
8	23	0	Developer
9	13	0	Developer

Analysing these results we can see that the developers of the experiment have completed the task much quicker than the other participants. Thus, taking only data from the first seven subjects in the experiment, we can see that the average time to complete the task has been 77,5714 seconds and a standard deviation of 39,1256 seconds (see table V).

TABLE V
DESCRIPTIVE STATISTICS FROM HUMAN-ROBOT EXPERIMENTAL RESULTS

Description	Result
Subjects	7
Mean	77,5329s
Standard Deviation	39,1256s
Minimum Time	50s
Maximum Time	161s

F. Questionnaires

Finally, eight of the participants replied to a set of questions for a subjective evaluation of our research. Table VI presents the results.

The results were positive except on the question, *How natural did your interactions with the haptic device seem?* where the users expectations are different in terms of normality against other devices as a video game pad.

In their comments, the subjects asked to add more force feedback in the haptic device and more information about the distance from the gripper to the board game on the table.

VI. CONCLUSION

This papers presents an experiment for teleoperating a Baxter robot by means of a Geomagic Touch haptic device. The task to be performed is playing Checkers, and the goal is to know if getting haptic feedback improves the user experience.

The designed environment includes the haptic device in the master role and Baxter in the slave role. An adapted board and chips were used for nine users to perform two basic tasks: dragging and raising a chip. Besides the metrics used for the quantitative evaluation, a questionnaire was also used for the qualitative evaluation.

The results obtained by using the GEHID guide, show that the users that are new to the system, that is, the ones that are not the developers, do not find the interaction to be natural. They also suggest adding more force feedback in the haptic device when the robot is dragging or holding a chip. They propose including more accurate information about the distance from the gripper to the board game.

In future work we will try to improve the correspondence between the haptic device and Baxter's arm movements in order to make them more natural. We will test several force feedback information and improve the operator's interface for offering more precise information. We are also considering comparing this environment to a parallel one using a leap motion device for testing if the sense of touch improves the user experience. To do so, we need to compare an environment guided by haptic response to another one based only on graphical information.

ACKNOWLEDGMENT

This work is supported by the *Cátedra Telefónica–Universidad de León* and it has been partially funded by Spanish *Ministerio de Economía y Competitividad* under grant DPI2013-40534-R.

REFERENCES

[1] J. Vertut and P. Coiffet, *Teleoperation and Robotics. Applications and Technology*, Spring Netherlands. Vol. 3B, 1985.

[2] D.D. Woods, J. Tittle, M. Feil and A. Roesler, *Envisioning human-robot coordination in future operations*, IEEE Transactions on Systems, Man and Cybernetics: Part C (Applications and Reviews). Vol. 34, no.2, pp. 210–218, 2004.

[3] D. Woods and J. Watts, *How not to have to navigate through too many displays*, In Handbook of Human-Computer Interaction.estricting the field of view: Perceptual and performance effects. 2nd ed, M. Helander, T. Landauer and P .Prabhu, Eds. Amsterdam, The Netherlands: Elsevier Science, pp. 1177–1201, 1997.

[4] P.L. Alfano and G. F. Michel, *Restricting the field of view: Perceptual and performance effects*, Perceptual and Motor Skills. Vol. 70, no. 1, pp. 35–45, 1990.

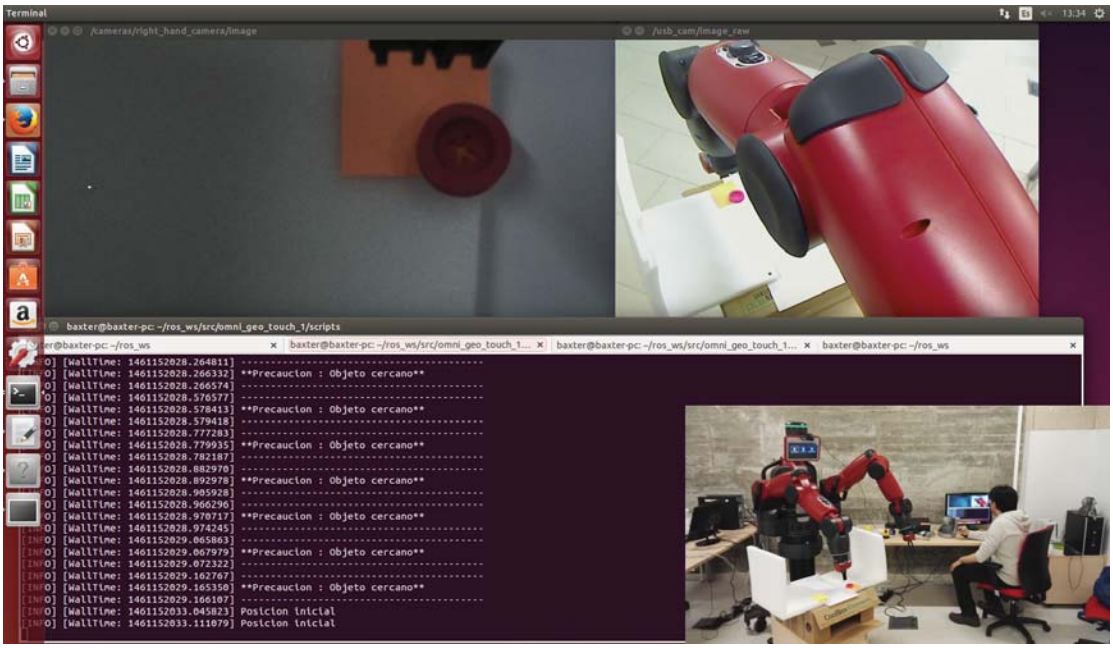


Fig. 6. Information presented to each subject during the test.

TABLE VI
RESULTS OF A QUESTIONNAIRE USING A LIKERT SCALE:1 NOT AT ALL - 7 EXTREMELY

Questions	**Average**
How much were you able to control arm movements with the haptic device?	4
How responsive was the robot to actions that you initiated (or performed) with the haptic device?	4.25
How natural did your interactions with the haptic device seem?	3.375
How much did the visual perception of the environment help you to perform the task of grasping the chip?	4.875
How compelling was your haptic sense while the robot was grasping the chip?	4.857
Were you able to anticipate what would happen next in response to the actions that you performed?	4.375
Before the experiment, do you feel the experiment like a waste of time?	1.5
After the experiment, do you feel the experiment like a waste of time?	1.125
What type of board game would you like to play with a robot?	checkers (3), Chess (3), Other (1)
In which role do you like to play?	remote operator (3), opponent (4), both (1)

[5] H.I. Son, A. Franchi, L.L. Chuang, J. Kim, H.H. Bulthoff and P.R. Giordano, *Human-Centered Design and Evaluation of Haptic Cueing for Teleoperation of Multiple Mobile Robots*, IEEE Transactions on Systems, Man and Cybernetics: Part B (Cybernetics). Vol. 43, no. 2, pp. 597–609, 2013.

[6] M. Sitti and H. Hashimoto, *Teleoperated Touch Feedback From the Surfaces at the Nanoscale: Modeling and Experiments*, IEEE ASME Transactions on Mechatronics. Vol. 8, no. 2, pp. 287–298, 2003.

[7] N. Diolaiti and C. Melchiorri, *Tele-Operation of a Mobile Robot Through Haptic Feedback*, In Proceedings of the IEEE International Workshop on Haptic Virtual Environments and Their Applications (HAVE 2002). , Ottawa, Ontario, Canada, 17-18 November, 2002.

[8] C.H. King, M.O. Culjat, M.L. Franco, C.E. Lewis, E.P. Dutson, W.S. Grundfest and J.W. Bisley, *Tactile Feedback Induces Reduced Grasping Force in Robot-Assisted Surgery*, IEEE Transactions on Haptics. Vol. 2, no. 2, pp. 103–110, 2009.

[9] A. Kron, G. Schmidt, B. Petzold, M.I. Zah, P. Hinterseer and E. Steinbach, *Disposal of explosive ordnances by use of a bimanual haptic telepresence system*, In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 04). pp. 1968–1973, 2004.

[10] F. Suárez-Ruiz, *ROS Packages for Sensable Phantom Omni device*, GitHub, 2014. [Online]. Available: https://github.com/fsuarez6/phantom_omni. [Accessed: 06–Apr–2016]

[11] Rethink Robotics, *Baxter PyKDL*, GitHub, 2014. [Online]. Available: https://github.com/RethinkRobotics/baxter_pykdl [Accessed: 06–Apr–2016]

[12] L.M. Muñoz, P. Ponsa and A. Casals, *Design and Development of a Guideline for Ergonomic Haptic Interaction*, In Human-Computer Systems Interaction: Backgrounds and Applications 2: Part 2, Z. Hippe and J.L. Kulikowski and T. Mroczek, Eds. Springer Berlin Heidelberg, pp. 15–19, 2012.

[13] B. Andonovski, P. Ponsa and A. Casals, *Towards the development of a haptics guideline in human-robot systems*, Human System Interactions (HSI), 2010 3rd Conference on, Rzeszow, pp. 380–387, 2010.

[14] ISO, *ISO 9241-11:1998 Ergonomic requirements for office work with visual display terminals (VDTs) part 11: Guidance on usability*, 1998.

[15] Barbu, Andrei and Narayanaswamy, Siddharth and Siskind, Jeffrey Mark, *Learning Physically-Instantiated Game Play Through Visual Observation*. Robotics and Automation (ICRA), 2010 IEEE International Conference on, 1879–1886, 2010, IEEE

Human Body Feature Detection and Classification for Rehabilitation Therapies with Robots

Eva Mogena, Pedro Núñez and José Luis González

***Abstract*—Current modern society is characterized by an increasing level of elderly population. This population group is usually ligated to important physical and cognitive impairments, which implies that older people need the care, attention and supervision by health professionals. In this paper, a new system for supervising rehabilitation therapies using autonomous robots for elderly is presented. The therapy explained in this work is a modified version of the classical 'Simon Says' game, where a robot executes a list of motions and gestures that the human has to repeat each time with a more level of difficulty. The success of this therapy from the point of view of the software is to provide from an algorithm that detect and classified the gestures that the human is imitating. The algorithm proposed in this paper is based on the analysis of sequences of images acquired by a low cost RGB-D sensor. A set of human body features is detected and characterized during the motion, allowing the robot to classify the different gestures. Experimental results demonstrate the robustness and accuracy of the detection and classification method, which is crucial for the development of the therapy.**

***Index Terms*—Robotics in rehabilitation, RGBD image analysis, SVM, Decision tree, KNN**

I. Introduction

THERE is a huge demographic change in the current modern society, which is characterized by a significant and continuous increase of the elderly population[1]. In Spain, for instance, the last report of Spanish Statistics Institute (INE) shows as elderly population has been on an upward trend over years [1] (see Fig. 1). Similar reports have been published in most industrial countries. These reports also conclude that regular physical and cognitive exercises adapted for this population group are associated with a lower risk of mortality and morbidity, and with a better quality of life. In fact, to improve their quality of life, Information and Communications Technologies (ICT) are an excellent tool, which have already made significant progresses in assisting people with different needs either in hospital or in the patient's home. The development of physical and/or cognitive therapies supported by these new technologies and supervised by professionals have increased in the last decade. The use of robots in these therapies, for instance, has several interesting adventages, such as the ability to record the exercise using its sensors for a later analysis, $e.g.$ cameras or microphone, the possibility to adapt itself to changes in the environment or in the elderly during the session or to provide a resource to occupational therapists

Pedro Núñez and Eva Mogena are with University of Extremadura. E-mail: emogenac@alumnos.unex.es

José Luis González is with Fundación CénitS-ComputaEX.

[1]World Heath Organization (WHO) defines "Elderly people" as those individuals who are aged 60 years or older [2]

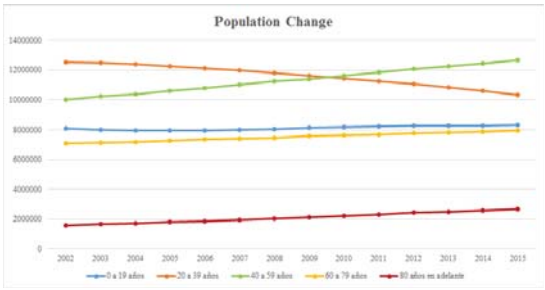


Fig. 1. Demographinc evolution for Spain. Source: INE report

that allows them to capture the attention of their patients with a more motivating therapy.

Current therapies for elderly population are focused on regular physical exercises adapted for them and supervised by a human professional. Most of these exercises are based on imitating several motions that use different joitns ($e.g.$, to move an arm making an arc or to walk straight a distance). Other therapies focused on cognitive exercises are based on mental activities ($e.g.$, memory, judgement, abstract reasoning, concentration, attention and praxis) where the elderly interacts with the environment and the professional [3]. Therapies that combines both the physical development of the patient and memory exercises are also common in the literature. Making such therapies with a specific robot, either replacing or accompanying the human therapist, offers seniors an alternative to conventional therapy sessions.

The main goal of this paper is the development of a system that allows to play the classical 'Simon Says' game with senior patients, in order to perform a physical and cognitive therapy supervised by an autonomous robot. In this novel therapy, the therapist robot executes a list of motions and gestures that the human has to repeat each time with a more level of difficulty. The described therapy help work both the physical condition of patients, $i.e.$ by imitating the movements, and their memory, $i.e.$ by remembering the sequence of movements. The robot is equipped with a low cost RGB-D camera, and the sequence of images is analyzed for extracting and characterizing a set of human body features. These features are the input of a second stage, where different classificators are evaluated.

This article is structured as follows: Section II provides a brief summary about similar works in this field of research. In Section III, a description of 'Simon says" game used

as threrapy is made. Section IV provides the description of the proposed approach. Experimental results are detailed in Section V. Finally, the main conclusions of this paper is presented in Section VI.

II. Related Work

In the last decade robotics has rendered outstanding services in the field of medicine. Limiting this study to therapist robots, several agents have been developed by the scientific community for helping seniors in their therpies. On one hand, there exist robotics structures that allow patients to achieve physical tasks. Armeo Spring [4] is a robotic device intended for patients with multiple sclerosis. It aids for the rehabilitation of upper extremities and consists of an anti-gravity support that can be adjusted to the arm and the forearm, and help the gradual reorganization of the brain. ReoGo robot [5] is also a device that helps patients recover from cerebrovascular accidents to regain the upper motor functions. It consists of two connected platforms: a seat with an armrest ring and a monitor that sends signals that direct the arm by a sequence of movements. MIT Manus robot [6] is also a robotic device designed to help recover from cerebrovascular accidents where the patient hold a robotic control lever.

On the other hand, there exist autonomous robots that perform therapies directly with patients, like a professional supervisors that interact, record and adapt itself to several real situations. Ursus robot [7] is an autonomous agent developed by RoboLab, and it is designed to propose games to children with cerebral palsy in order to improve their recovery. The last version is equipped with RGB-D cameras for acquiring information from the environment, and with a speaker, microphones and two robotics arms for interacting with humans. Ursus has been used in real therapies in Hospital Virgen del Rocío from Sevilla, Spain, with excelent results [7]. In similar therapies, also with chidren with some kind of disability in their upper limbs, Nao robot has been succesfully used [8]. Both robots, Ursus and Nao, makes the exercises that children should imitate, and they are able to perceive the patients reactions and interact with them, modifying the exercises when they are not properly conducted. The aim of these works is that the child perceives the therapy as a game, where in addition, the robots look like toys, and thus the little patients encourage to perform the exercises of the therapy.

III. 'Simon says' Game

'Simon Says' is a traditional game played with an electronic device with four buttons of different colours (yellow, blue, green and red). The device marks a colours and sounds sequence, that the players should remember and reproduce by pushing the different buttons of the device. When the turn played increases, the difficulty levels increases too ($i.e.$ the number of buttons that came into the sequence is increased). Therefore this game develops the users memory, because the sequence that Simon is showing, each time with more difficulty, has to be remembered. In this paper, a modified version of 'Simon says' game is described for using in terapies with elderly. Instead of a sequence of colours and sounds, a patient has to reproduce a sequence of different exercises during the therapy. Henceforth the proposed game helps to develop both the patients psychomotricity and his memory.

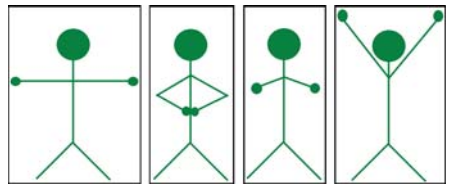


Fig. 2. Set of exercises used in the therapy with robot proposed in this paper.

A. Exercises

In this section, the set of exercises proposed in the therapy are described. It is composed of four different human pose with a relatively low difficulty level. These exercises have been defined by a occupational therapist team, and they are especially designed for elderly people. Fig. 2 illustrates the set of exercises used in the proposal which are described below.

- **Exercise 1**: To cross the arms. To the correct realised of the exercise, it is important that the hands are at the equal level of the shoulder.
- **Exercise 2**: To put the hands in the hips, with arms akimbo.
- **Exercise 3**: To hold the arms out in front of the body, in such a way that the hands stand in the shoulders level.
- **Exercise 4**: To raise the hands over the head, with the arms outstretched.

IV. RGBD data analysis for human pose recognition and classification

The proposed approach is based on the analysis of RGBD image sequences. From each image, the human skeleton is detected and a set of human body features is extracted and characterized. First, the system has to determine when the senior is achieving a movement and when he has finished it and has changed his pose. Then, the robot has to extract body features and classify the human pose into the set of pose associated to each exercise.

A. RGBD image sequence acquisition and human skeleton detection

In this work, a low level RGB-D camera is used for acquiring video and distance data (Kinect sensor for Windows). This sensor is characterized by a 640x480 video resolution, and by a spatial x/y resolution of 3 mm at 2 m distance from the sensor [9]. This camera produces a real-time 30fps stream of VGA frames, which is enough for a common therapy. In order to quickly and accurately predict 3D positions of body joints from a single depth image, using no temporal information, the method described in [10] is used. Finally, only eight body joints are specified: head $H_{(x,y,z)}$, left and right shoulders, $LS_{(x,y,z)}$ and $RS_{(x,y,z)}$), left and rigth elbows, $LE_{(x,y,z)}$ and $RE_{(x,y,z)}$), left and right hands, $LH_{(x,y,z)}$ and $RH_{(x,y,z)}$), and chest $Ch_{(x,y,z)}$.

B. Body gesture detection and characterization

Once the skeleton of the patient is detected, the system extracts and characterizes a set of human body gesture. These features are described below:

- **Quantity of motion** (QoM). It is a measure of the amount of detected skeleton motion. This feature is useful to know when the patient is moving and then, discriminate the data collected at that time (exercises performed in the proposed therapy are static). Let N being the total number of consecutive frames taken, and x_i^A being the 3D position of an articulation at an instant of time i. Then, QoM^A is defined as:

$$QoM_i^A = \frac{1}{N} \cdot \sum_{k=0}^{N} x_i^A - x_{i-1}^A \tag{1}$$

  where $A \in$ (*left hand, right hand, left elbow, right elbow, chest, head)*. Finally, the total QoM is evaluated as the average value of QoM^A.
- **Contraction Index** (CI). It measures the amount of contraction and expansion of the body. It takes range of values between 0 and 1, and is bigger when the patient's posture keeps limbs near the baricenter. To calculate CI value, the relationship between the human chest and the hand poses has been taken into account. This relationship has been carried out by the area of the triangle defined by these three points ($i.e.$, chest and two hands). First, the semiperimeter s of the triangle is calculated:

$$s = \frac{u + v + w}{2} \tag{2}$$

  where u, v and w are the sides of the triangle. The CI, using Heron's Method, remains as follows:

$$CI = \sqrt{s \cdot (s-u) \cdot (s-v) \cdot (s-w)} \tag{3}$$

- **Angle Arm** (α). This feature allows the system to get information on whether the arms are stretched or bent. In order to calculate α value, a triangle is builts as:
  - Side A: length from the shoulder to the elbow
  - Side B: length from the elbow to the hand
  - Side C: length from the shoulder to the hand

  As hands, elbows and shoulders positions are known, these lengths can be easily calculated. Hence, the desired α angle can be calculated by the law of cosines, which reads as follows:

$$\begin{aligned} A^2 &= B^2 + C^2 - 2BC \cot cos(\alpha) \\ \alpha &= arccos\left(\frac{B^2 + C^2 - A^2}{2BC}\right) \end{aligned} \tag{4}$$

- **Height of the hands** (Y). It shows the height at which the hands are. First, it has been assumed that the ground of the reference system is placed in the subject's foot to solve the problem that the reference axis is located in the Kinect. The maximum value that can take the hand is calculated from the value of arm's length plus the shoulder's height. By defining $H = Hand$, $E = Elbow$, $S = Shoulder$ and $L = Length$, and being X, Y and Z the 3D-coordinate associated to a joint, then

$$L_{EH} = \sqrt{(E.X^2 - H.X^2) + (E.Y^2 - H.Y^2) + (E.Z^2 - H.Z^2)}$$

$$L_{SE} = \sqrt{(S.X^2 - E.X^2) + (S.Y^2 - E.Y^2) + (S.Z^2 - E.Z^2)}$$

$$ArmLength = L_{EH} + L_{SE} \tag{5}$$

$$Y_{max} = ArmLength + S.Y \tag{6}$$

  Finally, the hand height (Y) can be normalized as is shown in (7).

$$Y = \frac{H.Y}{Y_{max}} \tag{7}$$

- **Depth of the hands** (D) This feature reports whether the hands are in the z plane of the body or not. To calculate this value, the difference between the depth value of hands and the depth value of the chest has been computed, as is shown in (8).

$$D = |H.Z - Ch.Z| \tag{8}$$

Finally, expected values for the features described in this section for the different exercises of the therapy 'Simon Says' are summarized in Table I.

C. Human body pose clasification

Classification models are algorithms that are able to learn after performing a training process with a known data. Then, classification algorithms are able to make decisions and classify new data based on this training. In this paper these models allow the therapist robot to have more autonomy by deciding if the exercises have been implemented correctly. In the proposed work three well-known models are studied: *Decision Tree* (DT) [11] [12], *K-Nearest Neighbours* (KNN) [13] [14], and *Support Vector Machine* (SVM) [15] [16].

V. Experimental Result

In order to demonstrate the accuracy and robustness of the proposal, a set of experiments has been conducted. Fig. 3 shows the robot used in the experiments with the RGB-D sensors marked on the picture. Currently, this robot has two different RGB-D cameras, but only those one marked as '1' is used for the therapy. First, this section describes the database used for the experiments and the main interface of the application. Then, a set of representative tests are evaluated. Both tests and application are programmed using RoboComp framework, developed by RoboLab [17].

A. DataBase

A database with recorded data for the training and recognition process has been created. This database consists of data of 20 subjects that have been recorded with the Kinect. All files that compose the database are in text format (.txt). Data from 5 of them has served to train the component, whereas the recognition tests were carried out with the remaining 15 subjects whose data have not been used for training.

This database has been shared and it is available through the download service of the Universidad de Extremadura, through the link http://goo.gl/IxovJ4, so that any developer can use it with their systems.

Exercise	CI Value	$\alpha_{RightArm}$	$\alpha_{LeftArm}$	$Y_{RightHand}$	$Y_{LeftHand}$	$D_{RightHand}$	$D_{LeftHand}$
Exercise 1	High	180	180	Right Shoulder Height	Left Shoulder Height	0	0
Exercise 2	Low	90	90	Right Hip Height	Left Hip Height	0	0
Exercise 3	Low	180	180	Right Shoulder Height	Left Shoulder Height	Arm Length	Arm Length
Exercise 4	Medium	180	180	Over Head	Over Head	0	0

TABLE I
EXPECTEDFEATURES FOR DIFFERENT EXERCISES OF THE THERAPY 'SIMON SAYS'

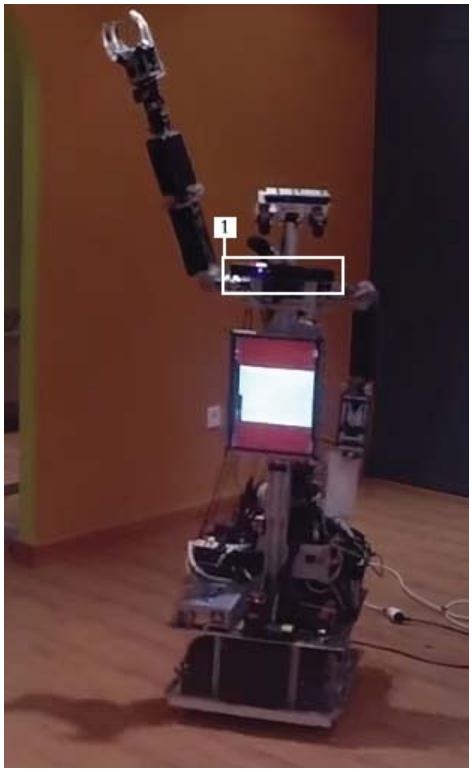


Fig. 3. New robot Ursus used in the experiments

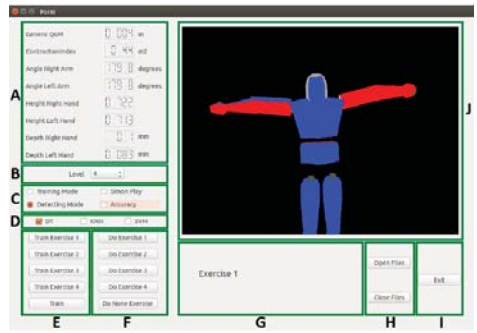


Fig. 4. Interface of the application described in this work

B. Interface

As is shown in Fig.4, the interface of the application is composed by several parts that are detailed below:

- **Part A:** This block shows the different calculated features.
- **Part B:** It allows to choose the level of play in the "Simon Mode".
- **Part C:** This part enables to chose the mode wherein the component is developed: Training, Detecting, Accurace[2] and 'Simon Game' modes.
- **Part D:** It lets you select the decision method to be used.
- **Part E:** These buttons allow to train the component.
- **Part F:** These buttons simulate the performance of one of the defined exercises.
- **Part G:** This label shows different messages depending which mode is selected.
- **Part H:** These buttons stores variables and positions data in a Matlab file to generate graphs.
- **Part I:** That button successfully stops the program.
- **Part J:** This screen shows the patient performing the different exercises.

[2]This mode allows to calculate the accuracy of the different algorithms used, and the success rate of each exercise for each method

C. Evaluation of the features extraction algorithm

An analysis of the features extracted for each exercise has been conducted. The tests check whether if these features are among the expected values, and if they discriminate between the different exercises. To do this, some graphics have been generated using Matlab software.

Fig.5 shows the Contraction Index that has been calculated for the different exercises. As is shown in the figure, the values of the Contraction Index are the expected: a high value for the Exercise 1, in which the hands are widely separated from the body, a median value for the Exercise 4, since this exercise also separates the arms from the body, but the arms are close together, and a low value for the Exercises 2 and 3.

In Fig. 6 and Fig. 7 are illustrated the data collected with respects to the angles formed by the right and left arms, respectively. As the graphics show, the values of the angles are between expected, being approximately $180°$ for the Exercises 1, 3 and 4, due to in these exercises the arms are stretched, and about $90°$ for the Exercise 2, because in this exercise the arms are flexed. As can be appreciated, for this subject there are several oscillations in the value of the calculated angle for

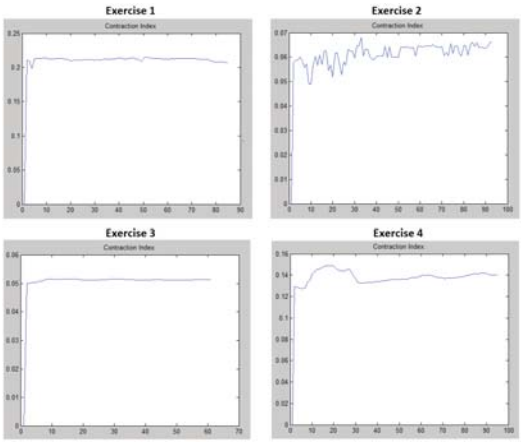


Fig. 5. Contraction Index for the different exercises

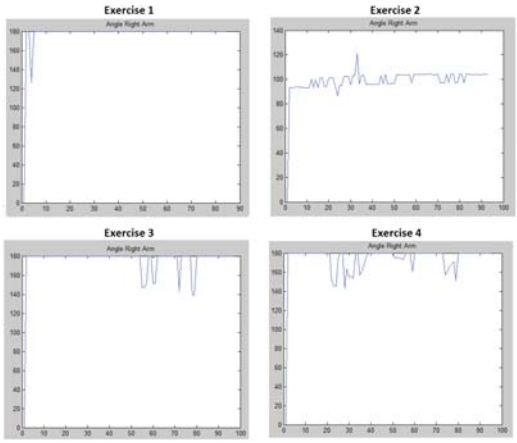


Fig. 6. Angle of the Right Arm for the different exercises

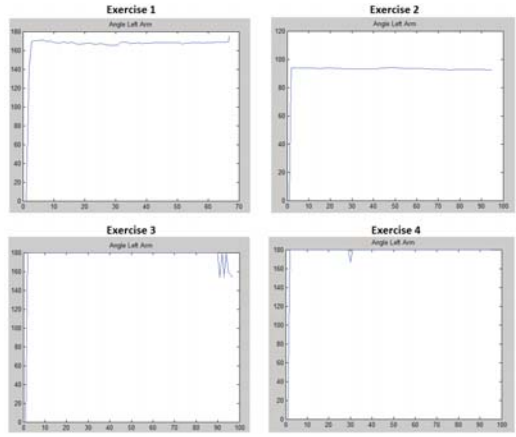


Fig. 7. Angle of the Left Arm for the different exercises

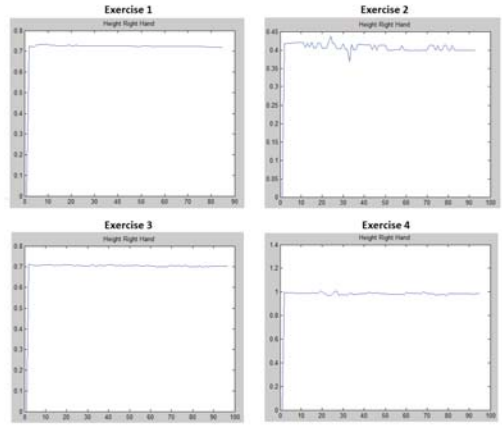


Fig. 8. Height of the Right Hand for the different exercises

the right arm, due to the values detected by the kinect. This could create confusion in the recognition process.

Fig. 8 and Fig. 9 show the heights of the hands normalized to 1 relative to each exercise. The results are again as expected, since in Exercise 4, in which the hands are above the head they find their maximum with an average of 1. For Exercises 1 and 3 hands are at shoulder height, obtaining an average value of 0.7. And finally, for Exercise 2 hands are on his hips, the hands take an even lower value of around 0.4.

Finally, data associated to the depth of both hands are shown in Fig.10 and Fig.11. It can be seen that for the Exercises 1, 2 and 4, in which the hands are kept in the plane of the body, some very low values, close to 0, are obtained. However, for the exercise 3 a somewhat higher value, between 60 and 70 cm, is obtained, which is about the length of the arm, as expected, because in this exercise the arms are stretched forward.

D. Recognition Accuracy

This section evaluates the accuracy of the three decision methods described in this paper. Table II shows the results obtained. As is illustrated in the table, the decision method that gets the best results in recognizing the exercises is the Decision Tree, with an average of probability of success of 99.61%. KNN method also generally give good results, although not as good as with the DT algorithm. This may be because as explained, the value of K influence the result, but at the same time the optimum value of K depends on the dataset, so as the calculation is made with a single value of K, the best possible results may not have achieved (K = 5 in the experiments, after an analysis of the results). SVM method presents some results in which the success rate is very satisfactory and others where it is not as good, but in any of the cases the accuracy obtained is higher than the accuracy obtained in the other methods. This low accuracy can be obtained because some exercises are very similar to each other, differentiating between them only by two

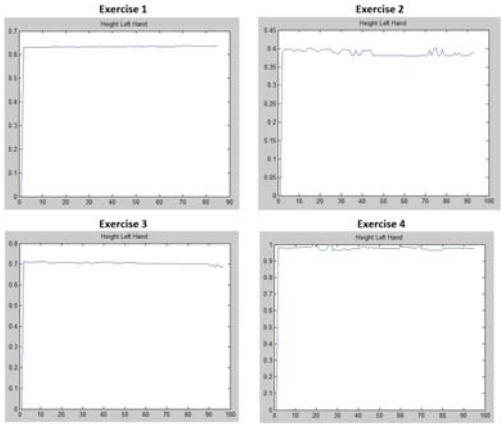


Fig. 9. Height of the Left Hand for the different exercises

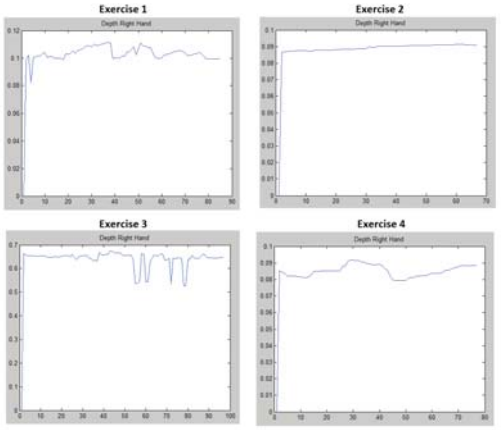


Fig. 10. Depth of the Right Hand for the different exercises

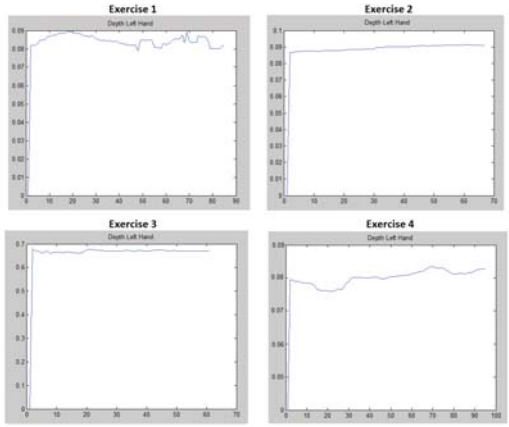


Fig. 11. Depth of the Left Hand for the different exercises

Subject	DT	KNN	SVM
1	100.00%	91.67%	80.00%
2	100.00%	95.83%	87.50%
3	100.00%	81.67%	62.50%
4	100.00%	72.50%	61.67%
5	99.17%	70.00%	39.17%
6	100.00%	77.50%	42.50%
7	100.00%	80.00%	75.83%
8	95.00%	68.33%	56.67%
9	100.00%	71.67%	45.00%
10	100.00%	95.00%	87.50%
11	100.00%	62.50%	38.33%
12	100.00%	68.33%	50.00%
13	100.00%	97.50%	65.83%
14	100.00%	76.67%	67.50%
15	100.00%	81.67%	65.83%
Mean	99.61%	79.38%	61.72%

TABLE II
ACCURACY OBTAINED FROM THE DIFFERENT DECISION METHODS

features.

Therefore the Decision Tree algorithm gets better results because it asks questions about the variables to decide whether an exercise or another. On the other hand the KNN and SVM algorithms represent the points spatially, then the point that must be classified is painted to decide which exercise belongs, KNN algorithm by comparing to the closest points, and the SVM algorithm depending on the area where the point is. As these exercises are differentiated by a few characteristics, it is possible that the points are together in the spatial representation and this creates confusion when deciding.

VI. CONCLUSIONS

This paper present a novel system for supervising rehabilitation therapies using autonomous robots for elderly. In the therapy 'Simon says' described in this work a robot executes a list of motions and gestures that the human has to repeat each time with a more level of difficulty. Four different exercises ($i.e.$, human body poses) have been described in the game, thus, the therapist robot has to be able to detect, recognize and classificate human body poses. To do that, in this paper has been described a set of human body features in order to characterize these poses, and also three different classification algorithms have been evaluated. The results of this work demonstrates the accuracy of the described algorithm. Finally, a RoboComp component has been included in the repository for later development.

ACKNOWLEDGMENTS

This work has been partially supported by the MICINN Project TIN2015-65686-C5-5-R, by the Extremadura Goberment project GR15120, and by COMPUTAEX fundation.

REFERENCES

[1] Instituto Nacional de Estadstica in http://www.ine.es/
[2] Global Health and Aging. World Health Organization. NIH Publication no. 11-7737, October 2011

[3] M. Jara. *La estimulación cognitiva en personas adultas mayores*. Revista Cúpula.
[4] Armeo Therapy Concept in http://www.hocoma.com/fileadmin/user/Dokumente/Armeo/bro_Armeo_Therapy_Concept_140226_en.pdf
[5] ReoGo Robot in http://www.sanar.org/salud/terapia-robotica
[6] MIT Manus Robot in http://universitam.com/academicos/?p=1052
[7] L.V. Calderita, P. Bustos, C. Surez, F. Fernndez, R. Vicianad, and A. Bandera. Asistente Robtico Socialmente Interactivo para Terapias de Rehabilitacin Motriz con Pacientes de Pediatra. ScienceDirect Revista Iberoamericana de Automtica e Informtica industrial Vol. 12, pp. 99-110, 2015.
[8] NAO Robot in http://www.eurekalert.org/pub_releases/2015-04/ciuo-dar042015.php
[9] Microsoft Kinect for Windows Information in URL http://www.microsoft.com/en-us/kinectforwindows/meetkinect/default.aspx
[10] RGBD Data Analysis for Real-Time Emotion Recognition from Upper Body Movements. F. Cid, E. Mogena, L. J. Manso3, and P. Núñez. In Proceedings of International Workshop on Recognition and Action for Scene Understanding (REACTS 2015), 2015.
[11] R. Barrientos, N. Cruz, H. Acosta, I. Rabatte, M.C. Gogeascoechea, P. Pavn and S. Blzquez. *Decision trees as a tool in the medical diagnosis*. Revista Mdica Vol. 9(2), pp. 19-24, 2009.
[12] J. Trujillano, A. Sarria-Santamera, A. Esquerda, M. Badia, M. Palma and J. March. *Approach to the methodology of classification and regression trees*. Gac Sanit, Vol. 22(1), pp. 65-72, 2008.
[13] KNN Algorithm from OpenCv in URL http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_ml/py_knn/py_knn_understanding/py_knn_understanding.html
[14] A. Moujahid, I. Inza and P. Larraaga. *Clasificadores K-NN*. Departamento de Ciencias de la Computacin e Inteligencia Artificial Universidad del Pas VascoEuskal Herriko Unibertsitatea.
[15] G. A. Betancourt. *Las mquinas de soporte vectorial (SVMs)*. Scientia et Technica Ao XI, UTP. ISSN 0122-1701, No 27, Abril 2005.
[16] D.A. Salazar, J.I. Vlez and J.C. Salazar. *Comparison between SVM and Logistic Regression: Which One is Better to Discriminate?*. Revista Colombiana de Estadstica, Nmero especial en Bioestadstica, Vol. 35, No. 2, pp. 223-237, Junio 2012.
[17] L. Manso, P. Bachiller, P. Bustos, P. Núnez, R. Cintas and L.Calderita. RoboComp: a Tool based Robotics Framework, In Proceedings, SIMPAR Second International Conference on Simulation, Modeling and Programming for Autonomous Robots. pp 251-262. 2010.

Cybersecurity in Autonomous Systems: Evaluating the performance of hardening ROS

Francisco Javier Rodríguez Lera, Jesús Balsa, Fernando Casado, Camino Fernández, Francisco Martín Rico, and Vicente Matellán

***Abstract*—As robotic systems spread, cybersecurity emerges as major concern. Currently most research autonomous systems are built using the ROS framework, along with other commercial software. ROS is a distributed framework where nodes publish information that other nodes consume. This model simplifies data communication but poses a major threat because a malicious process could easily interfere the communications, read private messages or even supersede nodes. In this paper we propose that ROS communications should be encrypted. We also measure how encryption affects its performance. We have used 3DES cyphering algorithm and we have evaluated the performance of the system, both from the computing and the communications point of view. Preliminary results show that symmetric ciphers using private keys impose significant delays.**

***Index Terms*—Autonomous systems, Cybersecurity, Data security, Cyber-physical systems, ROS, Performance analysis**

I. Introduction

AUTONOMOUS systems are spreading not just in the virtual world (Internet, software systems), or in science-fiction movies, but in our ordinary real world. It is possible to find driverless cars in the streets, autonomous vacuum cleaners in our homes, autonomous robotic guides at museums, etc. These robotic systems, as any computer-based system, can suffer different types of cyber-attacks, and some degree of cybersecurity [6] is required.

Our research group is developing an assistant robot [4] for the elderly. When we initiated experiments involving potential users, caregivers asked us about the security of our robot and about the privacy of its communications [1]. When an assistant robot carrying a camera is deployed in a home, the access to this camera should be secured; even more when the robot is managing medical information.

We have developed all the software that controls the autonomous behavior of our robot using ROS (Robotic Operating System) [7] which has become the most popular framework for developing robotic applications. It started as a research framework, but currently most of manufacturers of commercial platforms use ROS as the *de facto* standard for building robotic software. For example, object-manipulation robots like Baxter (by Rethink robotics) [2] or service robots as our RB1 (by Robotnik) are ROS based platforms.

Francisco Javier Rodríguez Lera, Jesús Balsa, Fernando Casado, Camino Fernández, and Vicente Matellán are with the Robotics Group (http://robotica.unileon.es) and the Research Institute on Applied Sciences to Cybersecurity (http://riasc.unileon.es) at Universidad de León (Spain).

Francisco Martín Rico is with Robotics Lab, Signal Theory, Communications, Telematic Systems and Computation Department at Universidad Rey Juan Carlos, Madrid (Spain).

Corresponding author: vicente.matellan@unileon.es

A. Security assessment

There are three basic vulnerabilities threatening any computer system: availability (data interruption), confidentiality (data interception) and integrity (data modification). Other authors also add two more [9]: authenticity and non-repudiation (data fabrication from a non-trusted origin). These concepts can be easily translated to industrial control applications [3] or to robotic environments, due to the distributed approach used in most used robotic frameworks (Yarp, ROS, ROBOCOMP).

Let's illustrate this problems considering a robot deployed in a home environment. This robot provides information to users and carries out its behaviors in order to fulfill required tasks. An attacker could attempt to make this robot or its network resources unavailable, this is a denial-of-service attack. The attacker could also intercept and modify command messages in order to change robot behavior, and capture robot information about the environment and about the users. Finally the attacker could simulate a sensor generating new data and sending it to the robot. Fig. 1 summarizes graphically these vulnerabilities in a robotic environment.

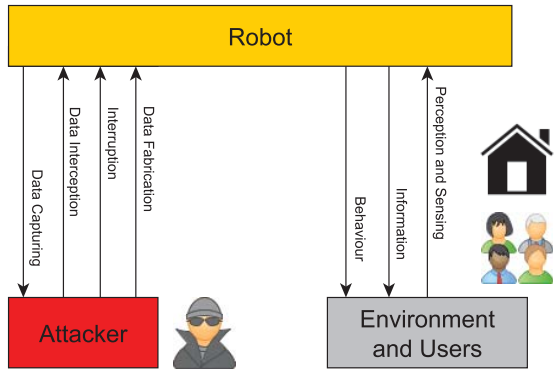


Fig. 1. Conceptual model of the security attacks.

In this research we focus on the confidentiality problem of the data sent to and from the robot. Our proposal consists in encrypt data transmitted between ROS processes, adding two nodes for encryption and decryption task. We don't change ROS messages nor ROS standard functions to send the data. The question is if this hardening of ROS will impact on its performance.

The reminder of this paper is organized as follows: Next section describes the ROS framework communication process.

Section III defines the testbed designed to measure the performance of the encrypted ROS system. Section IV shows the data obtained in the proposed experiments as well as the discussion. Finally section VI presents the conclusions obtained and further work.

II. ROS OVERVIEW

ROS provides specific libraries for robotics similar to classical operating system services such as hardware abstraction (for sensors and actuators), low-level device control, and inter-process communication. Inter-process communication is organized as a graph architecture where computation takes place in ROS processes named nodes. These nodes can receive and send messages. Unfortunately no security was considered in the communication mechanism.

ROS framework is basically a message-passing distributed system. Its architecture is based on processes that publish *messages* to *topics*. For instance, a process (*node*) can be in charge of accessing a sensor, performing the information processing, and publishing it as an information structure on a named topic. Another process can *subscribe* to this topic, that is, read its information. Then the process can make a decision about the movement of the robot. Next, this node will publish the commands in another topic to send them to the motors. ROS nodes can be running in the same computer or in different computers.

Usual ROS configuration is composed by at least one ROS Master and some clients. ROS Master is the key element in the ROS system. It runs as a nameservice and manages registration information about all topics and services used by ROS nodes.

A node communicates with the Master to register its information. Then, the node gets information about other registered nodes, to be able to establish new connections with their topics appropriately. The Master is updated in real time by nodes registering information and topics they publish/subscribe. Fig. 2 summarized the six steps involved in this process as presented in [8].

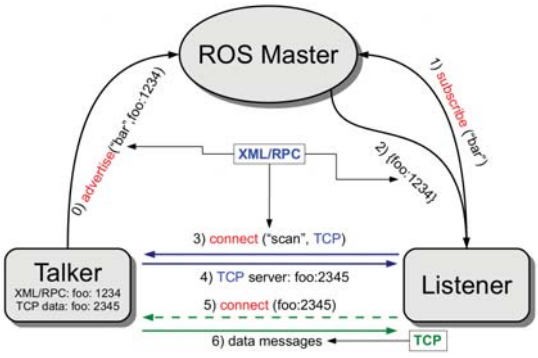


Fig. 2. Conceptual model of ROS topics presented by Radu Rusu in his tutorial[8].

The ROS distributed approach is very convenient for developers but can be easily tampered by malicious hackers. For instance, in [5] an experiment involving a ROS-based honeypot is described. The honeypot was a radio model truck with two cameras and a compass as sensors, and it was controlled from a remote ROS node written in Javascript and hosted in a remote enterprise grade web server. Vulnerabilities described in the paper comprise plain-text communications, unprotected TCP ports and unencrypted data storage.

The first step to solve some of these problems is to secure the communication channels by using an encryption mechanism. But how does encryption impact on the performance of a robotic system? This is the goal of this paper, characterize and evaluate different alternatives to secure ROS communication system and measure their performance.

III. TESTBED DESCRIPTION

In order to evaluate the performance of the encrypted version of ROS communications environment, we designed the following scenario. Fig. 3 provides a graphical representation of the scenario created.

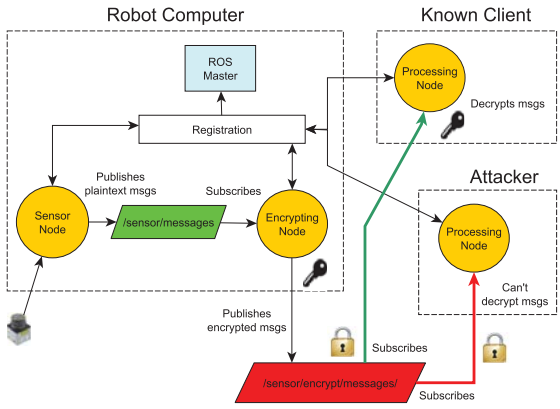


Fig. 3. Scheme of the scenario used for testbed.

First, we installed ROS Indigo in the on-board computer of the our RB1 mobile robot. The ROS master component ran in this platform. The robot was connected by wire to our laboratory network infrastructure for this experiments. The robot computer has two nodes: one node connected to a sensor that publishes the data into */sensor/messages* topic; and one node connected to this topic that performs data encryption and publishes them into a */sensor/encrypt/messages* topic.

Second, we used one desktop computer as a "known client", also with ROS Indigo installed and connected by cable to our network. This client knows the master ROS IP, so it can communicate with master. We run a decryption node in the client computer, which registers to master and subscribes to the topic */sensor/encrypt/messages*. This node decrypts data and prints them on the screen.

Third, we used another desktop computer as a simulated "attacker", connected to the same cable network. This computer has the same ROS version running on it. The attacker doesn't know the master ROS IP, but he can easily discover it performing a network scan with well known tools like

nmap. Then, the attacker could execute a malicious node for attempting to read laser data, which is being published in the topic */sensor/encrypt/messages*. Despite the node could subscribe to that topic, all data received is encrypted. As a result, the malicious node can't see original laser messages because the attacker doesn't have the key to decrypt them.

The cryptographic key is stored in the master node and it is known by the legitimate clients. In a system in production this mechanism can be implemented as a public-private key schema as RSA to safely share the key between the nodes.

A. Encrypting ROS messages

In this first approach we have changed the data published by the node, and instead of publishing the information in a plain manner, we publish the information encrypted.

ROS uses a *messages description language*[1] for describing the data values that each node publish. In this manner, it is easy for ROS tools to automatically generate source code for the message type in several target languages as ICE or IDL specification language.

There are built-in types (14), array types (4) and customized types. For instance, if we get the *sensor_msgs/Image.msg* message, we find a composed message by one non built-int message, and five built-in (3 x uint32, 1 x string, 1 x uint8)and one array type (uint8[]).

```
std_msgs/Header header #custom msg

uint32 height
uint32 width
string encoding
uint8 is_bigendian
uint32 step

uint8[] data #matrix data of an image
```

In this case is *data* field the element that contains the information of the image grabbed by the camera, so we are going to encrypt this element before to be publisher in ROS distributed system.

We use the 3DES algorithm to provide a security layer to ROS data. It is a known that 3DES cyphering speed is slower [10] than other cipher methods as for instance AES. We use this algorithm as the worse environment to analyze its behavior in a real environment with ROS. Triple DES (3DES) references the Triple Data Encryption Algorithm (TDEA or Triple DEA). It is a symmetric-key block cipher that applies the Data Encryption Standard (DES) algorithm three times to each data block. It is standardized by NIST in the Recommendation for the Triple Data Encryption Algorithm Block Cipher (NIST Special Publication 800-67).

3DES algorithm provides three keys that are 128 (option 1) or 192 (option 2) bits long. In option 1, the key is split into K1 and K2, whereas $K1 = K3$. The option 2 is a bundle of three 64 bit independent subkeys: K1, K2, and K3. To highlight that the three keys should be different, otherwise 3DES would degrade to single DES.

The data block of the algorithm has a fixed size of 8 bytes where 1 out of 8 bits is used for redundancy and do not contribute to security. In that manner, the effective key length is respectively 112 in option 1 and 168 bits in option 2.

The algorithm presents the next behavior, the plain text is encrypted three times: first it is encrypted with K1, then decrypted with K2, and finally encrypted again with K3. The ciphered text is decrypted in the reverse manner.

3DES is cryptographically secure, although it is slower than AES algorithm. In a next stage, we will substitute 3DES with AES in our encryption system and repeat all the test we have done, to compare the performance.

From the development side, it was used the PyCrypto package. It is an extended python Cryptography Toolkit that allows to simply the method to encrypt or decrypt in multiple encryption algorithms.

IV. Experimental Measurements

To illustrate the described approach, we present an ad-hoc implementation of an encryption system to change part of the message used for transmit or receive robot sensors information.

We have added a function to our program in order to measure the time spent on each encryption and decryption call. The function is a python method presented as a decorator pattern, which is used here to extend the functionality of encryption/decryption at run-time.

```
def fn_timer(function):
          @wraps(function)
          def function_timer(*args, **kwargs):
              v_time_0 = time.time()
              result = function(*args, **kwargs)
              v_time_1 = time.time()
              return result
          return function_timer
```

We have divided the experiments in three parts. First we have analyzed how the "Publisher-listener" ROS nodes work under encrypted conditions. Then we have stressed the same nodes publishing bigger text messages, in plain text as well as encrypted text. Finally we have analyzed a camera sensor, under the same encryption/decryption conditions.

A. Hardware/Software Set-up

We want to evaluate how the encryption of communications would affect the performance of ROS. We have used the better of the cases where a 8x Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz with 16231MB of RAM Memory and running an Ubuntu 14.04.4 LTS Operating System. The ROS master system has 234 process running by default, the client is running 242 process.

The two computers are connected by an Alcatel-Lucent OmniSwitch 6860E switch This hardware conditions are the most favorable against the real environment of a robotic platform for instance wireless or hardware restrictions.

B. Test 1: HelloWorld Publisher-Listener Node

In this test we have used the version of talker/listener tutorial proposed by ROS[2]

[1] http://wiki.ros.org/msg

[2] http://wiki.ros.org/rospy_tutorials/Tutorials/WritingPublisherSubscriber

This package distributed by ROS as a demo, presents a simple ROS package that creates two rospy nodes. The ”talker” node broadcasts a *Hellow world + Timestamp* message on topic ”chatter”, while the ”listener” node receives and prints the message.

TABLE I
TIME IN SECONDS OF CPU SPENT ON A SIMPLE PUBLISHER/LISTENER RUNNING TEST.

	Plain Publication	Plain Subscription	Encrypt Publication	Decrypt Subscription
Time running	34.491	34.484	36.882	34.995
Time user	0.184	0.196	0.348	0.24
Time sys	0.024	0.08	0.064	0.056
Total CPU	0.208	0.276	0.412	0.296

Table I presents the CPU time used when the nodes are running in different machines. The values are the result of launching ROS nodes using the Unix command *time*. The values present a minimal consumption of CPU associated to the process. In case of plain publish/subscribe it represents than 1% and in the case of the publisher in encrypt mode it represents approximately the 1.1%, that is almost the same than in plain mode.

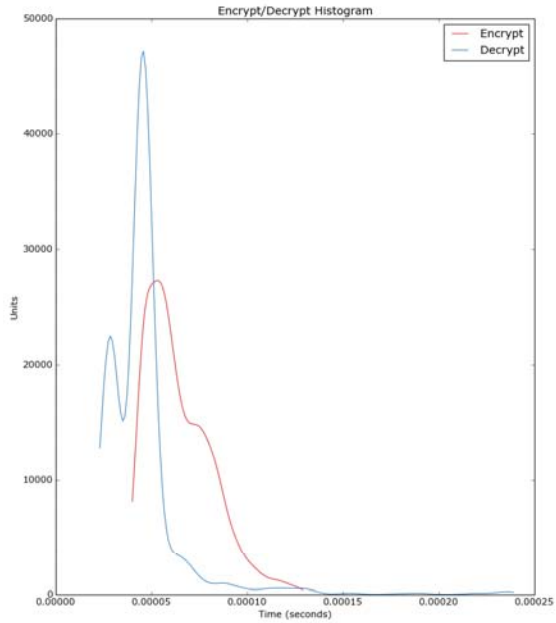


Fig. 4. Histogram showing the time spent on a encrypt/decrypt DES3 function call.

Figure 4 presents the histogram associated to the calls to encrypt/decrypt method used in these tests. We can observe that the average time in the encryption process is slightly higher (0.000065 seconds) than the decryption process (0.000045 second). However, in both cases it is almost negligible. The worst case presented a 0.000129 seconds in ciphering time and 0.000239 second of deciphering time.

Figure 5 presents the screenshot of this experiment. It shows the three terminals from the two nodes involved in this experiment. Publisher node (top-left window) that presents a simple log, subscriber node with key (bottom-left) that presents the message after the decryption call and subscriber node without key (*rostopic echo*) that shows the encrypted message in the topic.

C. Test 2: Custom Text Publisher-Listener Node

In this test we have used the same version of talker/listener nodes proposed by ROS. In this particular test we generate a synthetic text strings with different sizes:

- T1: this is a string message of 262144 bytes (256 KB)
- T2: it is a string message of 524288 bytes (512 KB)
- T3: presents a string message of 1048576 bytes (1024 KB)

We want to determine the duration of execution of our talker and the time spent by our listener nodes. Again, we run the Linux *time* command to measure the total CPU time consumed by the ROS talker process.

Initially we run the test with the three size types of messages using plain text. We have performed the test three times, running the nodes for a time lapse of 30 to 35 seconds. Firstly, reviewing the T1 type (it is presented in a time window of 32.884 seconds) we find a CPU time of 1.408 seconds. It was a user time of 1.364 seconds and a sys time of 0.044s. I Secondly, we analyse the string of type T2. It was launched in a window of 33.749 seconds, and needed a total CPU of a 2.672 seconds (user time of 2.508s and a sys time of 0.164s). Finally, we analyse T3, that presents in a time window of 34.731 seconds a total CPU of 5.260 seconds (it is divided by a user time of 5.140s and a sys time of 0.120s).

This is totally different when we are working with encrypted text messages. Fig 6 presents the main differences. We have repeated the same experiment, but this time calling an encryption method that encrypts from plain text to 3DES. It is possible to see that the encrypt process consumes more CPU than the plain process. For instance, T1 type presents running for 34.486 seconds a total of 23.008 of CPU. This means that the total CPU time increases almost 62%. This is even higher in T2 and T3 types with almost the 98% of real execution time. It is clear that the encrypted/decrypted approach consumes more CPU (yellow bars in Fig. 6) than plain text (orange bars in Fig. 6).

We know that the encryption process needs more CPU, but how long does the encryption/decryption takes in each iteration?. Just for clarification, we review the encryption phase in the publisher node. In case of T1 messages, during its window (34.486s) the node is able to encrypt 1040 messages. The minimal time to perform this task is 0.059246 seconds and the maximum time to encrypt the string is 0.683408 seconds. In average, the system spends 0.063858 seconds (standard deviation of 0.000388s). This time increases with T2 and T3 types. T2 type needs 0.123383 seconds in average with standard deviation of 0.000040s and T3 0.247303s with a standard deviation of 0.000137s. Fig. 7 presents the histogram associated to this experiment.

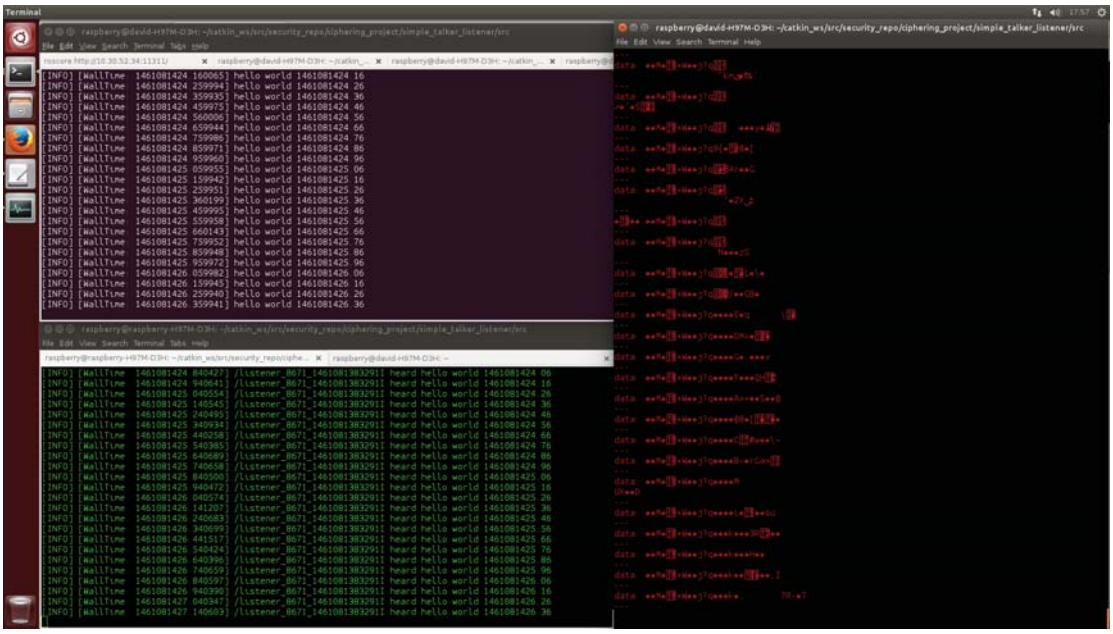


Fig. 5. Screen-shot with the nodes involved in the simple publisher/subscribe test. Upper left terminal presents the publisher node before encryption. Bottom left terminal depicts the subscriber node after decryption. Right terminal presents the results of rostopic echo on */chatter* topic used for transmitting info through nodes.

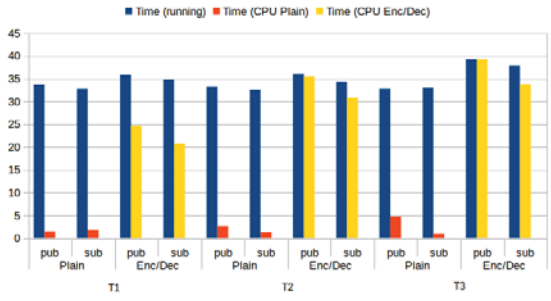


Fig. 6. Time of CPU spent by publiser/subscriber nodes sending T1, T2 and T3 messages in plain and Encrypted/Decrypted on test 2.

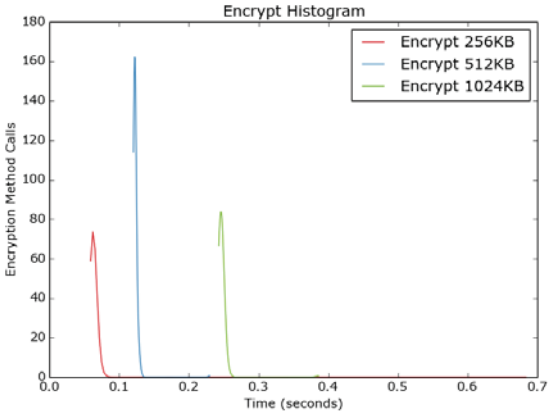


Fig. 7. Histogram by each publisher node attending each call to the encryption method.

D. Test 3: Camera Node

The third experiment has been developed using a RGB camera. A Logitech, Inc. QuickCam Pro 9000 webcam providing 15 frames per second (fps) using a 640x480 pixels resolution. This sensor was registered on ROS system to deliver images to the system.

This test involved three ROS nodes, two publishers: usb_cam, encrypted_node and one listener: decrypted_node. First ROS node,*usb_cam*[3] was based in the code contributed by Benjamin Pitzer and still maintained by ROS developers. It is in charge of recording information from the camera and publishing in the *usb_cam* topic.

The second node runs in the same computer and it is in charge of encrypting the message published by *usb_cam*. The third node runs on a different computer and it is in charge of decrypting the messages got from the publisher and of displaying the received image on the display.

Figure 8 graphically represents this set up for a given moment T at the listener node. We used the ROS node *image_view* to visualize images in both topics: the non-

[3]http://wiki.ros.org/usb_cam

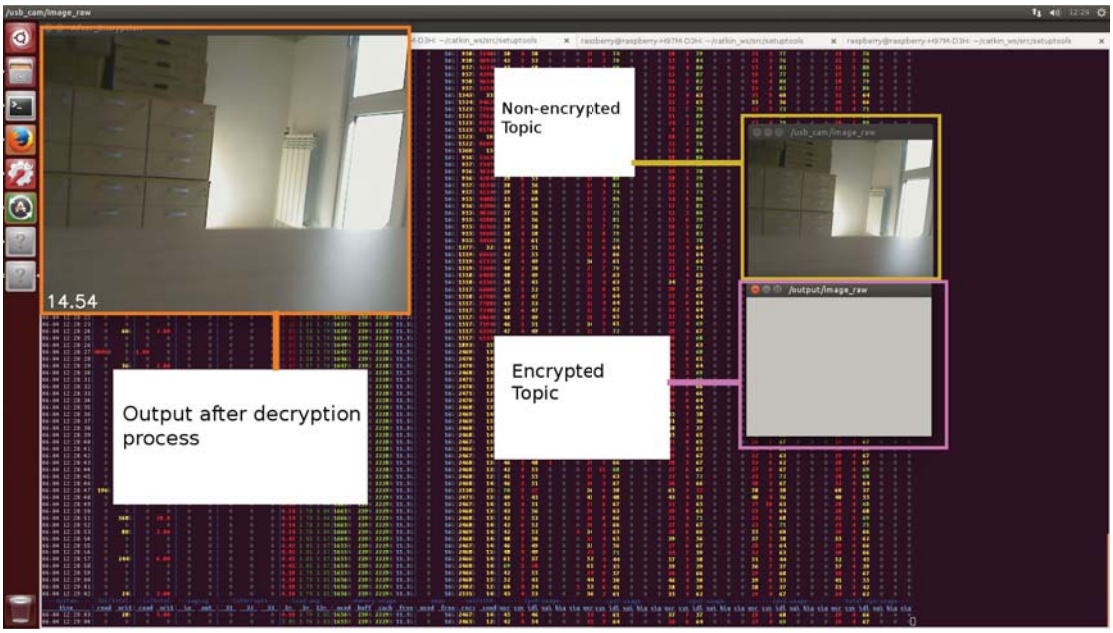


Fig. 8. Screenshot taken during image encryption test.

encrypted one, that to validate delays, and the encrypted one (that cannot be visualized) and the frame that shows the image after decryption.

We measured the time needed to encrypt and decrypt the ROS message (*sensor_msgs/Image.msg*). We just encrypted and decrypted the field data, defined as `uint8[]` and processed as a text string. Our nodes shows this information during the execution in a visualization frame that we used to measure the frame rate.

We ran this experiment for 18 hours sending in total 971.790 frames. The average time in the process of encrypt the sensor data was 0.010948 seconds (stddev 0.000004s). Minimal processing time was 0.001309 seconds and max processing time was 0.026909 seconds. The average time to decrypt the images was 0.008828 seconds (stddev 0.000003s). The max time decrypting an image was 0.039130s and the minimal 0.001288s.

V. Discussion

This study provides an in-depth characterization of the use of 3DES encryption in ROS communications. First, we have configured two ROS nodes equipped with state of the art computer power capabilities. The only software running on both machines in the first experiment was limited to the one needed for the test, no extra computational process were running during the experiments.

Results presented in this test showed that there is no difference in the performance between sending a plain string of chars *Hello world + timestamps* and an encrypted string of chars. However, the experiments showed that CPU time used by the encrypting of messages has a geometric progression with the size of the message, which means that the performance of the global system is reduced.

ROS has a logging system that measures the publish/subscribe performance among connected nodes. When the system is active on a node, the statistical data is published in a specific topic that is updated at 1Hz.Using this system we have observed two main issues:

- Traffic in bytes grows lineally with the number of messages:
  - plain mode delivers 10 messages of 346 bytes (average) of traffic.
  - encrypted mode delivers 10 messages per second whose size is 420 bytes in average.
- Performance decrease quadratically with the size of the messages:
  1) plain mode delivers 10 messages per second of 11.010.132 bytes of traffic (in average).
  2) encrypted mode only delivers 5 messages per second in average of 5.242.880 bytes of traffic (in average).

The performance reduction is due to the time consumed by the encryption and decryption process in each message: Publisher uses 0.247303 seconds per message to encrypt every message. Listener needs 0.240323 seconds per message just for decrypting. This means that the publisher process needs more time to produce messages and the listener more time to consume it.

In the same manner, if the publisher needs to process this information in any way, for instance showing the message

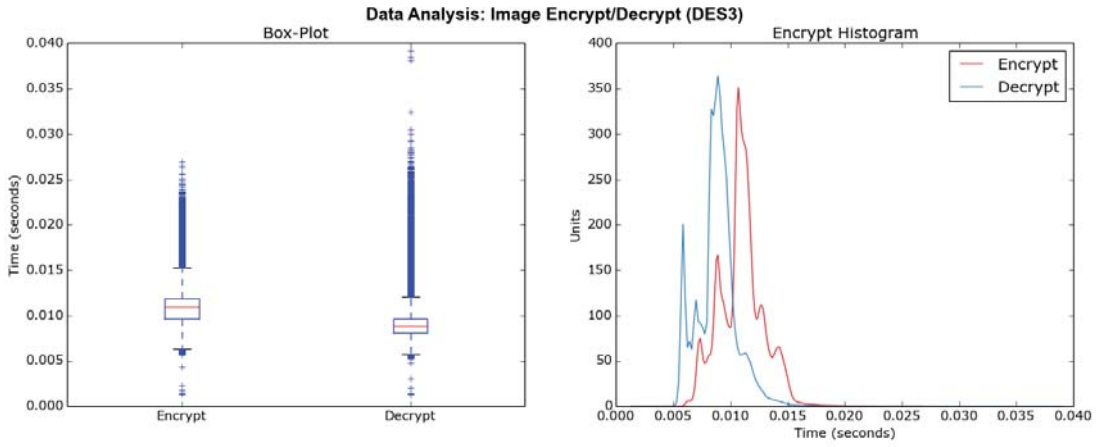


Fig. 9. Time spent on each call to encryption/decryption function during the image processing in test 3.

in the screen, the overall performance producer/consumer is reduced.

Finally, the experimental evaluation made in test 3 does not show a significant delay due to the encryption process. For instance, we got 14.56 frames per second in the publisher, and 14.54 frames per second in the listener.

Fig. 9 shows the performance of the publisher cyphering data. Left part of the figure summarizes the time used for encrypting and decrypting the three types of messages. The right part is the histogram of times used for calls to the encryption method. We observe that processing times in average are similar, the encryption process is slightly higher (2ms) than in the decryption process.

VI. Conclusion and Further Work

We have shown that using ciphered communications avoids security problems related with the plain-text publish/subscribe paradigm used by ROS. However, the overhead of CPU performance and communication load should also be considered in distributed architectures that need to work on real time.

This article presents a performance analysis of the use of encrypted communications in a ROS system. We have evaluated the impact of this added feature from two points of view: CPU consuming and network traffic.

As we pointed out in the introduction, securing communications is just one dimension in the cybersecurity of autonomous systems. If we want to see these machines working in our homes we need to secure navigation abilities and interaction mechanisms, to avoid manipulated or malicious behaviors and make robots reliable assistants, in particular if we want to install mobile robots in private spaces as homes.

As a further work we would like to test Real-Time Publish Subscribe (RTPS) protocol used by the ROS2 communications design. In that manner we plan to evaluate in a quantitative way the network and CPU performance under similar conditions as well as to compare qualitatively the pros and cons of the DDS based solution proposed in the new ROS design[4].

[4]http://design.ros2.org/articles/ros_on_dds.html

Acknowledgment

The authors would like to thank the Spanish Ministry of Economy and Competitiveness for the partial support to this work under grant DPI2013-40534-R and to the Spanish National Institue of CyberSecurity (INCIBE) under grand Adenda21 ULE-INCIBE.

References

[1] Tamara Denning, Cynthia Matuszek, Karl Koscher, Joshua R. Smith, and Tadayoshi Kohno. A Spotlight on Security and Privacy Risks with Future Household Robots: Attacks and Lessons. In *Proceedings of the 11th International Conference on Ubiquitous Computing*, 2009.

[2] Conor Fitzgerald. Developing Baxter. In *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference*, pages 1–6. IEEE, 2013.

[3] Peter Huitsing, Rodrigo Chandia, Mauricio Papa, and Sujeet Shenoi. Attack taxonomies for the Modbus protocols. *International Journal of Critical Infrastructure Protection*, 1:37–44, 2008.

[4] Francisco Martín, José Mateos, Francisco Javier Lera, Pablo Bustos, and Vicente Matellán. A robotic platform for domestic applications. In *XV Workshop of Physical Agents*, 2014.

[5] Jarrod McClean, Christopher Stull, Charles Farrar, and David Mascareñas. A preliminary cyber-physical security assessment of the Robot Operating System (ROS). *SPIE Defense, Security, and Sensing*, 8741:874110, 2013.

[6] Santiago Morante, Juan G Victores, and Carlos Balaguer. Cryptobotics: Why robots need cyber safety. *Frontiers in Robotics and AI*, 2(23):1–4, 2015.

[7] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009.

[8] Radu Bogdan Rusu. Ros-robot operating system. *Tutorial Slides, November*, 1, 2010.

[9] Y. Sattarova Feruza and Tao-hoon Kim. IT security review: Privacy, protection, access control, assurance and system security. *International Journal of Multimedia and Ubiquitous Engineering*, 2(2):17–31, 2007.

[10] Gurpreet Singh. A study of encryption algorithms (rsa, des, 3des and aes) for information security. *International Journal of Computer Applications*, 67(19), 2013.

Influence of Noise Spatial Correlation Variations on GCC-based Binaural Speaker's Localization

J.M. Perez-Lorenzo, R. Viciana-Abad, F. Rivas, L. G. Perez and P. Reche-Lopez

***Abstract*—Traditionally, the performance of methods to determine the azimuth angles of speakers, such as those algorithms under the generalized cross-correlation framework, has been evaluated via characterizing the acoustic properties of the environment in terms of two parameters: reverberation time and signal-to-noise ratio. In this work, it is shown that in environments with spatial correlated noise these two parameters may not be enough to describe the performance of two of the most used algorithms, Cross Correlation and Phase Transform methods. The Phase Transform is usually considered the most robust algorithm in reverberant environments. However, the results obtained from an extensive number of simulations, based on varying the absorption and the scattering coefficients of a room under different signal-to-noise ratios, indicate that the Phase Transform weighting is more vulnerable to spatial correlated noise, which is the type of noise usually found in real life scenarios. Moreover, under low signal to noise ratio conditions, its performance when localizing a source of interest may get worse under conditions of lower reverberation times, contrary to its behaviour when the noise is not active. This behaviour is due to its dependence with the degree of spatial correlation of the environment noise, which traditionally has not been considered when evaluating the algorithms to use. Also, while maintaining fixed values of signal-to-noise ratio and reverberation time, the experiments show that the performance may depend on additional properties of the room in noisy environments. Thus, other acoustic properties should be considered along with the reverberation time and signal-to-noise ratio when analysing and comparing the Phase Transform method with other localization methods, as well as for designing an adaptive audio localization system under changing conditions in noisy environments.**

***Index Terms*—Binaural localization, Generalized Cross-Correlation, Phase Transform, Spatial correlated noise.**

I. Introduction

METHODS for localization of acoustic sources with two microphones are used in applications such as robotics, hands-free telephony, teleconferencing, voice-controlled systems, hearing aids or modelling of psychophysical studies, being the research of binaural models of computational auditory scene analysis a growing field [1]. As stated in [2], the stage of sound localization is probably the most important low-level auditory function in applications such as robotics, where the system should be able to understand sound sources with permanent background noises, intermittent acoustic events, reverberation and so on. This challenging environment may be dynamic, due to effects such as sound sources changing their acoustic properties. Once the sounds are localized, they can be separated so as to provide clean speech signals, noise signals, etc. Also, an acoustic map of the robot's surroundings can be used as input to higher layers in order to merge the information with visual features [3], [4] or to gaze towards the source of interest, interact with a human speaker and other related issues. In this context, an important problem is the influence of noise on speaker-localization performance and, being the robust localization of speakers an important building block, despite decades of research, the task of robustly determining the position of active speakers on adverse acoustic scenarios has remained a major problem for machines [1].

The authors are with M^{2}P Research Group, University of Jaén.

E-mail: jmperez@ujaen.es

Methods based on the *time difference of arrival* (TDOA) have been widely used with the goal of detecting audio sources, and they rely on the relative delays between the signals arriving at two microphones. The Generalized Cross-Correlation (GCC) framework was first introduced in [5], and it has become the most common approach for binaural TDOA estimations [2]. Within this framework, TDOA is estimated as the delay that, applied to one signal, maximizes the cross-correlation function between the two-microphone signals. This cross-correlation is weighted with a specific function, leading to the different methods within the framework. Depending on this weighting function, each method presents different properties. The *Phase Transform* (PHAT) weighting is often used in reverberant environments due to its high performance, and also employed as a ground-truth method for performance evaluation of localizations methods. Also, it has been extended recently to the localization of multiple sources [6], [7], showing that it can be used even in cases where there is no previous information about the numbers of active sound sources.

When studying the performance of the GCC framework, the noise is considered in most of the cases additive and uncorrelated (or diffuse) at both microphones, being the type of noise assumed by the framework. However, these are typical features of the noise used to model internal/thermal noise of the measurements devices, which is minimized in nowadays technology. In contrast, the noise of real scenarios is due to external noise sources such as those related to machinery, air conditioners, electric motors, etc. Although it is known that the acoustic properties of a room may modify the spatial correlation of external noises, the evaluation of localization methods usually considers only the reverberation time parameter and the signal-to-noise ratio, without concerning about other ones. In [8] the performance of methods within the GCC framework were evaluated in four different real scenarios with stationary white noise. The study probed that these methods exhibit different performances in scenarios featured with similar reverberation times and signal-to-noise ratios due to the differences in the spatial correlation of

the noise captured at both microphones. The fact that the ability to localize speakers is strongly influenced by the spatial diffuseness of the interfering noise, and that the impact on this ability does not only depend on the overall signal to noise ratio and reverberation time, has been later corroborated in [1]. In order to show that the presence of a compact noise source is a challenge for correlation-based approaches, some interfering sources such as factory noise or speech-shaped noise were simulated in experiments for the localization of multiple speakers, and the diffuseness of the noise was varied by controlling the number of active noisy sources in the simulation. Also, it is mentioned in [1] that the vast majority of previous studies have investigated the influence of only diffuse noise on sound-source localization, which complies with the assumption of the GCC framework (although not necessarily realistic for a life scenario), being the work in [8], [1] some of the exceptions.

Being this lack of studies the main motivation of the present work, a set of numerical simulations has been systematically carried out via changing both the absorbing and scattering wall properties in an environment with a stationary noise source. In this way, based on a single configuration for the relative locations among the microphones and the audio and noise sources, both Cross Correlation and Phase Transform algorithms have been checked in seventy-two different scenarios and under two different signal-to-noise ratios. Although methods of GCC framework have been widely employed for sound source localization purposes, the influence of the absorbing/scattering properties of noisy environments has not been systematically addressed in terms of performance degradation due to the misassumption of noise spatial uncorrelation. The main difference of the present work respect to [8] is that many more scenarios have been tested by using the simulator, giving thus the possibility of a more generalized deduction. On the other hand, while in [1] a single scenario has been used with a fixed reverberation time, and the number of noisy sources is changed to control the noise spatial correlation, in the present work, an unique setting of sources and microphones has been used under different acoustic environments, which leads to different combinations of reverberation times and noise correlations. Based on the results, some conclusions can be deduced on how the performance of the methods are affected by the different properties of the rooms. This knowledge can be useful in the design of an adaptive system that tries to minimize the errors committed in noisy and complex environments.

The rest of this paper is organized as follows. Section II briefly presents the GCC framework. In Section III the methodology followed in the systematic evaluation is described. Section IV presents a discussion of the relation among the scenario and noise properties and the performance of the acoustic localization task. Finally, conclusions and future work are described in Section V.

II. GCC Framework

The GCC framework is based on a propagation model with added noise described as [9]:

$$x_1(t) = r_{(\mathrm{s,m1})}(t) * s(t) + n_{\mathrm{m1}}(t), \tag{1}$$

$$x_2(t) = r_{(\mathrm{s,m2})}(t) * s(t) + n_{\mathrm{m2}}(t) \tag{2}$$

being $x_1(t)$ and $x_2(t)$ the signals at both microphones, $s(t)$ the acoustic signal emitted by the source, $*$ the time convolution operator, $r_{(\mathrm{s,m1})}(t)$ and $r_{(\mathrm{s,m2})}(t)$ the room impulse responses at their corresponding source/microphone positions, $n_{m1}(t)$ and $n_{m2}(t)$ the noise at both microphones. As stated in Section I, a more realistic model consists of considering the noise as originated from external sources. In this case, (1) and (2) could be rewritten as:

$$x_1(t) = r_{(\mathrm{s,m1})}(t) * s(t) + r_{(\mathrm{n,m1})}(t) * n(t), \tag{3}$$

$$x_2(t) = r_{(\mathrm{s,m2})}(t) * s(t) + r_{(\mathrm{n,m2})}(t) * n(t), \tag{4}$$

being $r_{(\mathrm{n,m1})}(t)$ and $r_{(\mathrm{n,m2})}(t)$ the room impulse responses at both microphones respect to the noise source, and $n(t)$ the signal emitted by the noise source. In this second model, the acoustic properties of the room may not only influence the signal of interest to be detected at both microphones, but also the noise captured in the microphones.

The *time difference of arrival* or TDOA is implicitly found in the difference of the room impulse responses and it can be estimated as [5]:

$$\hat{\tau} = \arg\max_{\tau}\{\varphi^{g}_{x_1x_2}(\tau)\} \tag{5}$$

being

$$\varphi^{g}_{x_1x_2}(\tau) = \int_{-\infty}^{\infty} \psi_g(f) \cdot \Phi_{x_1x_2}(f) e^{j2\pi f\tau} df \tag{6}$$

the cross-correlation function between the signals $x_1(t)$ and $x_2(t)$. $\Phi_{x_1x_2}$ represents the cross power spectral density of the signals and $\psi_g(f)$ is known as the *generalized frequency weighting* [5]. The differences among the methods within this framework is found in this frequency weighting. The simpler one is known as CC (*Cross-Correlation*), where the weighting is the unity:

$$\psi_{CC}(f) = 1 \tag{7}$$

Among the other methods, the most well-known is the PHAT (*Phase Transform*) due to its improvement in performance compared to CC method in reverberant and noisy environments. PHAT is chosen in this study for being the most widely used as well as a reference to compare the performance of other methods. PHAT weighting only takes the phase information of the cross-correlation by weighting each frequency component with the inverse of its magnitude [5]:

$$\psi_{PHAT}(f) = \frac{1}{|\Phi_{x_1x_2}(f)|} \tag{8}$$

III. Methodology

In this section the methodology of the experiments is described along with the parameters used to assess the performance of the localization methods under different experimental conditions. GCC framework assumes a spatially uncorrelated white noise but, in the case of considering external noises, the correlation of the noise captured at both microphones will depend on the properties of the scenario.

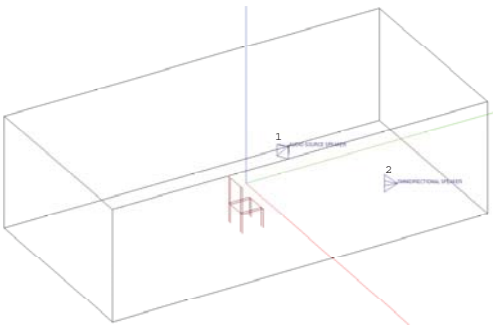


Fig. 1. Snapshot from EASE software. A room with size (10 x 5 x 3) m has been simulated. The software allows to change both absorption and scattering coefficients of the room. The chairs symbols are used to show the coordinates where the impulse responses of the room are computed (i.e. to simulate the microphones). The loudspeakers symbols are used to locate the audio sources. The omnidirectional loudspeaker (2) is placed with an azimuth angle of 120° respect to the middle of the microphones, while the directional one (1) is placed at 60°.

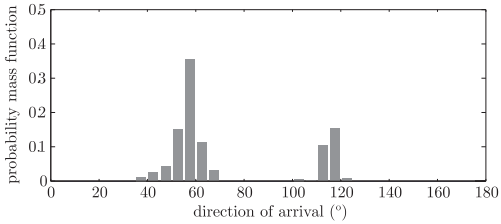


Fig. 2. Example of a probability mass function for a given scenario. The voice sound is placed at 60° and the noise source at 120°. The presence of the noise source in the detections is measured through the Noise Score, with a value of 0.27. The mean error of voice source angle estimations, that is to say the Mean Absolute Error, is 12.37°.

Typical scenarios used in acoustic localization methods are only characterized with the *reverberation time* (RT) parameter and the *signal-to-noise ratio* (SNR), and there is a lack in the literature of studies that evaluate the influence on localization performance of other acoustical parameters.

In order to evaluate the influence of the acoustic properties of the scenario on the spatial correlation of the noise acquired in the two microphones and, in turn, its influence in the performance degradation of the methods, the absorption and scattering coefficients of a simulated shoe-box shaped room with size (10 x 5 x 3) m have been modified to obtain different scenarios using EASE© ray-tracing software [10] (see Fig. 1). The absorption coefficient value has been set within a range from 0.1 to 0.8 with a step of 0.1 and the scattering coefficient value within a range from 0.1 to 0.9 with a step of 0.1, obtaining 72 scenarios. A pair of microphones has been placed in the middle of the room with a separation of 8 cm. The sound source whose direction of arrival must be estimated has been placed at an angle of 60° referred to a vertical plane containing the two microphones (azimuth angle) and at a distance of 2 m. This angle has been set as a middle value between 90° (where the TDOA would be zero) and 30°, because inaccuracies appear due to the non linear relation between the TDOA and the estimated angle for values under 30° [11]. The noise source has been simulated with an omnidirectional speaker located at 120° with respect to the microphones and at a distance of 3.5 m. Both speech and noise sources have been configured with the same elevation of the microphones. These sources have been placed with a wide separation between them to try to isolate the noise influence in a room with good acoustic conditions.

Through the simulations, the impulse responses $r_{(\mathrm{s,m1})}(t)$, $r_{(\mathrm{s,m2})}(t)$, $r_{(\mathrm{n,m1})}(t)$, $r_{(\mathrm{n,m2})}(t)$ defined in 3 and 4 have been obtained for each scenario. These impulse responses also include the directivity properties of the acoustic sources. For the audio signal $s(t)$, an anechoic male voice of the *First Stereo Audio Source Separation Evaluation Campaign* [12] has been used and, for the noise signal $n(t)$, a Gaussian white noise has been generated with a pseudo-random sequence. Signals $x_1(t)$ and $x_2(t)$ at each microphone have been low-filtered at 5 kHz before the cross-correlation computation and different SNR, measured after the low-pass filtering, have been obtained by controlling the levels of $s(t)$ and $n(t)$.

Once estimated the time difference of arrival $\hat{\tau}$ with (5), from the signals of two microphones separated a distance d, and being this separation significantly lower than the distance to the source, the azimuth angle θ where the source is located is easily computed as:

$$\theta = \arccos \frac{c \cdot \hat{\tau}}{d}, \theta \in [0°, 180°] \tag{9}$$

being c = 343 m/s the speed of sound. A source placed in the normal direction produces $\hat{\tau} = 0$, which corresponds with $\theta = 90°$. In order to analyse the data of the simulations, overlapped Hann windows of 25 ms. have been used to obtain the direction of arrivals, and a histogram has been built by accumulating the measured values θ for each scenario. These histograms can be used to estimate the most likely source position in a voice source localization problem [9], [13], [14]. Based on them, a probability mass function $f_\Theta(\theta)$ for each scenario (see Fig. 2) has been estimated as:

$$f_\Theta(\theta_i) = Pr(\theta = \theta_i), \quad 0 < i < N-1 \tag{10}$$

$$\theta_i = (i{+}0.5)\frac{180°}{N} \tag{11}$$

being N the number of bins of the histograms. N has been set to 36, next to the square root of the number of estimations, in order to obtain smooth histograms without losing relevant information [15].

For the purpose of measuring the performance in the task of detecting the source of interest while rejecting the noise source, two parameters have been defined to analyse the histograms: *mean absolute error* (MAE) and *noise score* (NS). The mean absolute error is the mean error of the angle estimations respect to the actual sound source angle:

$$\mathrm{MAE} = \sum_{i=0}^{N-1} \mid \theta_i - \theta_s \mid f_\Theta(\theta_i) \tag{12}$$

being θ_s = 60°, the localization of the source of interest.

In addition, the parameter noise score is proposed to show the influence of the noise source in the source angles estimations and has been defined as:

$$\text{NS} = \Pr(\theta_n - \Delta\theta < \theta < \theta_n + \Delta\theta) \tag{13}$$

being θ_n the angle where the noise source is placed, i.e. 120°, and $\Delta\theta$ has been set to 15° in the experiments. Fig. 2 shows an example of the probability mass function $f_\Theta(\theta)$ for a given scenario together with the values of the mentioned parameters. The values of NS and MAE for an ideal scenario should be zero.

IV. Results and Discussion

In this section the results are analysed attending to the influence of the noise correlation on the localization task. Then, the influence of the reverberation time is analysed along with the rest of properties of the scenarios. Finally, an overall comparison between PHAT and CC weighting is presented according to the mean absolute error of the detections.

A. Influence of spatially correlated noise

The basis of the analysis of the simulations is to study the influence of the level of noise correlation at both microphones due to the presence of the noise source $n(t)$ in the azimuth estimations of the source of interest. Thus, the spatial correlation ρ was measured by computing the maximum value of the cross-correlation of the energy normalized signals received at both microphones, when only the noise source is active:

$$\rho = \max\{E[n_{m1}(t+\tau) \cdot n_{m2}(t)]\} \tag{14}$$

Fig. 3 shows these correlation values for all the possible experimental conditions, that is to say, for all the combinations of scattering and absorption coefficients of the simulated scenarios. The maximum value of spatial correlation ($\rho = 0.836$) was found for the highest absorption value and the lowest scattering coefficient. In line with this, the scenario with lowest absorption and highest scattering was associated with the minimum value of spatial correlation ($\rho = 0.1$), showing a nearly uncorrelated noise at the microphones.

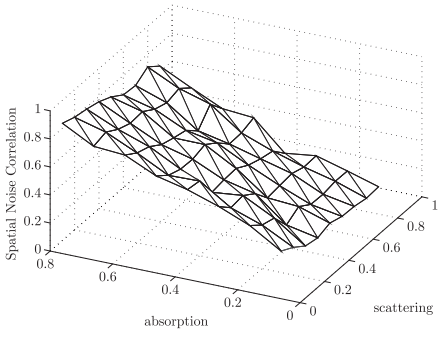


Fig. 3. Spatial correlation of the noise captured at both microphones in each scenario.

The influence of the noise spatial correlation due to the presence of the noise source was evaluated by analysing the noise score NS defined in (13), and depicted in Fig. 4. The NS was computed for the distributions of the estimated direction of arrival values for CC and PHAT methods considering two values of SNR, 20 dB and 0 dB. Under the condition of SNR = 20 dB, the noise source is hardly present in the source angle estimations as can be seen in Fig. 4a - 4b. In the case of CC method, the lowest value of NS is zero and the highest value is 0.136, being still significantly low. For the PHAT method, the lowest value of NS is 0.016 and the highest is 0.146, which also indicate a low effect of the noise in the estimations. Thus, it can be concluded that with a SNR of 20 dB the performance of the angle estimations is hardly affected by the noise source for all the combinations of absorption and scattering coefficients.

As expected, the performance analysis made for SNR = 0 dB showed a significant increase of the noise score for both methods (see Fig. 4c - 4d). Thus, the lowest (NS_{CC}=0.1; NS_{PHAT}=0.13) and highest (NS_{CC}=0.53; NS_{PHAT}=0.947) values of NS considerably increased for both methods due to the influence of the noise. However, this study also highlighted differences in performance between CC and PHAT method. In those scenarios with high absorption and low scattering coefficients, the negative influence of the noise was much more evident for PHAT method. Indeed, the high values obtained for PHAT method are very close to ideal histograms where the source of interest would be the noise source. As can be seen in Fig. 3, the conditions of high absorption and low scattering were related with the highest spatial correlation of the noise captured at both microphones. Thus, scenarios with a high spatial correlation of noise showed a more evident increase in the noise score of PHAT method. Also, when the spatial correlation decreases, the noise score gets lower, obtaining the lowest values in the scenarios where the spatial correlation of the noise is minimum (see Fig. 3 and Fig. 4d). To strengthen this point a Spearman's rank correlation has been computed between the score noise for the PHAT weighting and the spatial correlation of noise, obtaining a value of $r = 0.928$ and $p < 10^{-30}$, being the alternative hypothesis that the correlation is not zero. Therefore, it can be concluded that the correlation of the noise at both microphones is strongly related with the presence of the noise in the estimations of the DOAS when using the PHAT weighting under low signal-to-noise ratios. In environments with a highly spatial correlated noise, the noise source is expected to be detected as a source of interest using the PHAT weighting, while the simpler CC method seems to be less sensible to the presence of this noise source. Moreover, the presence of the noise in the CC estimations seems not to depend on its spatial correlation at both microphones.

B. Dependence of the PHAT weighting performance on the reverberation time and noise

In this section, the performance of PHAT algorithm is analysed attending to the mean absolute error of the localization task. Fig. 5a - 5b show the mean absolute error on the localization of the source of interest for SNR of 20 dB and

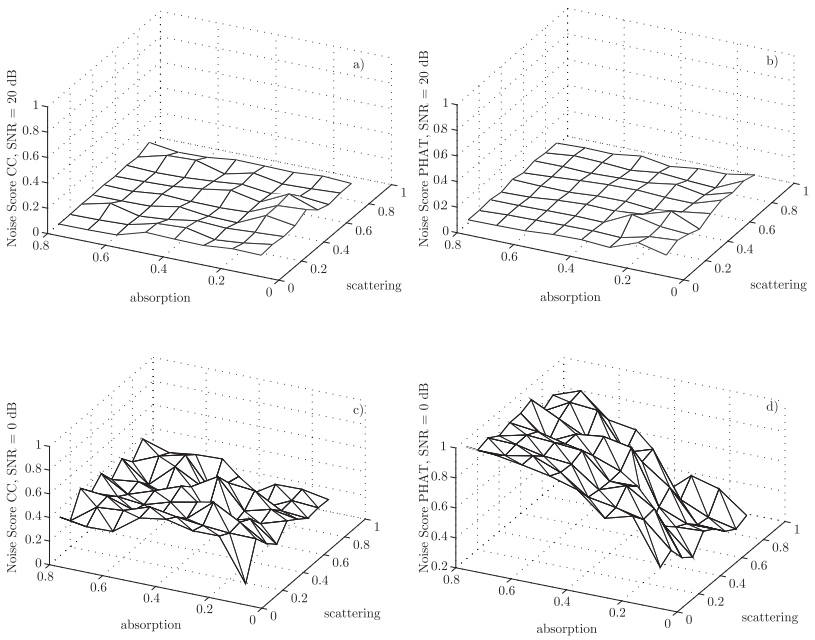


Fig. 4. Noise Score for each scenario. (a) CC weighting with SNR = 20 dB (b) PHAT weighting with SNR = 20 dB c) CC weighting with SNR = 0 dB d) PHAT weighting with SNR = 0 dB

0 dB, respectively. In order to analyse the relation between the MAE and the reverberation time of the rooms, this last value has been computed for all the simulated scenarios by Schroeder backward integration [16], with values ranged from 0.09 s to 1.26 s (see Fig. 6).

Under a SNR of 20 dB, the MAE is close to zero in most of the scenarios with the exception of those with the lowest absorption coefficients, related to the highest reverberation times. In these conditions, the maximum value of the MAE is 12.8°. In order to know the relation between MAE and RT, a Spearman's rank correlation has been computed between the two parameters for all the scenarios, obtaining a value of $r = 0.598$ and $p < 10^{-7}$. Therefore, the performance of the localization is worse in terms of mean absolute error when the reverberation time increases, being the expected behaviour as explained in the literature.

However, the relation between MAE and RT was different for low values of SNR (i.e. 0 dB). The maximum value of the MAE increases to 54.3°, being the environments with a higher absorption and lower scattering coefficients those with the greatest mean error in the estimations. In this case, the values obtained for the Spearman's correlation are $r = -0.797$ and $p < 10^{-16}$. These values show that, under a low signal-to-noise ratio, the localization task seems to be worse in those rooms with lower reverberation times. This effect is explained by the high influence of the correlated noise on the PHAT method explained in Sec. IV-A, as the noise source is detected as the source of interest when the spatial correlation increases. In fact, if the Spearman's correlation is computed between the mean absolute error and the spatial correlation of the noise at both microphones for all the scenarios, the value obtained is $r = 0.875$ with $p < 10^{-30}$. The value indicates that the performance is worse under high correlated noise and it improves when the spatial correlation gets lower, as the noise acts more like an additive and independent noise at both microphones and it is closer to the type of noise expected by the PHAT method. It can be concluded that, while the reverberation time is a good parameter to know the variation of the performance of PHAT algorithm in high signal-to-noise ratios situations, other parameters related to spatial correlation should be taken into account in order to predict the performance in those scenarios or time intervals when the signal-to-noise ratio is low.

Also, it can be seen that if only the reverberation time and the signal-to-noise ratios are taken into account, the performance of the PHAT method may not be uniquely defined. In Table I the performance is presented in terms of the MAE for some of the simulated scenarios where the time reverberation is similar, but with differences in the absorption/scattering parameters, as well as in the noise spatial correlation. For scenarios with similar reverberation time, the mean error may be different. Also, it can be seen that the difference of the performance varies from one situation to other. For RT = 0.15 s, the variation in the mean error represents a 10% respect to the minimum value of both errors. However, in the case of RT = 0.53 s the variation represents the 80% of the minimum error. So, not taking into account the spatial diffuseness properties of a scenario when studying the performance of the PHAT

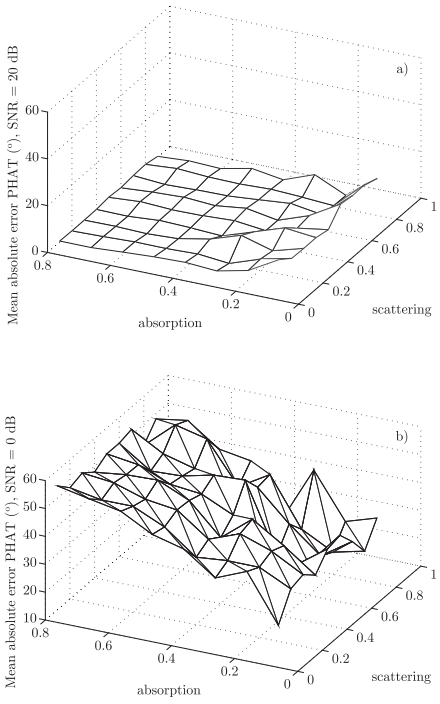


Fig. 5. Mean Absolute Error for PHAT weighting. (a) SNR = 20 dB (b) SNR = 0 dB

method could lead to inadequate expectations when dealing with low signal-to-noise ratios.

C. PHAT MAE vs CC MAE under correlated noise

Next, CC and PHAT weightings are compared according to the mean absolute error committed in the simulated scenarios. When including spatially correlated noise in the comparison of the PHAT weighting with other algorithms as the simple case of CC, the results differ respect to the situation of just using an additive and independent noise at both microphones. Traditionally, the inclusion of the PHAT weighting has been

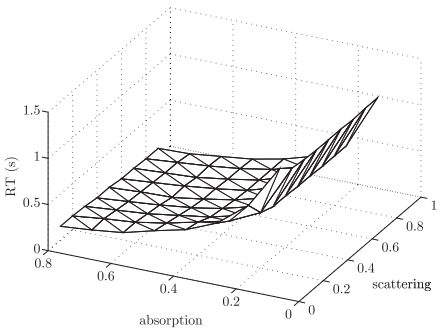


Fig. 6. Reverberation Time of the scenarios.

RT (s)	Absorption	Scattering	Noise Corr. ρ	PHAT Error
0.15	0.8	0.1	0.835	54.3
0.15	0.6	0.8	0.536	49.3
0.25	0.5	0.3	0.496	45.1
0.25	0.4	0.9	0.203	21.5
0.35	0.4	0.2	0.533	34.2
0.35	0.3	0.8	0.226	23.9
0.53	0.3	0.1	0.473	32.8
0.53	0.2	0.8	0.170	18.0

TABLE I
PERFORMANCE OF PHAT METHOD FOR SCENARIOS UNDER SNR = 0 dB WITH SIMILAR RT AND DIFFERENT ABSORPTION/SCATTERING VALUES

used to obtain a more robust detection in terms of reverberation and presence of additive noise. However, as it has been presented in Sec. IV-A, the presence of the noise when the SNR drops to 0 dB can affect seriously the estimations of the directions of arrival, and even the noise source can be detected as the source of interest. Fig. 7 represents the performance in all the simulated scenarios under a signal-to-noise ratio of 20 dB and 0 dB, where each point is the comparison in terms of the mean average error for one single scenario. Under high signal-to-noise ratios, the PHAT algorithm overcomes the simpler CC one in almost all the situations as expected. Only in a few situations both methods exhibit a similar performance or the CC weighting presents a slightly better performance, but it occurs in scenarios where both mean errors are close to zero. However, for the signal-to-noise ratio of 0 dB it can be seen that the comparison between both methods changes dramatically, presenting the PHAT weighting a greater mean error in most of the rooms. This may be attributed to the whitening process in the PHAT weighting as explained in [1]. In this whitening process, the frequency components are equally weighted, so the noise components are amplified. And as the noise gets more directional, the noise energy becomes a detected source by the azimuth estimations. Only in some scenarios where the noise exhibits low spatial correlation, the PHAT weighting presents a better performance than the CC one. Again, this issue should be considered when using the PHAT method in those noisy scenarios or time intervals with low signal-to-noise ratios under the presence of a non-diffuse noise source. This knowledge can be used by a system that, based on the detection of noisy sources and the monitoring of its spatial correlation, would be able to select the framework weighting in order to adapt itself to the environment.

V. CONCLUSIONS AND FUTURE WORK

In this work, the influence of the spatial correlation of an external noise source on the performance of PHAT and CC methods has been analysed through systematic simulations of scenarios with different signal-to-noise ratios and absorbing/scattering properties that lead to different values on the spatial correlation of the noise captured at both microphones. While PHAT weighting overcomes the CC one under high SNRs, the results indicate that PHAT weighting is more vulnerable to the correlated noise under low SNRs, leading to the detection of the noise source as the main source of interest. Also, the spatial correlation has exhibited more influence on the localization task performance than the reverberation time,

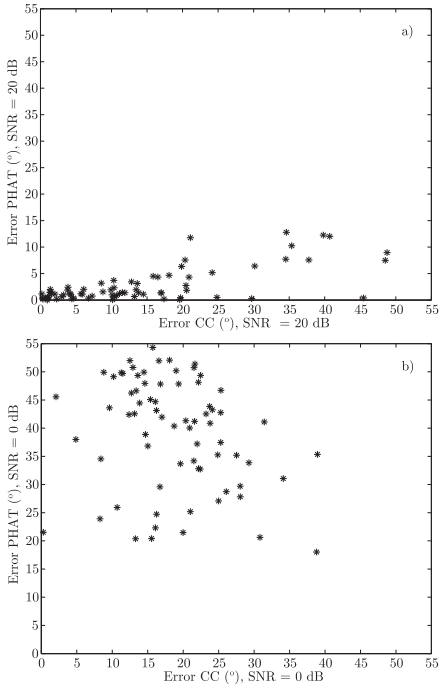


Fig. 7. Comparison of the MAE of PHAT and CC weighting. Each point represents a single scenario. (a) SNR = 20 dB (b) SNR = 0 dB

which has been one of the most traditional acoustic parameter to characterize the environments. This influence makes that even in situations under low signal to noise ratios, the performance of PHAT weighting can be worse in environments with lower reverberation times, contrary to the behaviour under high signal to noise ratios. In contrast, the behaviour of the CC algorithm respect to the influence of the noise source in the estimations does not present this dependence on its spatial correlation.

It has been shown that the performance of the PHAT weighting may not be uniquely defined by the RT and SNR values. Then, future work will include the use of additional acoustics parameters related with the spatial correlation, such as direct-to-reverberant ratio or clarity, to monitor the environment conditions. Based on the knowledge of the surroundings through these parameters, an adaptive system could be designed in order to minimize the errors provoked by noisy conditions. Also, knowing the influence of additional parameters on the performance of methods for direction of arrival may allow to focus on the improvements of traditional methods in a wide range of scenarios. Finally, another line of research may include an analysis of how biological systems might be dealing with these additional parameters, which could also be of interest in the design of bio-inspired binaural adaptive systems.

ACKNOWLEDGEMENTS

This work has been supported by the University of Jaén and Caja Rural de Jaén (Spain) under Project No. UJA2013/08/44 and by Economy and Competitiveness Department of the Spanish Government and European Regional Development Fund under the project TIN2015-65686-C5-2-R (MINECO/FEDER, UE).

REFERENCES

[1] May T., van de Par S., Kohlrausch A. Binaural Localization and Detection of Speakers in Complex Acoustic Scenes, in The Technology of Binaural Listening, Ch. 15, 397–425, Springer, 2013.
[2] Argentieri S., Portello A.,Bernard M.,Danes P.,Gas B. Binaural Systems in Robotics, in The Technology of Binaural Listening, Ch. 9, 225–254, Springer, 2013.
[3] Ferreira J.F., Lobo J., Bessière P., Castelo-Branco M., Dias J. A Bayesian Framework for Active Artificial Perception. IEEE Transactions on Cybernetics, 2013; 43(2): 699–711.
[4] Viciana-Abad R., Marfil R., Perez-Lorenzo J.M., Bandera J.P., Romero-Garces A., Reche-Lopez P. Audio-Visual perception system for a humanoid robotic head. Sensors, 2014; 14(6): 9522–9545.
[5] Knapp C., Carter G. The generalized correlation method for estimation of time delay. IEEE Transactions on Acoustics, Speech and Signal Processing, 1976; 24(4): 320–327.
[6] Reche-Lopez P., Perez-Lorenzo J.M., Rivas F., Viciana-Abad R. An unsupervised kurtosis-guided split-EM algorithm for the azimuth localization of multiple sound sources in reverberant environments. In: 3rd Workshop on Recognition and Action for Scene Understanding (REACTS), 2015. pp. 29–42.
[7] Escolano J., Xiang N., Perez-Lorenzo J.M., Cobos M., Lopez J.J. A Bayesian direction-of-arrival model for an undetermined number of sources using a two-microphone array. Journal of Acoustical Society of America, 2014; 135(2): 742–753.
[8] Perez-Lorenzo J.M., Viciana-Abad R., Reche-Lopez P., Rivas F., Escolano J. Evaluation of generalized cross-correlation methods for direction of arrival estimation using two microphones in real environments. Applied Acoustics 2012; 73(8): 698–712
[9] Martin R., Heute U., Antweiler C. Acoustic Source Localization with Microphone Arrays, in Advances in Digital Speech Transmission. John Wiley & Sons, 2008.
[10] EASE (Enhanced Acoustic Simulator for Engineers) software, Ahnert Feistel Media Group, Berlin, Germany, http://ease.afmg.eu/
[11] Yu Y., Silverman H.F. An improved TDOA-based location estimation algorithm for large aperture microphone arrays. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '04), 2004; pp. 77–80.
[12] Vincent E., Sawada H., Bofill P., Makino S., Rosca J.P. First Stereo Audio Source Separation Evaluation Campaign: Data, Algorithms and Results. Int. Conf. on Independent Component Analysis and Signal Separation (ICA 2007), 2007.
[13] Cobos M., Lopez J. J., Martinez D. Two-microphone multi-speaker localization based on a Laplacian Mixture Model. Digital Signal Processing, 2011; 21(1): 66–76.
[14] Chen J., Benesty J., Huang Y. Time Delay Estimation in Room Acoustic Environments: An Overview. Eurasip Journal on Applied Signal Processing, 2006.
[15] Montgomery D.C., Runger G.C. Applied statistics and probability for engineers. John Wiley & Sons; 2011.
[16] Schroeder M.R. New method of measuring reverberation time. Journal of Acoustical Society of America 1965; 37(6):409–412.

Foveal vision mapping and multiscale segmentation within an AP SoC

M. González, A. Sánchez-Pedraza, R. Marfil and A. Bandera

***Abstract*—Many embedded systems need image processing applications that, fulfilling real-time performance, will also address restrictions of size, weight or power consumption. Furthermore, applications such as tracking or pattern recognition do not necessarily precise to maintain the same resolution across the whole image sensor. In fact, they must only keep it as high as possible in a small region, but covering a wide field of view. Foveal vision proposes to sense a large field of view at a spatially variant resolution. One small region, the fovea, is mapped at a high resolution while the rest of the image is captured at a lower resolution. This work proposes a hardware system for mapping an uniformly sampled sensor to a space-variant one. Furthermore, this mapping is intimately tied with a software-based, multiscale segmentation of the generated foveal images. The whole hardware/software architecture is designed and implemented to be embedded within an All Programmable System on Chip (AP SoC). Preliminary results show the flexibility of the data port for exchanging information between the mapping and segmentation parts of the architecture and the good performance rates of the mapping procedure. Experimental evaluation also demonstrates that the multiscale segmentation method provides results comparable to other more computationally expensive algorithms.**

***Index Terms*—foveal sensor, image processing, hardware/software codesign, AP SoC**

I. Introduction

ROBOTIC vision needs to solve a large variety of tasks that demand different parameters. In order to solve all of them using the same sensor, this must be adjustable to the specific task. However, it is also usual that some of these tasks run simultaneously. Thus, for instance, the module responsible of the navigation skill could need a large field of view (FoV), while the one in charge of recognizing an object could simultaneously need to capture this at a high resolution. Furthermore, this image acquisition and processing must be done quickly, as the robot hopes to be an agent able to communicate and interact with people fluently within a dynamic world. Although it is currently possible to acquire a video sequence from a high resolution sensor (up to 10 megapixels) in real-time using dedicated hardware, any algorithm that requires multiple frames will need a large off-chip memory access. This requirement will typically be the bottleneck of the framework [4].

The evolution of biological vision systems has reached a balance between what they can perceive and what they are capable of processing at all times quickly. Foveal vision reduces the data volume by sampling the FoV at a variant resolution. A reduced area of the FoV, the *fovea*, is captured at a high resolution. This will be the area of the FoV where, for instance, we must include the objects that we want to recognize. Surrounding this fovea, the rest of the FoV is captured at a lower resolution. In fact, resolution will typically be reduced as we move away of this fovea, as Figure 1 shows. Through fast movements of the eyes, we could have the impression that we have a detailed vision of all the world in front of us. But, this is only an illusion [21]. Kevin O'Regan describes this very graphically: the world *is its own memory* [18]. And the mechanism to access to this 'memory' is by moving the eyes.

M. González, A. Sánchez-Pedraza, R. Marfil and A. Bandera are with University of Málaga. E-mail: {martin, aspedraza, rebeca, ajbandera}@uma.es

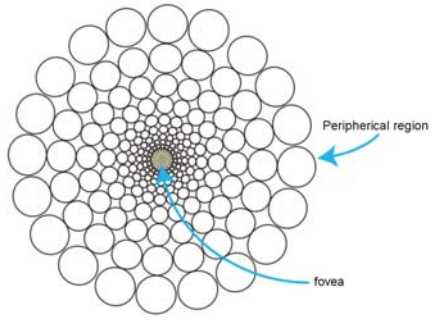


Fig. 1. Approximated distribution of cones on the human retina, providing *brain pixels* of adapted size distributed following a polar pattern (adapted from the original at C. Ware, 2008, p. 5 [21]).

Against this need of image detail, in which the eyes move by the scene depending on the task, it is usually necessary to keep a record of what is happening with the rest of the scene. As aforementioned, this could be demanding by a robot navigating within a dynamic environment. In this application, the main goal is to perceive a large FoV, perhaps to lower resolution, but with the enough one to be able to act if necessary. Opposite to central vision, peripheral vision is the responsible of finding, recognizing, and responding to information in different areas of the FoV around the object on which the attention is fixed. In the human vision system, the peripheral retina is especially sensitive to displacement, being its most characteristic function to detect motion. However, it has been also demonstrated its use during actions of reaching and catching. Thus, it seems to play an important role in the visuo-motor coordination, and also in the spatial posture and locomotion.

This paper revisits the research field on spatial-variant vision, proposing a whole framework that (i) maps the sensor data into a foveal lattice and (ii) proceeds this stream for providing a multiscale segmentation of the foveal images. Space-variant vision experimented its greatest popularity in

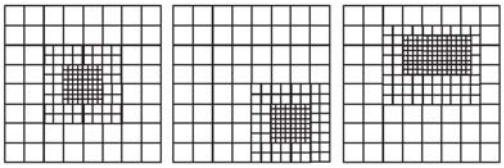


Fig. 2. (left) CFG with m = 3 and d = 2; (center) GMFD of generalized motion with s_h = { 0,2 } and s_v ={ 0,2 }; and (right) GMFD of adaptive motion with L_d = 2, R_d = 1, T_d = 1 and B_d = 3.

the nineties. Intimately tied to the concept of *active perception* (*it does not make sense to have a fovea if the eyes cannot be swiftly moved over possible regions of interest* [20]), it was the origin of several physical implementations in silicon. However, the design of an specific retina-like sensor was always a hard and expensive work. Hence, most of the proposals finally laid in a software-emulated foveal sensor, which took the input data from an uniform, real sensor [4]. Currently, this option is also supported by the fact that, mainly driven by its use on the smartphones, it is possible to acquire CMOS sensors of very high resolution at a very reduced prize. On the other hand, the processing limitations of software emulation can be overcome by performing the mapping through hardware emulation. This was a wide part of the research within our group in the past [3], [7], [8]. This work however tests the option of embedding the complete hardware and software solution on a All Programmable System on Chip (AP SoC) platform. Thus, the foveal mapping is synthesized on the FPGA (programmable logic) and the multiscale segmenter is programmed on an ARM (processing system). Data between memory and modules are exchanged through Direct Memory Access (DMA), allowing the whole framework to run at 10 frames per second (fps). The rest of the paper describes the proposed framework: Section II briefly reviews and describes the foveal lattice synthesized on the FPGA. An overview of the whole framework is present at Section III. Then, Sections IV and V present the design and implementation of the chain of processing on the logic part of the AP SoC, and the communication between these cores and the ones running on the ARM, respectively. Section VI briefly describes the hierarchical segmentation of the foveal images performed on the software part. Sections VII and VIII summarizes the experimental evaluation of the proposed framework and draws the main conclusion and future work, respectively.

II. Cartesian foveal geometries

Cartesian Foveal geometries (CFG) encode the field of view of the sensor as a square-shaped region at the highest resolution (the *fovea*), surrounded by a set of concentric rings with decreasing resolution [3]. In the majority of the Cartesian proposals, this fovea is centered on the geometry and the rings present the same parameters. Each ring is typically composed by several subrings. Thus, the geometry is characterized by two constant values: the number of rings surrounding the fovea (m), and the number of subrings (d) within each ring. The size of the fovea is stated according to d: if we set its dimensions to $4d \times 4d$, there will not be discontinuity between the fovea and the periphery regions. Figure 2(left) shows an example of a fovea-centered CFG. We refer to each cell on the lattice as a *rexel* (resolution cell).

Among other advantages, there are CFGs that are able to move the fovea through the FoV (shifted fovea). Vision systems that use the fovea-centered CFG require to place the region of interest in the center of the image. That is usually achieved by moving the cameras. A shiftable fovea can be very useful to avoid these camera movements. Figure 2(center) shows one example of the generalized algorithm [3]. Within this algorithm, each ring of resolution is shifted with respect to the center position according to two vectors (s_v and s_h). Furthermore, the adaptation of the fovea to the size of the region of interest can help to optimize the consumption of computational resources [3]. Figure 2(right) shows the rectangular structure of an adaptive fovea. The geometry is now characterized by the subdivision factors at each side of the fovea (L_d, R_d, T_d and B_d). This will be the geometry finally implemented within this work.

III. Overview of the framework

Continuous evolution of logical programmable devices has motivated the current availability of low-cost development boards such as the Zedboard from Avnet, which is based on the Zynq-7000 AP SoC XC7Z020-CLG484-1 from Xilinx. The Zynq-7000 combines software, hardware and input/output capability over a single silicon integrated circuit. This will be the hardware base for setting our proposal. Figure 3 shows an overview of the proposed framework.

Hardware configuration for the AP SoC can be stated as the interaction between two different parts: the processing system (PS) and the programmable logic (PL). The software processing is addressed on the PS, which is mainly composed by the ARM cores, I/O peripherals, clock system generators, and memory interfaces. In our case, foveal images are generated on the PL, while the initial configuration of the sensor, an OV5642 5 megapixels from Omnivision, and the multiresolution segmentation is conducted on the PS. The figure shows that, although this configuration is addressed from the PS, it accesses to the sensor via the PL. PS and PL parts are interconnected through the AXI Video Direct Memory Access (AXI VDMA) core. The AXI VDMA is a soft Xilinx IP core that provides high-bandwidth direct memory access between memory and AXI4-Stream type video target peripherals.

Figure 3 shows that there is one block synthesized in the PL: the Foveal image acquisition/Color conversion. which is in charge of obtaining the uniform RGB image from the sensor, mapping this image into a foveal lattice, and converting the RGB color values of the CFG rexels to HSV color space. As aforementioned, the sequence of foveal images are stored on the external memory (DDR3 SDRAM) through AXI VDMA, Interconnect and HP interfaces/ports. In Figure 3, cores involved on the connection between the PL and the PS parts are filled with a white background. From the DDR3, the multiresolution segmenter takes each frame as the base

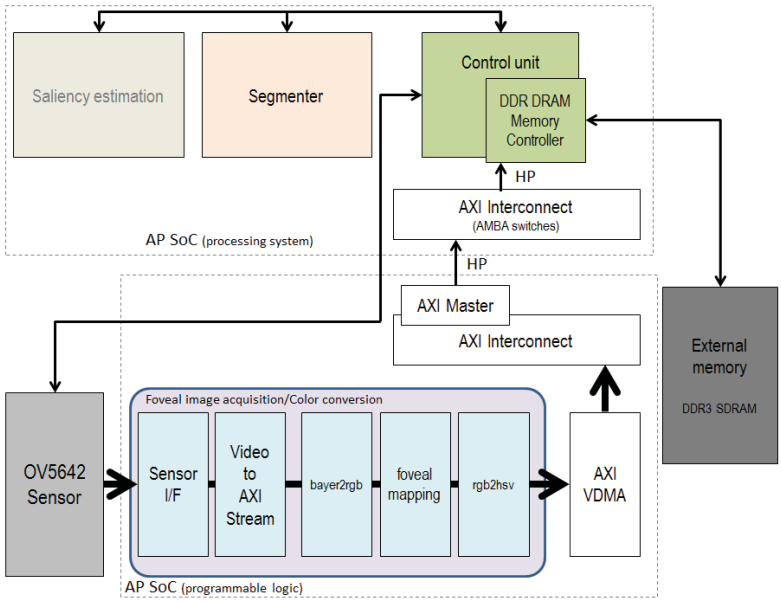


Fig. 3. Overview of the proposed framework.

level and generates the hierarchy of segmentation images. Each level on the hierarchy is encoded as an irregular pyramid. This will force to encode these levels as graphs. Finally, the overview also includes a Saliency estimation module. This module has been successfully implemented on the ARM, and will determine what the position of the new fovea will be. For achieving this, the module computes the parameters $\{L_d, R_d, T_d, B_d\}$, which will drive the next fovealization. But centered on the mapping and segmentation processing, this paper will not provide a description on how it works.

IV. Hardware-emulated foveal mapping

Figure 3 illustrates the stages involved on the data processing within the PL. The sensor employed (OV5642) is able to provide color frames of 5 megapixels. The Sensor I/F core is able to provide the video input, consisting of parallel video data, video syncs, blanks and data valid. For bridging the video input and the video processing cores, we use the Xilinx LogiCORE^{TM} IP Video In to AXI4-Stream core. This core interfaces our video source to the AXI4-Stream Video Protocol Interface. On one hand, this forces us to use this protocol for communicating the rest of IP cores. But, on the other hand, it simplifies the design, handling the asynchronous clock boundary crossing between video clock domain and the processing (AXI4-Stream) clock domain.

The foveal mapping and color conversion are addressed by the three last cores on the 'Foveal image acquisition/Color conversion' block. The bayes2rgb core aims to reconstruct a color image, at the full resolution, from the commonly-used Bayer color filter array pattern provided by the sensor. Specifically, it implements a linear approach, encoded on masks of 3×3 pixels [19]. It provides the input data to the foveal mapping. This core is in charge of generating all rexels of the foveal lattice.

For achieving this goal, we parallel the generation of the rexels of all sizes. The foveal mapping is then obtained by a sequence of 4-to-1 averaging stages, which will generate a complete sequence of images of reduced resolution. These images could define a pyramid, as Figure 4 shows, where the base level is the input image. It must be noted that all these images are obtained at the same rate imposed by the sensor, i.e. we obtain the last rexel at all images when the last pixel of the uniform frame is provided by the sensor. The 4-to-1 averaging process can be addressed in two steps. In the first one we obtain the average of the two pixels on the upper row. In the second step we average the two pixels on the lower row and compute the final value. The process is schematized at Figure 5. In the figure, $ldi_{i=0,1..}$ denotes a cell that must be registered; $ui_{i=0,1..}$ indicates the storing of the first averaging process; $pi_{i=0,1..}$ implies the recovery of the first averaging value required for obtaining the second averaging value; and $rdi_{i=0,1..}$ denotes the instant in which the 4-to-1 averaging process is complete. We have marked on the figure as gray cells these instants of interest. They are referred to the reading of the original image (i.e., the level i equal to 0). Figure 5(left) shows the generation of the first level. The pixel at row 0 and column 0 must be registered to be averaged with the next pixel on the same row. The result of this averaging process will be available at the column 1 of the row 0, and should be stored for completing the 4-to-1 averaging process with the two pixels at row 1. This same process (registering, processing and storing) will be replicated for the rest of pixels at row 0. Thus, we will

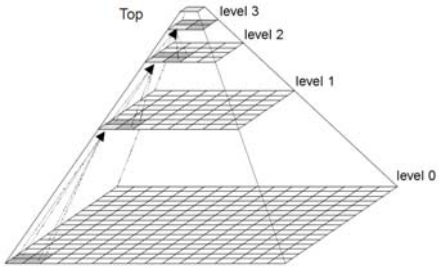


Fig. 4. Pyramid composed by images of reduced resolution. It includes all the rexels needed to generate any possible foveal layout.

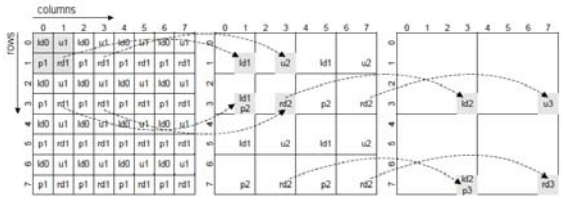


Fig. 5. Generating the sequence of images of reduced resolution.

need a buffer with a length equal to the half of the width of the original image. When we will process the row 1, the values on the buffer will be extracted in the same order that they will be stored. The required buffer is then a FIFO structure. The values of the rexels at level 1 are obtained when we proceed the pixels of the input image associated to all odd columns of the odd rows (labeled with $rd1$ on the figure). This defines a new data flow that is processed in the same way for generating the values associated to the level 2 and so on to higher levels. Figure 6 shows the datapath for generating the rexels of the three first levels. It should be noted that the structures should be triplicated for dealing with the three channels of the RGB color encoding.

Finally, the function of the rgb2hsv core is to translate the RGB values of the rexels to the HSV color space. The stream of data resulting of this conversion should be stored in the external memory for being shared with the ARM cores. These cores will built the foveal lattice from the resolution levels provided by the PL and then will obtain the segmentation results.

V. Communicating the PL and PS cores

The proposed framework needs to handle the stream of foveal data from the PL to the DDR3 SDRAM memory at the highest possible rate. Direct memory access (DMA) will allow the rgb2hsv core to gain access to the main bus linking the processor with the DDR3 memory. This avoids the use of the ARM processor to perform load or store operations, giving the Zynq-7000 AP SoC a large I/O bandwidth and low latency in the integration of a custom logic with a custom software. Thus, once the DMA transfer has been set up by the ARM core, this will wait to be notified when a complete chunk of resolution levels is received. When a transfer is completed, the ARM core generates the foveal image and then the hierarchy of segmentation results while a new transfer is in progress. This mechanism saves CPU cycles and increases the data throughput.

In the system block at Figure 3, the generated foveal frame is transferred through AXI VDMA (Video Direct Memory Access). AXI is a standardized IP interface protocol based on the ARM AMBA4 and AMBA3 AXI specifications. The AXI VDMA core can be used to transfer AXI4-Stream protocol based video stream to DDR memory and vice versa. AXI VDMA implements a high-performance, video-optimized DMA engine with frame buffering, scatter gather, and 2-dimensional (2D) DMA features. AXI VDMA transfers video data streams to and from memory and operates under dynamic software control or static configuration modes. To provide high-speed AXI master interfaces in the PL with lower latency, connections to the high performance (HP) interfaces are required. AXI Interconnect and AXI HP ports on the Zynq-7000 AP SoC together implement a high-bandwidth and high-performance memory system for use in applications where multiple devices share a common memory controller. Briefly, the AXI VDMA is connected to a HP interface by means of the AXI Interconnect and is controlled by the Cortex-A9 processor.

Within the PS, the code is organized to run the same algorithm within two independent threads. The aim is to better exploit the power of the two cores of the Cortex-A9 processor. Furthermore, the execution of these two cores can concurrently work with the reception of a new foveal frame.

VI. Hierarchical segmentation of the foveal images

Segmentation is an important task in image processing that plays a chief role in understanding images. It can be defined as the process of decomposing an image into regions which are homogeneous according to some criteria. The segmentation algorithm must adapt the resulting regions to the contents of the image and can be also driven by the attention stage [16]. Pyramids are hierarchical structures which have been widely used in segmentation tasks [15]. Instead of performing image segmentation based on a single representation of the input image, a pyramid segmentation algorithm describes the contents of the image using multiple representations with decreasing resolution. Pyramid segmentation algorithms exhibit interesting properties with respect to segmentation algorithms based on a single representation [15]. The first attempts for developing pyramid-based approaches for image segmentation were based on setting in advance the size of all levels on the hierarchy. These regular pyramids were very efficient for solving the segmentation problem, but they were not able to correctly encode the image layout within such a rigid structure [14], [15]. In contrast to regular pyramids, irregular ones have variable data structures and decimation processes which dynamically adapt to the image layout. Thus, the size of each level and the height of the structure are unknown.

Irregular pyramids allow coarse-to-fine strategies by encoding a hierarchy of successively reduced graphs. Level l is

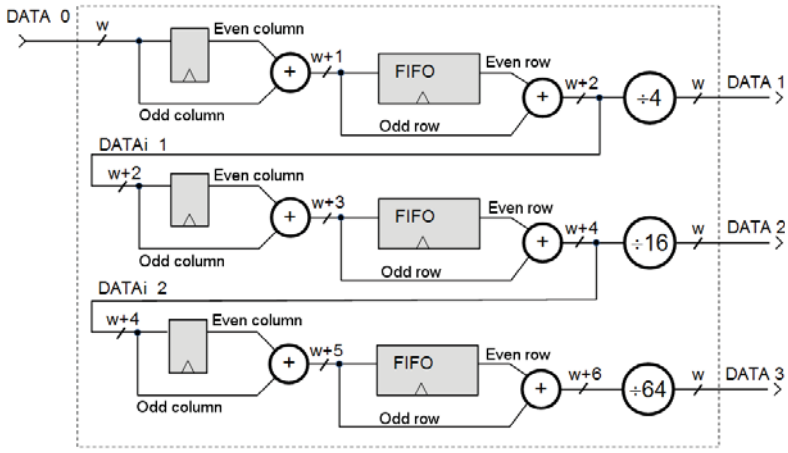


Fig. 6. Complete structure of the datapath for generating the first three levels using data from the input image (DATA 0).

represented by a graph $G_l = (V_l, E_l)$ consisting of vertexes $v \in V_l$ and edges $e \in E_l$. In this hierarchy, each graph G_{l+1} is built from G_l by selecting a subset of V_l. The selected vertexes are called *surviving vertexes*. Non-surviving vertexes of V_l are linked to surviving ones. Thus, each vertex v of G_{l+1} has associated a set of vertexes of G_l, the reduction window of v, which includes itself and all non-surviving vertexes linked to it. This is a decimation process which requires rules for:

- The selection of the vertexes V_{l+1} among V_l. These vertexes are the surviving vertexes of the decimation process.
- The allocation of each non-surviving vertex of level l to a survivor, which generates the son-parent edges (*inter-level edges*).
- The creation of edges E_{l+1} (*intra-level edges*) by defining the adjacency relationships among the surviving vertexes of level l.

The receptive field of one surviving vertex is defined by the transitive closure of the son–parent relationship and must be a connected set of vertexes in the base level. Rules for the definition of the set of surviving vertexes and the set of edges connecting each non-surviving vertex to its parent vary according to the considered decimation algorithm used within the irregular pyramid. Therefore, the reduction procedure used to build one graph from the one below strongly influences the efficiency of the pyramid. On the other hand, each level of the hierarchy is encoded by a graph and, since many graph algorithms suffer from a high computational complexity, the efficiency of the irregular pyramid is also influenced by the selected graph encoding. Among other relatively complex schemes, the data-driven decimation process (D3P) was proposed by J.M. Jolion [14] has a two-steps algorithm for choosing the surviving vertexes within an irregular structure based on simple graphs. For addressing this decimation process, the algorithm characterizes each vertex on the pyramid by the variance value v_i, estimated from the set of vertexes within its reduction window (they are set to 0 for the vertexes at base level). Then, the algorithm uses two additional binary-state variables for describing each vertex: p_i and q_i. In the first step, the local maxima according to v_i are labeled with p_i equal to 1 and q_i equal to 0. In the second step, the algorithm only evaluates the vertexes with q_i equal to 1: those ones that have a local maxima (p_i = 1) in its neighborhood are labeled with q_i equal to 0. Then, vertexes with p_i or q_i equal to 1 are the surviving ones (p_i = 1). The surviving vertexes define V_{l+1}. Then, a vertex v_i of G_l survives if and only if it is a local maximum or does not have any surviving vertex in its neighborhood. Figure 7 provides a simple example.

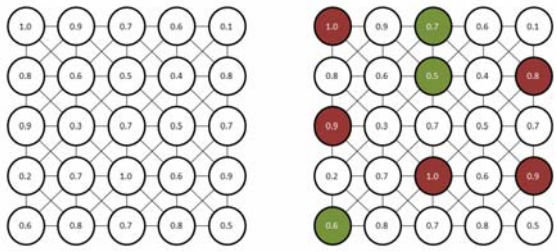


Fig. 7. (left) original graph -within each vertex we have annotated its v_i value-; and (right) surviving vertexes marked in red (local maxima) and green.

After obtaining the survivors, the inter-level edges at this new level, E_{l+1}, are obtained as

$$E_{l+1} = \{(i,j) \in V_{l+1} \times V_{l+1} : i \neq j \cap path(i,j) \leq 3\} \quad (1)$$

where

$$path(i,j) = 1 \Leftrightarrow \delta_{ij}^{(k)} \quad (2)$$

$$path(i,j) = 2 \Leftrightarrow \exists y \in V_k : \delta_{iy}^{(k)} \cap \delta_{yj}^{(k)} \cap y \notin V_{k+1} \quad (3)$$

$$path(i,j) = 3 \Leftrightarrow \exists y, x \in V_k : \delta_{iy}^{(k)} \cap \delta_{yx}^{(k)} \cap \delta_{xj}^{(k)} \cap y, x \notin V_{k+1} \quad (4)$$

$\delta_{ij}^{(k)}$ being equal to 1 if there exists an intra-level edge between vertexes i and j in level G_k. Inter-level edges, linking each

non-surviving vertex with a surviving one at level k + 1, are established by imposing that both vertexes must be neighbors at level k.

A. The Bounded D3P (BD3P)

The main advantage of the D3P is that, being very simple and fast, provides results comparable to other irregular approaches (see Section VII). However, there are two main problems related with its implementation. On one hand, it does not bound the number of neighbors of the vertexes. This makes impossible to bound the computational times and memory resources. On the other hand, the estimation of the $path(i, j)$ values on Eqs. (3-4) is a very hard problem, as it is needed to evaluate all possible paths in level k for obtaining E_{k+1}. This computation must be iterative, provoking that the whole algorithm will be slow.

This paper proposes to address both problems, alleviating the computational cost and memory resources needed to run the algorithm. This new version, the Bounded D3P (BD3P), takes into account the final task for reducing the number of edges on E_{k+1} and speeding up its computation. The first issue is addressed by removing from the neighborhood of vertex x those vertexes whose color is *very different* from the color of x. This qualitative assertion is implemented by a simple thresholding. Thus, the neighborhood N_x of a vertex x is now defined as

$$\{y \in N_x : \delta_{xy} \cap d(x, y) < t_c\} \tag{5}$$

where $d(x, y)$ defines the HSV color distance between vertexes x and y and t_C is a threshold value. The effect of this thresholding is illustrated on Figure 8. Figure 8(left) provides a naive example of the application of the D3P algorithm to a set of 11 vertexes. When the rules for obtaining the surviving vertexes are applied, the algorithm chooses three survivors. The color of one of them is light gray (red) and the color of the other two is dark gray (blue). When the non-surviving vertexes looks for a surviving one for establishing the intra-level edges, one of the blue vertexes (with a 0.6 value), *must link* to the red survivor, as this is the only surviving vertex within its neighborhood. This linking clearly disturbs the definition of the reduction window of the red surviving vertex (composed by four red vertexes and one blue vertex). Figure 8(right) shows the application of the BD3P to this same set of vertexes. Assuming that t_c avoids the linking of red and blue vertexes, the neighborhood of each vertex is now quite different. We have marked as bold lines this new set of intra-level edges on level k. There are again three surviving vertexes at level k + 1, but they are different from the previous case (the 0.6-valued vertex is substituted by the 0.8-valued one). All non-surviving vertexes must now find a survivor within its neighborhood, but intra-level edges at level k impose that linked vertexes are of the same color. Subsequently, the three reduction windows are now composed by vertexes of the same color.

With respect to the second issue, the removal of the computation of the $path(i, j)$ values on Eqs. (3-4) is achieved by determining that there will be an edge between two vertexes x and y at level k + 1 if the reduction windows of these two vertexes are in contact at level k (and if $d(x, y) < t_c$). Thus, E_{k+1} can be computed by a single evaluation of the vertexes at level k.

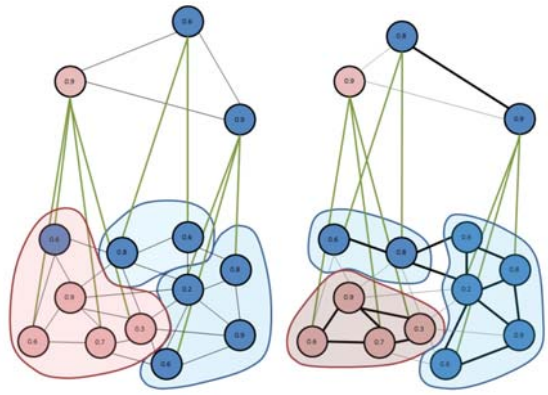


Fig. 8. Influence of the thresholding of intra-level edges within the internal structure of the D3P algorithm: (left) original D3P; and (right) BD3P (see text for details).

TABLE I
SYSTEM RESOURCES AND PERFORMANCE

	Image demosaicing	Foveal mapping	rgb2hsv
DSP	3	0	9
BRAM_18K	6	18	0
FlipFlop	500	1227	6953
LUT	543	1242	9226
Clock (ns)	9,4	5,4	8,16
Latency	5.043.391	13	43

VII. EXPERIMENTAL RESULTS

A. Performance and utilization of the PL cores

The hardware conforming the image acquisition, foveal mapping and color conversion were tested on the programmable logic of the Zedboard. Table I shows the system resource utilization for providing these functionalities. The latency of the image demosaicing takes 5.043.391 clock cycles. This is basically one clock cycle for processed pixel (the image size is 2592 $\times$ 1944 (5.038.848 pixels)). Thus, the time measured to process the whole image is 47,4 ms. But it can be noted that the delay with respect to the reception of the last pixel of the frame from the sensor is only 4.543 clock cycles. The resources for the foveal mapping allow the processing of the three color channels. It generates six video streams, associated to the whole image from the sensor (it will provide the fovea) and to five resolution levels (with rexels ranging from 2 $\times$ 2 to 32 $\times$ 32 pixels). The additional latency is only 13 clock cycles. The rgb2hsv core needs 43 additional clock cycles. Thus, the total delay with respect to the frame reception is 4.599 clock cycles (0,043 ms using a clock of 9,4 ns).

B. Evaluation of the BD3P

Table II assesses the BD3P with other decimation approaches using 50 color images from the Coil-100 database.

TABLE II
Q VALUE AND NUMBER OF LEVELS OF THE PYRAMID FOR SEVERAL DECIMATION ALGORITHMS (SEE TEXT)

	Q_{min}	Q_{typ}	Q_{max}	h_{min}	h_{typ}	h_{max}
LRP	1052,1	1570,3	2210,3	9	9	**9**
WRP	1133,7	1503,5	2080,8	9	9	**9**
D3P	355,6	**818,5**	1301,1	11	32,9	64
BIP	**343,2**	1090,9	1911,3	**8**	**8,7**	15
HIP	460,5	955,1	1530,7	9	11,4	19
CIP	430,7	870,2	**1283,7**	9	74,2	202
BD3P	412,6	831,5	1497,1	9	13	18

Images have been resized to 256 $\times$ 256. Among the approaches, there are regular ones, such as the Linked pyramid (LRP) [6] or the Probabilistic one (WRP) [13], and irregular approaches, such as the Data-driven decimation process (D3P) [14], the Bounded irregular pyramid (BIP) [15], the Hierarchy of partitions (HIP) [12] and the Combinatorial pyramid (CIP) [5]. All approaches have been tuned for obtaining the best score on the Q parameter, an estimator of the segmentation accuracy [15]

$$Q(I) = \frac{1}{1000 \cdot N \cdot M\sqrt{R}\sum_{i=1}^{R}[\frac{e_i^2}{1+logA_i} + (\frac{R(A_i)}{A_i})^2]} \quad (6)$$

being $M \times N$ the size of the image, and R the number of segmented regions. A_i and e_i are the area of the region i and its average color error, respectively. $R(A_i)$ being the number of segmented regions with area equal to A_i. Table II shows that the results provided by the BD3P are comparable to the ones obtained by the best approaches. However, it is able to run at less than 100 ms on the Cortex-A9 processor.

The BD3P algorithm has been also evaluated using the Precision-Recall metric and the database of natural images proposed by the group of Prof. Jitendra Malik on the University of Berkeley (BSDB500) [17], [2]. Our algorithm has two main parameters: the t_c value employed for thresholding the intra-level edges; and the maximum number of neighbors n_{max} for vertex. This second parameter has been set for providing a maximum bound to the number of neighbors. This Section analyzes how to choose correct values for these parameters and what results are obtained by using the BSDB500 data set[1].

On the other hand, it is important to determine how the segmentation scheme will be apply over a static image. That is, as we mentioned at Section I, the use of a foveal strategy does not make sense if the fovea cannot be swiftly moved over possible regions of interest [20]. For verifying the performance of the foveal segmenter, we will firstly obtain a segmentation result centering the fovea on the FoV. The saliency of the regions composing this segmentation image will be evaluated (we will use the color and intensity contrasts and orientation, as proposed by Marfil et al [16]) and we will choose the five most salient regions. Once these regions are chosen, we will segment the image by setting the fovea over each one of these regions. Figure 9 schematizes the procedure. The borders

[1]For foveal processing, the images on the BSDB500 dataset will be converted from 481 $\times$ 321 size to 480 $\times$ 320. The width and lenght of the FoV should be a multiple of the side of the maximum rexel on the lattice (in our case, 32 $\times$ 32 pixels).

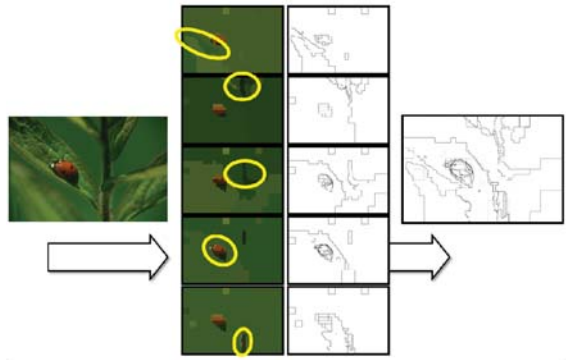


Fig. 9. Obtaining the contour map associated to an input image as a combination of five fovealizations.

within each segmentation image are obtained and labeled according to the rings where they are detected (strong borders will be associated to rexels of minor size). This provides us five sets of contours per input image. They are finally combined for providing the contour map that will be evaluated using the Precision-Recall metric.

1) Choosing the parameters: The BSDS500 provides a set of images for learning and a different set for testing. Using the first set, we have set the t_c value to 200 and the n_{max} value to 10. With respect to the sensibility of the approach to changes on the parameters, we can conclude that setting the n_{max} value to 10, the t_c value can range from 150 to 250 and the recall value typically remains over 0,9. When the t_c value is greater than 250, the recall value diminishes as some real contours are lost. When the t_c value is lower than 150, the number of vertexes on each level of the hierarchy grows, and with them the height of the pyramid and the space occupied by the structure on the system memory. Furthermore, the precision value decreases, as contours that are not present on the human segmentation arise. In fact, if the recall value is relatively easy of maintaining in high values, this will not be the same for the precision value. Figure 10 shows that the approach does not only provide the real contours, strongly marked when the fovea is close to the regions where they are, but also other contours, weakly marked on these same regions, which will be generated when these regions are perceived by the peripheral vision. These last contours make diminishing the precision value as, being for instance defined by rexels of 32 $\times$ 32 pixels, they are not really very precise. To compensate this effect we propose to label these contours by lower values (see Figure 10).

2) Evaluation on the BSDB500: Figure 11 allows to compare the performance of the proposed segmentation approach with respect to the ones provided by other approaches. Our approach is overcome by the gPb-owt-ucm [2] and the UCM [1], obtaining better results than other methods [10], [11], [9].

VIII. CONCLUSIONS AND FUTURE WORK

The basis of active vision lays on the existence of a mechanism able to detect what are the regions of interest

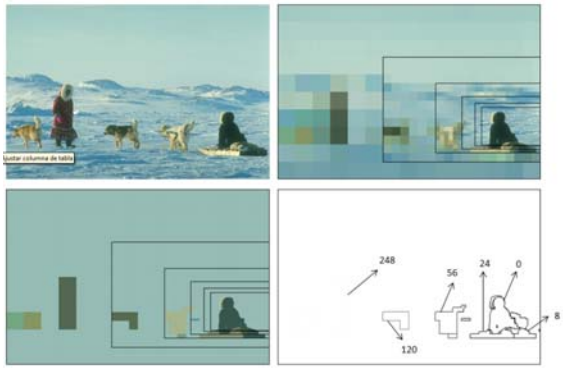


Fig. 10. (top) Image #310007 of the BSDB500 and one foveal image; and (bottom) segmentation image and associated contour map.

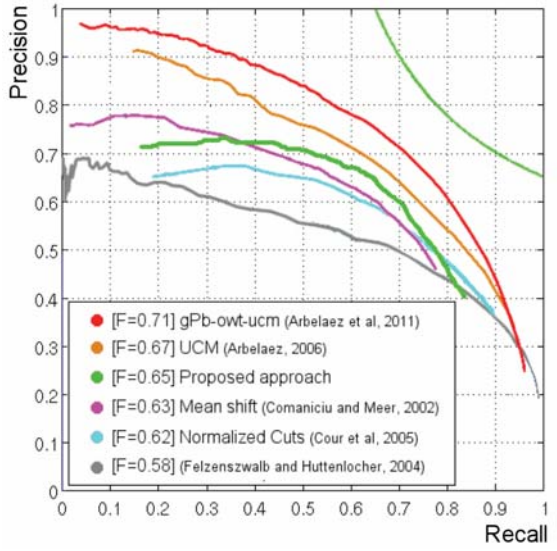


Fig. 11. Comparison of our segmentation proposal with other approaches. The precision-recall curves for these other approaches have been downloaded from http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/ [2]

in the scene, so that it is subsequently possible to direct the focus of attention on them. The aim is that, with a capacity of computing limited, will be able to process information of a scene getting, as a result, the overall perception and feeling in real-time of the entire scene as we discussed in the first paragraphs of Section I. This paper mainly focuses on two lines of work, complementary to the design of the mechanism of gaze control: the design of the whole architecture and its implementation in a dedicated device, a high-performance AP SoC; and the concrete development of a hardware capable of generating multi-resolution color images that will be the basis of the complete perceptual system. While the work also proposes a novel multiscale segmenter, it could be concluded that this module is currently the weakest one on the architecture. The obtained results are very positive, but they are only preliminary. We have to deeply work on the design of the segmenter for its future implementation in hardware. Of course, future work will also focus on the mechanism of attention, which currently does not include basic elements as an efficient inhibition of return.

ACKNOWLEDGMENTS

This paper has been partially supported by the Spanish Ministerio de Economía y Competitividad TIN2015-65686-C5 and FEDER funds.

REFERENCES

[1] P. Arbeláez, Boundary extraction in natural images using ultrametric contour maps, *Proc. 5th IEEE Workshop Perceptual Org. In Computer Vision*: 182–189, 2006

[2] P. Arbeláez, M. Maire, C.C. Fowlkes and J. Malik, Contour detection and hierarchical image segmentation, *IEEE Trans. Pattern Anal. Mach. Intell.* 33 (5): 898–916, 2011

[3] F. Arrebola, P. Camacho and F. Sandoval, Generalization of shifted fovea multiresolution geometries applied to object detection, *ICIAP* 2: 477–484, 1997

[4] D.G. Bailey and C-S. Bouganis, Implementation of a foveal vision mapping, *FTP*, 22–29, 2009

[5] L. Brun and W.G. Kropatsch, Construction of combinatorial pyramids, *Lecture Notes in Computer Science* 2726: 1-12, 2003

[6] P. Burt, T. Hong and A. Rosenfeld, Segmentation and estimation of image region properties through cooperative hierarchical computation, *IEEE Trans. Syst. Man Cybern.* 11 (12): 802-809, 1981

[7] P. Camacho, F. Arrebola and F. Sandoval, Adaptive fovea structures for space variant sensors, *Lecture Notes on Computer Science* 1311: 422–429, 1997

[8] F.J. Coslado, P. Camacho, M. González, F. Arrebola and F. Sandoval, VLSI implementation of a Foveal Polygon segmentation algorithm, *ICIAP*: 185–190, 1999

[9] D. Comaniciu and P. Meer, Mean-shift: a robust approach toward feature space analysis, *IEEE Trans. Pattern Anal. Machine Intell.* 24(5): 603–619, 2002

[10] T. Cour, F. Benezit and J. Shi, Spectral segmentation with multiscale graph decomposition, *Proc. IEEE CS Conf. Computer Vision and Pattern Recog.* 2: 1124–1131, 2005

[11] P. Felzenszwalb and D. Huttenlocher, Efficient graph-based image segmentation, *Int. Journal Computer Vision* 59(2): 167-181, 2004

[12] Y. Haxhimusa and W.G. Kropatsch, Segmentation graph hierarchies, *SSPR & SPR*, 343-351, 2004

[13] T. Hong and A. Rosenfeld, Compact region extraction using weighted pixel linking in a pyramid, *IEEE Trans. Pattern Anal. Mach. Intell.* 6(2): 222-229, 1984

[14] J.M. Jolion, Stochastic pyramid revisited, *Pattern Recognition Lett.* 24(8): 1035-1042, 2003

[15] R. Marfil, L. Molina–Tanco, A. Bandera, J.A. Rodríguez and F. Sandoval, Pyramid segmentation algorithms revisited, *Pattern Recognition* 39(8): 1430–1451, 2006

[16] R. Marfil, A.J. Palomino and A. Bandera, Combining segmentation and attention: a new foveal attention model, *Front. Comput. Neurosci.*, 2014

[17] D.R. Martin, C.C. Fowlkes, D. Tal and J. Malik, A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, *ICCV*: 416–425, 2001

[18] J.K. O Regan, Solving the real mysteries of visual perception: The world as an outside memory. *Canadian Journal of Psychology* 46: 461-488, 1992

[19] T. Sakamoto, C. Nakanishi and T. Hase, Software pixel interpolation for digital still cameras suitable for a 32-bit MCU, *IEEE Trans. Consumer Electronics* 44(4), 1998

[20] G. Sandini and G. Metta, Retina-like sensors: motivations, technology and applications, in F. Barth, J. Humphrey, T. Secomb (Eds.) *Sensors and Sensing in Biology and Engineering*, Springer, 2002

[21] C. Ware, *Visual thinking for design*, Morgan Kaufmann 2008

Determination of the most relevant images for scene recognition in mobile robotics

D. Santos-Saavedra, R. Iglesias, X.M. Pardo and C.V. Regueiro.

***Abstract*—Scene recognition is an important topic in mobile robotics; nevertheless, it is still difficult to reach robust and reliable solutions, in fact there are many studies which have clearly arrived at the conclusion that human visual scene recognition is view point dependant. This paper suggests a metric aimed to value the images according to their coverage and thus analyse whether they contain enough information to identify the scene. This metric prioritizes images which contain key points scattered all over, and which reflect the presence of close and far obstacles. The experimental results show to what extent the metric described in this paper resembles what the human would choose as relevant. In particular this metric was used to rank the images from two datasets and the results were compared with the selection made by humans who chose the most representative ones.**

***Index Terms*—Scene recognition, robotics, relevance and viewpoint.**

I. Introduction

Scene recognition is an important topic in the field of robotics. Almost any kind of robot-people interaction involves some kind of understanding of the environment where the robot is. Thus, for example, assistive robots willing to collaborate with humans must be endowed with the ability to perceive scenes or even recognize human intentions. In general, and despite of the notable progress experienced by mobile robots, they yet need a greater semantic understanding of their surroundings, so that they can perform increasingly complex tasks [1]. Identifying indoor scenes, such as an office or a Kitchen, should be something within the reach for a new-generation robots, especially if we really want them to live with other robots and humans in the same environments, and if we want them to become something valuable and useful for humans.

By *scene* we mean a place in which a human can act within or navigate. There is not such straightforward definition for *understanding a scene*, and it usually hides a lot of complexities. In this paper we will only concern about one of the basic aspects of scene understanding, and that is the ability to classify an image into a limited number of semantic categories. Nonetheless, even how these categories are built is strongly dependant on what are they going to be used for, i.e., the task to be solved. Therefore, we can finish either in a human-centric perspective or an unsupervised solution based on the robot's perspective. In the first case, we have the *supervised scene recognition* where the categories have a clear semantic meaning from the human point of view (scenes or places meant for specific actions such as eating (kitchen/restaurant), working (office), sleeping (bedroom), etc.). On the second case, unsupervised solutions, we speak about unsupervised scene recognition methods where the number of categories is not set in advance [2][3]. In this second case it is possible to skip the use of vast amounts of human-labelled training data and it is even possible to build a system aimed to categorize the environments where the robot moves in an ongoing way, instead of creating a model before the robot is deployed. In both cases, supervised or unsupervised scene recognition, it will be necessary to determine when the robot needs to identify and categorize the image acquired from its camera. An obvious solution would be to fix a reasonably sampling frequency, so that the robot identifies the scene using images regularly taken from the camera. In scene recognition this frequency affects directly to the size and quality of the captured dataset (supervised solutions) or the clusters that are being created (unsupervised recognition): low frequencies can cause the loss of relevant information and high frequencies will collect many identical or irrelevant data. Other not-so-obvious solutions could consider time, the motor commands carried out by the robot, or even its localization (SLAM) to determine when it is appropriate to identify the scene. In this case we would work with a non-regular sampling, or at least non-regular scene identification.

Finally, there is another important remark to be considered regarding scene recognition. Usually there is an important dependency amongst scene recogntion and the view used to capture the images [4][5][6][7]. Many studies of scene recognition may have unfairly biased results either towards view-dependency or an unwarranted degree of generalisation to novel views. Studies which limit viewing to just one or two viewpoints in a scene are unfairly biasing observers, in this case the robot, toward view-dependency. On the other hand, studies without some control of the observer movement would be able to conclude very little regarding generalisation to novel views. In our case, and inspired in what happens in *active vision*, we suggest the use of a two-module approach to solve the task of scene recognition. The first module will involve some kind of processing to determine whether the robot has just acquired a relevant image which contains enough information to carry out the recognition of the scene, or on the contrary the image is irrelevant. The second module is the one

D. Santos-Saavedra is in CiTIUS (Centro Singular de Investigación en Tecnoloxías da Información), Universidade de Santiago de Compostela.
E-mail: david.santos@usc.es

R. Iglesias and X.M.Pardo are in CiTIUS (Centro Singular de Investigación en Tecnoloxías da Información), Universidade de Santiago de Compostela.

C.V. Regueiro is in the Department of Electronics and Systems, Universidade da Coruña.

which actually draws the conclusion about the kind of scene, obviously this second module can use models created in an unsupervised or a supervised way.

In this paper we will describe the first module, i.e. the process that is being carried out to filter noisy images and on the contrary detect meaningful ones.

II. Detection of meaningful images for Scene Recognition

In this section, we will describe the metric we have developed to rate the images and thus analyse whether they are suitable to carry out the recognition of the scene. An important aspect to be considered is the fact that we will assume the existence of RGBD images, i.e. we get not only visual but also depth information. Nowadays this kind of sensor is widely accepted in the robotics community and therefore this assumption seems to be reasonable.

To determine whether an image contains enough information to identify the scene we have inspired in the concept of canonical views [8][9]. According to these papers, canonical views are images that gather three characteristics: coverage, orthogonality and likelihood. In the past we have already developed an algorithm [10] to get rid of redundant images focusing on properties such as orthogonality and likelihood, i.e., a cluster/category must group together a good amount of similar images (likelihood), and different clusters must be as orthogonal as possible (thus maximizing the inter clusters distance). In this paper we will focus on the coverage property, i.e., images that provide enough information and which are really representative of the scene. Both papers [8][9] assume multiple images taken at different locations of the same scene, therefore, these works assume some kind of supervised procedure, and offer a way to summarize image collections or determine the orthogonal views of the same object. In our case, we will build a metric which only will rank the images according to their content of information. To do so, we will detect the key points in the image, and we will use a metric that analyses their distribution across the image, i.e., how these key points are distributed or scattered along the whole image. Our metric will try to prioritize those images that contain relevant key points scattered all over it. The underlying motivation of this is the fact that we will recognize the scene using either global or local features, or a combination of both. Therefore, the image will be dealt with as a *whole*, a unique entity. Global (holistic) representations as Centrist, local difference binary patterns, etc, capture the general structural properties on the image. Another common representation is the gist of a scene, commonly associated with low-level global features such as colour, spatial frequencies and spatial organization. This Gist descriptor is a global representation that divides the image into a 4x4 grid, filtering each cell using a bank of Gabor filters. On the other hand, the discovery of salient points in the image (local descriptors such as SIFT or SURF [17]), together with the use of bag-of-words (which involves a clustering process of the key-points-descriptors into a limited set of categories: visual words), allows the representation of an image by the histogram of the visual words that it contains. Hence, even with this kind of local information the image is treated as an entity without further consideration about where the salient points have been detected or which object they are associated to. There are other alternatives where this does not happen, such as those based on the construction of strongly coupled models for object and scene identification [11]. These kinds of alternatives exploit the contextual relationships between scenes and objects. These coupled models usually build prior probabilities of each scene class based on the object-level likelihood estimates, i.e., joint probabilities of the presence of different object categories influence the estimate of each scene category. In our case we will not assume any kind of object detection. Therefore we are not interested in the detection of a particular object that might be the key element to guess the scene. This is the reason why we will prioritize images with salient points scattered all over it, instead of images where there might be few but representative objects which concentrate most of the saliency in the image (Fig. 1).

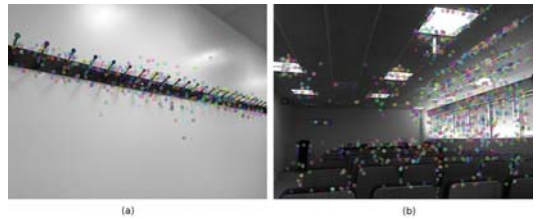


Fig. 1. Saliency points will be scattered depending of the information in the image. If there are few objects **(a)** the saliency points will be concentrated in a small region. However, extensive views **(b)** will get a high scattering of points.

On the other hand, we will also prefer those images where the distance to the objects in the scene is as far as possible. There are many studies which have arrived at the conclusion of the fact that human visual scene recognition is viewpoint dependent, such that recognition performances is better for experienced views than novel views [5][6][7]. In particular, Mou and McNamara [12][13] proposed that people use intrinsic frames of reference to specify locations of objects in memory. More specifically, the spatial reference directions which are established to represent locations of objects are not egocentric. Instead the special reference directions are intrinsic to the layout of the objects. In [4] several experiments were carried out to determine whether a generalisation to novel views is possible in the recognition of large scale spatial environments after restricted simulated movements through a scene during learning. In particular the Christou et al. examined whether observers who learn to recognize a room from one specific set of directions can recognize the same room from different, unfamiliar, directions. To see this they analysed the impact on the recognition of the scene when there are significant changes in the angular distance between the test view and the study view (i.e., when the same scene is presented to the user but when there has been a significant change of the angle of the viewpoint with respect to the scene), therefore these views contained new features not visible from the familiar direction and lacked some of the features detected in the

familiar direction. Finally, they also analysed the recognition of the same scene under specular reflections (mirror images). In general they found that familiar and novel views were recognized, although familiar direction views were always recognized faster and more often. This is clearly an indication of view-dependency in the internal representation immediately after training. Nevertheless in what is called *passive learning* this generalization to novel views become more difficult to achieve. By *passive learning* Christou et al. mean the procedure where the observer sat in front a computer monitor while series of images were repeatedly presented to them (there is not active exploration). This kind of learning is similar to what we carry out for the robot. Therefore, fixing a metric which determines when the images are relevant enough is somehow equivalent to reducing the number of possible viewpoint for the same scene. Only those viewpoints which maximize the metric will be elegible. From our point of view, images that capture both close and far objects in the scene will provide more robustness since the global properties will tend to be more similar. On the other hand somehow is equivalent to increase the likelihood that the intersection of train and test images taken under different angles will have more keypoints in common. Finally, there are several works that clearly state that both near and far scene points provide information for the recognition process [14]. Figure. 1 represents situations where we can get keypoints (detected using SURF) scattered through a significant region of the image, nevertheless, these images contain few and near objects, that making more difficult the identification of the scene.

Fig. 2. High scattering of saliency points is not just related to far or extensive views. Images showing many details **(a)** or many close objects **(b)** can produce high scattering of interest points.

Considering all this, we have designed an heuristic metric to determine the relevance, coverage or *representativeness* of an image:

$$\sum_{i=1}^{M}(H(NK(Q_i)-T))+\frac{1}{M}\sum_{i=1}^{M}(\bar{\mathrm{d}}(Q_i)*\frac{NK(Q_i)}{n\_pixels}) \quad (1)$$

This metric works on the basis that the original image is divided into a set of M regions of the same size (quadrants). $NK(Qi)$ represents the number of keypoints detected in the i^{th} quadrant of the image, n_pixels represents the number of pixels in every quadrant (which is the same for all of them), and $H(Nk(Q_i)-T)$ is the Heavised step function:

$$H(x)=\begin{cases}0 & if x<0\\ 1 & if x\geq 0\end{cases}$$

Finally, the value $\bar{\mathrm{d}}(Q_i)$, in the second term, represents the normalized average distance of the pixels in the i^{th} quadrant:

$$\bar{\mathrm{d}}(Q_i)=\frac{1}{n\_pixels}*\sum_{j=1}^{n\_pixels}(\frac{d_{pixel_j}}{max\_range})$$

As we can see in Eq. 1 , this first metric is made up of two terms: The first one reflects up to what extent the key points are scattered all over the image, i.e., the value of this first term, with the interval $[0, M]$ represents the number of regions in the image that contain more than T key points. The second term tries to prioritize those images that contain far objects in the scene. This is the reason why this second term will issue a number in the interval $[0, 1]$, where values close to 1 correspond to images where most of the quadrants contain many key points, and where, in average, the objects in every quadrant appear to be almost as far as the maximum range of distances that can be detected with the RGBD sensor.

We have carried out two sets of experiments to analysed our metric, in the first one the key points have been detected using SURF analysis [17]. In the second experiment, we have used new key point detectors specifically designed for 3D meshes, and which can be used directly in the metric described before: NARF [16], and Harris 3D [15].

In the case of NARF (*normal aligned radial feature*) the points selected as *key points* are located on object borders in positions where the surface is assumed to be stable (i.e. positions that can be reliably detected even if the object is observed from another perspective, and which also ensure robust estimation of the normal and where there are sufficient changes in the immediate vicinity). Therefore this method searches for object borders in the image and scores the points trying to reflect how much the surface changes at these positions. Finally it performs a smoothing and non-maximum suppression to find the final interest points.

Regarding the Harris3D key-point detector, this algorithm determines a local neighbourhood around a vertex. Then it tries to fit a quadratic surface to this set of points by applying principal component analysis. Next it computes the derivatives using a smoothing over the surface. These derivatives are used to compute the Harris response for each vertex and finally, the final set of interest points is selected.

To apply these new key point detectors and our previous metric, we transform our 2D image into a cube (Figure 3). In the new representation, the new Z-axis represents the distances at which the different points can be detected. In our case the values in this new axis range from 0.8m to 3.5m. Due to this new representation, what before were quadrants are now small 3D-volumes. Therefore, our metric will determine how many of these small 3D-volumes contain more than T key points. Finally, once again our metric will prioritize those cubes which contain most of the key-points in the 3D volumes furthest away (in the Z-axis). Nevertheless, in this case, the second term in eq. 1 has been slightly modified:

$$\sum_{i=1}^{M}(H(NK(Q_i)-T))+\ldots \\ +\frac{1}{M}\sum_{i=1}^{M}\frac{1}{n\_pixels}\sum_{j=1}^{NK(Q_i)}(\frac{d_j(Q_i)}{max\_range}), \quad (2)$$

where $d_j(Q_i)$ represents the distance to the interest point j in quadrant i.

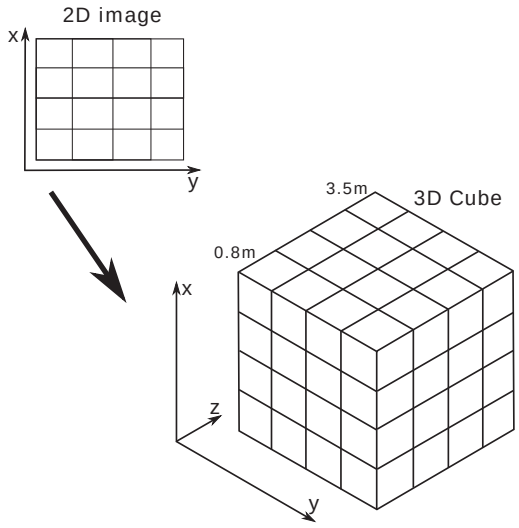


Fig. 3. When Harris 3D or Narf 3D are applied to detect the interest points in the image, the use of our metric involves dealing wiht the image as a cube instead of a 2D surface

Since the metric described in the previous sections is mostly heuristic, we must analyse to what extent it resembles what the humans would consider as relevant, or (as it was described in the introduction), it would help to improve scene recognition by the suppression of noisy images. This is what we describe in the next section.

III. Experimental results

Next we describe the experiments carried to analyse to what extent the metrics described in the previous section represent what the human would consider as relevant. We have built two datasets, one of them was used to analyse the metric described in Eq. 1 using SURF to detect the keypoints. The second data set was used to validate Eq. 2, in this case Harris3D was used to detect the interest points. The reason why we had to work with the two data sets is due to the fact that when we logged the first one, we didn't keep the cloudpoints provided by the RGBD sensor, and which are required to apply Harris3D.

In the first experiment, we worked with a collection of 3078 images in gray scale obtained at the Centro Singular de Investigacin en Tecnoloxas da Informacin (CiTIUS), University of Santiago de Compostela. These images correspond to 11 different classes (table III). For each one of these classes we asked different people to select the most *representative* images. Each voter could select the 10 most representative images of the class, and rank them according to its degree of *representativeness* (being 10 the most representative and 1 the least one). Table III shows the number of voters for each class. This table also shows the number of different images that have been selected at least once (i.e. the number of different images that have been selected by at least one of the voters). In general we have observed an important variability in the poll, i.e., there is little agreement amongst different voters. Figure 4 shows this clearly for one of the classes. In this case, we can see that 48 images were selected as relevant out of 417. Nonetheless, only 11 of the selected images have accumulated more than 10 points out of 70. Therefore less than 23% of the total number of images selected hardly manage to reach an accumulated score slightly meaningful accumulated. On the other hand, we also noticed that in general there isn't any subset of images that can be clearly labelled as relevant according to the human. This is why we conclude that in general there is a high level of *noise* in the experiment.

Figure 5 represents the percentage of images that have been selected by at least once of the human voters and which are also amongst the best ones according to our metric. This means that if we apply Eq. 1 to rank our images, and we take the first quartile of the images for each class, we end up with a set which contains more than 35% of the images which have been also selected as relevant by the humans. If we consider the first half of the most representative images according to our ranking, we end up with a new set which contains near to 70% of the images also selected as relevant by the human. Finally, although these numbers might seem a bit too low, we must remind the high level of noise or disagreement detected in the poll. In fact, if we try to get rid of this noise considering only the images that have been selected as relevant for at least three of the voters (hence looking for the images in which there is a higher level of coincidence), we end up with sets of images that are also highly ranked according to our metric (Figure 6).

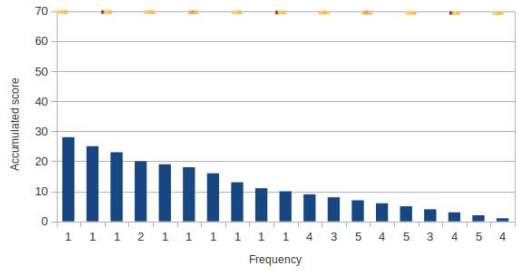


Fig. 4. score that has been accumulated by the images selected as relevant for class 1. The maximum score that an image could have reached is 70 (in case all voters had agreed in scoring it with 10 points, *most representative*. Nevertheless, there are only 11 images that reach an accumulated score above 10 points, thus showing a high disagreement amongs the voters.

We collected a second set of RGBD images to test the metric described in Eq 2. As we pointed out before, this new set was collected due to the necessity to log the point clouds got by the RGBD sensor in every acquisition (and not only the image together with the distances). Once again Table III shows the number of classes and voters for this new set.

Class	Number of Images	Number of voters	Number of images selected at least once
0 (Entrepreneurship Lab.)	417	7	48
1 (Kitchen)	172	7	49
2 (Staircase)	167	6	44
3 (Instrumentation Lab.)	129	8	50
4 (Laboratories)	296	7	51
5 (Common Staff areas (Ground floor)	465	8	64
6 (Common Staff areas (floor 1 and 2)	496	7	46
7 (Common Staff areas (S1 floor)	395	8	63
8 (Robotics Lab)	154	7	43
9 (Assembly Hall)	171	6	39
10 (Office)	216	6	46

TABLE I
SUMMARY OF THE FIRST DATA SET, PARTICIPANTS AND RESULTS OF THE POLL

Class	Number of Images	Number of voters
0 (Instrumentation Lab.)	61	30
1 (Laboratories)	92	30
2 Common Staff areas (floor 1 and 2)	144	30
3 Common Staff areas (Ground floor)	122	29
4 (Kitchen)	96	30

TABLE II
SUMMARY OF THE SECOND DATA SET AND PARTICIPANTS

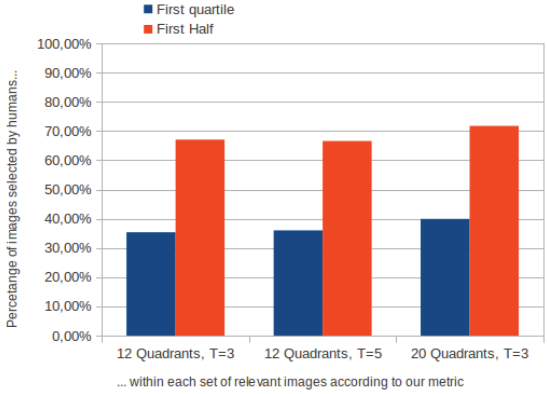


Fig. 5. This image shows the concordance amongst the set of relevant images selected by humand and our metric. The bars show the percentage of images which have been selected as relevant by at least one of the voters, and which are also in the first quartile or the first half of the most relevant images according to our metric.

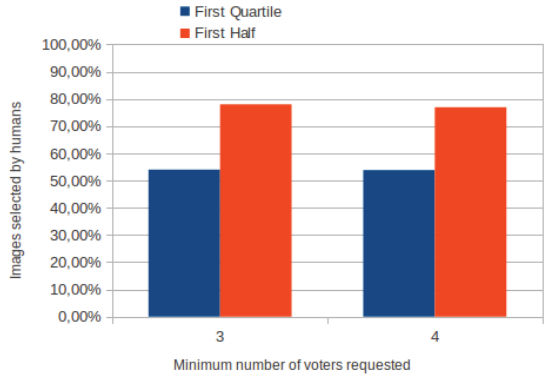


Fig. 6. Concordance amongst human selection and our ranking when the images considered as relevant are those which were selected by at least three or four voters. The increase in the requirements regarding human relevance, responds to the need of getting rid of noisy selections

We also have modified the design of experiment in order to reduce the amount of time required to do it. In other to reduce the level of noise of the poll we asked more people to take part in this new experiment, on the other hand, due to the subjective character of *representativeness* we decided to ask for the collaboration of people who were not familiarized with the environment were the data set was collected. In this case we ranked the images for every class, using the metric shown in Eq. 1. Therefore, we applied the metric which uses Harris 3D to detect the interest points in the 3D mesh. Once all the images have been ranked, we built three sets for every class: the set with the images in the first tercile of the ranked images, the set with the images in the second tercile, and the set with the images in the third tercil. Thus, each voter was presented

with an imaged randomly picked from each one of these three sets (no information was given to the voter about this), and the voter had to give a score to each one of these three images. This score could be 3 points (very representative), 2 points (medium degree of representativeness), 1 point (low representativeness), or 0 points (meaningless, the image does not provide information at all). Figure 7.

Fig. 7. Snaptshot of the screen taken during the performance of the second experiment. Each user was shown three pictures of the same scene. The user had to give a score to every picture, thus selecting the most representative, the second most representative and so on. The user had also the chance to determine that an image was completely meaningless.

Figure 8 summarizes the results of the experiment. As we can see in this figure there is a clear correlation amongst human-relevance and our ranking. The set formed by the images best ranked by our metric collected the highest sum of votes (according to the human). The set with second tercil of images ranked using our metric collected the second highest number of votes, and finally, the set with the images in the tail according to our metric was the one which also collected the lowest number of human votes. Table III shows the detailed information of the results.

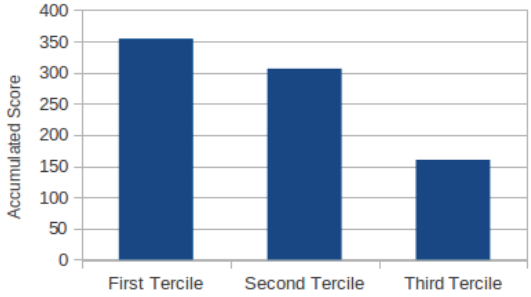


Fig. 8. Score accumulated by the images in the first tercile, second tercile and third tercile according to the raking obtained using our metric

.

As we can see in Table III. Most of images in the first third (according to our ranking) got the 3 or 2 points, showing a high relevance.

IV. Conclusions

Scene recognition is an important topic in mobile robotics; nevertheless, it is still difficult to reach robust and reliable solutions, in fact there are many studies which have clearly arrived at the conclusion that human visual scene recognition is view point dependant, and therefore there is an unwarranted degree of generalization to novel views. This paper suggests a metric aimed to rank the images according to their coverage and thus analyse whether they contain enough information to identify the scene. Somehow this is equivalent to fix valid viewpoints of a scene and discard noisy images that may be irrelevant. Our metric prioritizes images which contain key points scattered all over, and which reflect the presence of close and far obstacles. The detection of the key points has been done using SURF, but also new operators specifically developed for interest point detection on 3D meshes (Harris 3D and NARF 3D). The experimental results show to what extent the metric described in this paper resembles what the human would choose as relevant. In particular this metric was used to rank the images from two datasets collected at the CiTIUS research centre (University of Santiago de Compostela, Spain). The results were compared with the selection made by humans who chose the most representative images. From the experimental results it is clear that there is clear correlation amongst our metric and the selection made by humans. Nevertheless, there is still a high level of noise in the experiments, due subjective character of what people understand by *representativeness*. This is reflected by a low degree of agreement amongst different people during the poll. Besides, some people rely on the specificity of certain objects in the image to identify the scene (such as a fire extinguisher, etc). Therefore, in this case the knowledge about the environment might alter the results of the experiment. This is the reason why, on a second test, we only allow the participation of people who were no familiar with the environment were the pictures were taken. On the other hand we also reduced the number of images to be analysed by each person in this second experiment, to reduce the burden and thus lower the impact of tiredness and lack of attention.

Acknowledgments

This work was supported by the research grant TIN2012-32262 (FEDER) and by the Galician Government (Xunta de Galicia), Consolidation Program of Competitive Reference Groups: GRC2014/030 (FEDER cofunded).

References

[1] P. Espinace, T. Kollar, N. Roy, and A. Soto. Indoor scene recognition by a mobile robot through adaptive object detection. Robotics and Autonomous Systems, 61(9), 932-947. 2013.

[2] H. Madokoro, A. Yamanashi, and K. Sato. Unsupervised semantic indoor scene classification for robot vision based on context of features using Gist and HSV-SIFT. Pattern Recognition in Physics, 1(1), pp.93-103, 2013.

[3] Rudolph Triebel, Rohan Paul, Daniela Rus, and Paul Newman. Parsing outdoor scenes from streamed 3d laser data using online clustering and incremental belief updates. Twenty-Sixth AAAI Conference on Artificial Intelligence, Toronto, Canada, July 26-22, 2012.

[4] Chris Christou, and Heinrich H. Bülthoff. View-direction specificity in scene recognition after active and passive learning. 1997.

[5] C.G. Christou, and H.H. Bülthoff. View dependence in scene recognition after active learning. Memory & Cognition, 27(6), pp.996-1007, 1999.

[6] W. Mou, Y. Fan, T.P. McNamara, and C.B. Owen. Intrinsic frames of reference and egocentric viewpoints in scene recognition. Cognition, 106(2), pp.750-769, 2008.

First Tercile	3 points	2 points	1 point	0 points
Class 0	23	4	3	
Class 1	13	15	2	
Class 2	13	10	5	2
Class 3	11	14	4	
Class 4	18	9	2	1
Second Tercile	3 points	2 points	1 point	0 points
Class 0	4	13	10	3
Class 1	13	13	3	1
Class 2	14	11	3	2
Class 3	11	13	4	1
Class 4	10	15		5
Third Tercile	3 points	2 points	1 point	0 points
Class 0	2	13	14	1
Class 1	4	2	9	15
Class 2	3	9	10	8
Class 3	6	2	9	12
Class 4	2	5	5	18

TABLE III
DETAILLED INFORMATION OF THE RESULTS OF THE SECOND EXPERIMENTS WITH HUMANS. THESE TABLES SHOW THE SCORE RECEIVED BY THE IMAGES IN THE FIRST, SECOND AND THIR TERCILE ACCORDING TO OUR RANKING

[7] N. Burgess, H.J. Spiers, and E. Paleologou. Orientational manoeuvres in the dark: Dissociating allocentric and egocentric influences on spatial memory. Cognition, 94, 149-166, 2004.

[8] P.M. Hall, and M. Owen. Simple Canonical Views. In BMVC, 2005.

[9] I. Simon, N. Snavely, and S.M. Seitz. Scene summarization for online image collections. In Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on (pp. 1-8). IEEE, October, 2007.

[10] D. Santos-Saavedra, R. Iglesias, X. M. Pardo. "Unsupervised Method to Remove Noisy and Redundant Images in Scene Recognition". Robot 2015: Second Iberian Robotics Conference, Advances in Intelligent Systems and Computing 418, vol.2, 695-704, ISBN: 978-3-319-27148-4, 2015.

[11] T. Ehtiati, and J.J. Clark. A strongly coupled architecture for contextual object and scene identification. In Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on (Vol. 3, pp. 69-72). IEEE, August, 2004.

[12] W. Mou, and T.P. McNamara. Intrinsic frames of reference in spatial memory. Journal of Experimental Psychology: Learning, Memory, and Cognition, 28, 162-170, 2002.

[13] W. Mou, and T.P. McNamara, B. Rump, and C. Xiao. Roles of egocentric and allocentric spatial representations in locomotion and reorientation. Journal of Experimental Psychology: Learning, Memory, and Cognition, 32, 1274-1290, 2006.

[14] C. Cadena, D. Glvez-Lpez, J.D. Tards, and J. Neira. Robust place recognition with stereo sequences. Robotics, IEEE Transactions on, 28(4), pp.871-885, 2012.

[15] I. Sipiran, B. Bustos. Harris 3d: a robust extension of the harris operator for interest point detection on 3d meshes. The Visual Computer 27(11), 963-976, 2011.

[16] B. Steder, R.B. Rusu, K. Konolige, W. Burgard. Narf: 3d range image features for object recognition. In: Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), vol. 44, 2010.

[17] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool. Speeded-up robust features (SURF). Computer vision and image understanding, 110(3), 346-359, 2008.

Orientation Estimation by Means of Extended Kalman Filter, Quaternions, and Charts

P. Bernal-Polo and H. Martínez-Barberá

***Abstract*—In this paper, an orientation estimation algorithm is presented. This algorithm is based on the Extended Kalman Filter, and uses quaternions as the orientation descriptor. For the filter update, measures from an inertial measurement unit (IMU) are used. The IMU consists in a triaxial angular rate sensor, and an also triaxial accelerometer.**

Quaternions describing orientations live in the unit sphere of $\mathbb{R}^4$. Knowing that this space is a manifold, and applying some basic concepts regarding these mathematical objects, an algorithm that reminds the also called "Multiplicative Extended Kalman Filter" arises in a natural way.

The algorithm is tested in a simulated experiment, and in a real one.

***Index Terms*—Extended Kalman filter, quaternions, orientation estimation, IMU, manifold, charts.**

I. Introduction

THE estimation of the orientation of a system is a field of interest for many applications, among which are robotics, virtual reality, and vehicle navigation.
In such fields, knowledge of the orientation may be required to:

- Control an *Unmanned Vehicle*.
- Knowing the orientation of a camera in a scenario, and using it for virtual reality.
- Knowing the heading of a vehicle in a navegation system.
- Transform the acceleration measures for positioning in indoor navigation.

The problem has been addressed using different sensor configurations, but it is usual to include an Inertial Motion Unit (IMU, composed of a triaxial angular rate sensor, and an also triaxial accelerometer). Other sensors can be used for aiding the IMU, such as a magnetometer ([1]), a camera ([2],[3],[4]), or a GPS ([5]). In our case we will only use the IMU measures so that we can focus our attention on the estimation algorithm.

Pablo Bernal is with University of Murcia.
E-mail: pablo.bernal.polo@gmail.com

Humberto Martnez is with University of Murcia.
E-mail: humberto@um.es

Faculty of computer science. Applied engineering research group.

A. Quaternions

The problem is usually attacked using the extended Kalman filter, and quaternions as orientation descriptors ([6],[7]).
Quaternions are prefered over other orientation descriptors for several practical reasons:

1) Motion equations are treated linearly with quaternions.
2) There are no singularities (and then the "gimbal lock", present in Euler angles, is avoided).
3) They describe the orientation in a continous way (unlike axis-angle representation).
4) They are determined by 4 parameters (in contrast with a rotation matrix, that needs 9 of them).

Quaternions are hypercomplex numbers with three different imaginary units, and can be expressed as

$$\boldsymbol{q} = q_0 + q_1\,\boldsymbol{i} + q_2\,\boldsymbol{j} + q_3\,\boldsymbol{k} \quad .$$

They can also be expressed in a vectorial form as

$$\boldsymbol{q} = \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix} = \begin{pmatrix} q_0 \\ \mathbf{q} \end{pmatrix} \quad .$$

Quaternion product is determined by the Hamilton axiom

$$\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{i} * \boldsymbol{j} * \boldsymbol{k} = -1 \quad ,$$

wich produces the multiplication rule

$$\boldsymbol{p} * \boldsymbol{q} = \begin{pmatrix} p_0\,q_0 - \mathbf{p} \cdot \mathbf{q} \\ p_0\,\mathbf{q} + q_0\,\mathbf{p} + \mathbf{p} \times \mathbf{q} \end{pmatrix} \quad .$$

Quaternions describing rotations have the form

$$\boldsymbol{q} = \begin{pmatrix} \cos(\theta/2) \\ \widehat{\mathbf{q}}\,\sin(\theta/2) \end{pmatrix} \quad , \tag{1}$$

with $\widehat{\mathbf{q}}$ the unit vector that defines the rotation axis, and θ the angle of rotation.
Having this form, they satisfy the restriction

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1 \quad .$$

This means that quaternions describing orientations live in the unit sphere of $\mathbb{R}^4$. This space has dimension 3, although its elements are determined using 4 parameters.
We will have to use basic concepts of manifold theory in order to handle this kind of space.

II. Orientation distributions

When dealing with the Kalman filter, the distribution of a random variable, $\mathbf{x}$, is encoded by its mean, $\overline{\mathbf{x}}$, and its covariance matrix, $\mathbf{P}$, defined as

$$\mathbf{P} = E\left[(\mathbf{x}-\overline{\mathbf{x}})(\mathbf{x}-\overline{\mathbf{x}})^T\right] \quad . \tag{2}$$

This can be done when our random variables are elements of an Euclidean space. But when a random variable is an element of a manifold, the covariance matrix could be bad defined. This is our case, where the random variable $\boldsymbol{q}-\overline{\boldsymbol{q}}$, in general, does not describe an orientation. Then we need to redefine our covariance matrix in a different way, but we can not change the form of the definition of the covariance matrix if we want to use the Kalman filter, because this precise form is used in its derivation.

We will solve this problem by defining our covariance matrix in a different space.

a) Definition: Manifold:

An n-manifold, M^n, is a topological space in wich each point is localy homeomorphic to the euclidean space, $\mathbb{R}^n$. This is, each point $x \in M^n$ has a neighborhood $N \subset M^n$ for wich we can define a homeomorphism $f : N \to B_n$, with B_n the unit ball of $\mathbb{R}^n$.

b) Definition: Chart:

A chart for a topological space, M, is a homeomorphism, φ, from an open subset, $U \subset M$, to an open subset of the Euclidean space, $V \subset \mathbb{R}^n$. This is, a chart is a function

$$\varphi : U \subset M \to V \subset \mathbb{R}^n \quad ,$$

with φ a homeomorphism.

Traditionally a chart is expresed as the pair (U, φ).

A. The set of charts

Suppose we know the expected value of our distribution of quaternions, $\overline{\boldsymbol{q}}$. In such case, we can express any unit quaternion as

$$\boldsymbol{q} = \overline{\boldsymbol{q}} * \boldsymbol{\delta} \quad , \tag{3}$$

with $\boldsymbol{\delta}$ another unit quaternion.

And then, we can define the set of charts

$$\varphi_{\overline{\boldsymbol{q}}}(\boldsymbol{q}) = 2\left(\frac{\delta_1}{\delta_0}, \frac{\delta_2}{\delta_0}, \frac{\delta_3}{\delta_0}\right) \quad . \tag{4}$$

The set of charts (4) is used in ([7]), but this work does not talk about charts, and what we call "chart update" is not applied.

In each chart, the quaternion $\overline{\boldsymbol{q}}$ is mapped to the origin. As the space deformation produced in the neighborhood of the origin is small, being the variance small, the distribution in each chart will be similar to the distribution in the manifold.

The inverse transformations for these charts are given by

$$\varphi_{\overline{\boldsymbol{q}}}^{-1}(\mathbf{e}_q) = \overline{\boldsymbol{q}} * \frac{1}{\sqrt{4+|\mathbf{e}_q|^2}} \begin{pmatrix} 2 \\ \mathbf{e}_q \end{pmatrix} \quad . \tag{5}$$

B. Transition maps

Given two charts $(U_\alpha, \varphi_\alpha)$ and (U_β, φ_β) describing a manifold, with $U_{\alpha\beta} = U_\alpha \cap U_\beta \neq \emptyset$, a function $\varphi_{\alpha\beta} : \varphi_\alpha(U_{\alpha\beta}) \to \varphi_\beta(U_{\alpha\beta})$ can be defined as

$$\varphi_{\alpha\beta}(x) = \varphi_\beta\left(\varphi_\alpha^{-1}(x)\right) \quad ,$$

with $x \in \varphi_\alpha(U_{\alpha\beta})$.

Having the set of charts defined in (4), and having two charts centered in quaternions $\overline{\boldsymbol{p}}$ and $\overline{\boldsymbol{q}}$ related by $\overline{\boldsymbol{p}} = \overline{\boldsymbol{q}} * \overline{\boldsymbol{\delta}}$, then our transition map takes the form

$$\begin{aligned} \mathbf{e}_p &= \varphi_{\overline{\boldsymbol{q}},\overline{\boldsymbol{p}}}(\mathbf{e}_q) = \\ &= 2\,\frac{\overline{\delta}_0\,\mathbf{e}_q - 2\,\overline{\boldsymbol{\delta}} - \overline{\boldsymbol{\delta}} \times \mathbf{e}_q}{2\,\overline{\delta}_0 + \overline{\boldsymbol{\delta}} \cdot \mathbf{e}_q} \quad . \end{aligned} \tag{6}$$

III. Motion equations and measurement equations

The state of the system is defined by an orientation, encoded by a unit quaternion $\boldsymbol{q}$, and by an angular velocity measured in the reference frame attached to our system, given by a vector $\boldsymbol{\omega}'$.

The unit quaternion defines a rotation transformation that determines the orientation of the system. This transformation relates vectors expressed in a reference frame attached to the solid whose state we want to describe (denoted as $\mathbf{v}'$), with the same vectors expressed in an inertial reference frame (denoted as $\mathbf{v}$), in which the gravity vector has the expression $\mathbf{g} = (0, 0, -1)$.

Thus, our rotation transformation will yield

$$\mathbf{v} = T_{\mathbf{R}}(\mathbf{v}') \quad .$$

Using a rotation matrix,

$$\mathbf{v} = \mathbf{R}\,\mathbf{v}' \quad .$$

And using our unit quaternion,

$$\boldsymbol{v} = \boldsymbol{q} * \boldsymbol{v}' * \boldsymbol{q}^* \quad .$$

where this time $\boldsymbol{v} = \left(\begin{smallmatrix} 0 \\ \mathbf{v} \end{smallmatrix}\right)$, and $\boldsymbol{q}^*$ is the complex conjugate quaternion, that being $\boldsymbol{q}$ a unit quaternion, it is also its inverse.

A. Motion equations

Knowing what our quaternion means, we can write the motion equations for the random variables that we use to describe the state of our system:

$$\dot{\boldsymbol{q}}(t) = \boldsymbol{q}(t) * \begin{pmatrix} 0 \\ \boldsymbol{\omega}'(t) \end{pmatrix} \quad , \tag{7}$$

$$\boldsymbol{\omega}'_t = \boldsymbol{\omega}'_{t\text{-}\Delta t} + \mathbf{q}^{\omega}_t \quad , \tag{8}$$

where $\mathbf{q}^{\omega}_t$ is the process noise.

B. Measurement equations

Similarly, we can write the measurement equations that connect the random variables describing the state of our system, with the random variables describing the measures of our sensors:

$$\mathbf{a}_t^m = \mathbf{R}_t^T\ (\mathbf{a}_t - \mathbf{g}_t) + \mathbf{r}_t^a \quad , \tag{9}$$

$$\boldsymbol{\omega}_t^m = \boldsymbol{\omega}'_t + \mathbf{r}_t^\omega \quad , \tag{10}$$

where $\mathbf{r}_t^a$ is the noise in the accelerometer measure, $\mathbf{r}_t^\omega$ is the noise in the gyroscope measure, and $\mathbf{a}_t$ are not gravitational accelerations.

IV. State prediction

A. Evolution of the expected value of the state

Taking the expected value in equations (7) and (8), assuming the random variables $\boldsymbol{q}(t)$ and $\boldsymbol{\omega}'(t)$ to be independent, and the expected value of the process noise, $\overline{\mathbf{q}}^\omega(\tau)$, to be constant when $\tau \in (t-\Delta t, t]$, our differential equations are transformed in other ones whose solutions are

$$\overline{\boldsymbol{\omega}}'_t = \overline{\boldsymbol{\omega}}'_{t\text{-}\Delta t} + \overline{\mathbf{q}}_t^\omega \quad , \tag{11}$$

$$\begin{aligned}\overline{\boldsymbol{q}}_t &= \overline{\boldsymbol{q}}_{t\text{-}\Delta t} * \boldsymbol{q}_\omega = \\ &= \overline{\boldsymbol{q}}_{t\text{-}\Delta t} * \begin{pmatrix} \cos\left(\frac{\overline{\omega}'_t\,\Delta t}{2}\right) \\ \frac{\overline{\boldsymbol{\omega}}'_t}{\overline{\omega}'_t}\ \sin\left(\frac{\overline{\omega}'_t\,\Delta t}{2}\right)\end{pmatrix} \quad .\end{aligned} \tag{12}$$

B. Evolution of the state covariance matrix

As we need a covariance matrix of the form (2), we will define the covariance matrix of the state in the set of charts defined by (4).

1) Diferential equations for our charts: Having the definition for our charts in (3) and (4), we can find the differential equations for the quaternion $\boldsymbol{\delta}$, using the differential equations for $\boldsymbol{q}$ (7), and for $\overline{\boldsymbol{q}}$. And having the differential equations for the $\boldsymbol{\delta}$ quaternion, we can find the differential equations for a point $\mathbf{e} = 2\,\frac{\boldsymbol{\delta}}{\delta_0}$ on the charts:

$$\dot{\mathbf{e}} = \Delta\boldsymbol{\omega}' - \left[\,2\,\overline{\boldsymbol{\omega}}' + \Delta\boldsymbol{\omega}'\,\right]_\times \frac{\mathbf{e}}{2} + \frac{\mathbf{e}}{2}\,\frac{\mathbf{e}^T}{2}\,\Delta\boldsymbol{\omega}' \quad . \tag{13}$$

Note that, by the definition of the charts, the vector of random variables $\mathbf{e}_t$ is expressed in the chart centered in the quaternion $\overline{\boldsymbol{q}}_t$. For each instant, t, we have a quaternion $\overline{\boldsymbol{q}}_t$ defining the chart for that time. Then the differential equation (13) defines the evolution of the vector $\mathbf{e}$ that "travels" between charts.

2) Differential ecuations for the covariance matrix: We define the covariance matrix for the state of our system by

$$\begin{aligned}\mathbf{P}_t &= \begin{pmatrix} E\left[\,\mathbf{e}_t\ \mathbf{e}_t^T\,\right] & E\left[\,\mathbf{e}_t\ \Delta\boldsymbol{\omega}_t'^T\,\right] \\ E\left[\,\Delta\boldsymbol{\omega}'_t\ \mathbf{e}_t^T\,\right] & E\left[\,\Delta\boldsymbol{\omega}'_t\ \Delta\boldsymbol{\omega}_t'^T\,\right]\end{pmatrix} = \\ &= \begin{pmatrix} \mathbf{P}_t^{ee} & \mathbf{P}_t^{e\omega} \\ \mathbf{P}_t^{\omega e} & \mathbf{P}_t^{\omega\omega}\end{pmatrix} \quad .\end{aligned} \tag{14}$$

Notice that we do not write "$\Delta\mathbf{e}_t$". By the definition of the charts, the term $\mathbf{e}_t$ can be interpreted as a difference from quaternion $\overline{\boldsymbol{q}}_t$, which is mapped to the origin of the chart. Note also that being the covariance matrix symmetric, we do not need to find the evolution of all its terms. We just need to find the evolution of the terms $\mathbf{P}^{ee}$, $\mathbf{P}^{e\omega}$, and $\mathbf{P}^{\omega\omega}$.

We are looking for an estimation of the covariance matrix in t, using the information in $t - \Delta t$. This is, we want $\mathbf{P}_{t|t\text{-}\Delta t}$ using $\mathbf{P}_{t\text{-}\Delta t|t\text{-}\Delta t}$.
For $\mathbf{P}^{\omega\omega}$ it is easy to find this relation. Assuming the random variables $\boldsymbol{\omega}'_{t\text{-}\Delta t}$ and $\mathbf{q}_t^\omega$ to be independents,

$$\mathbf{P}_t^{\omega\omega} = \mathbf{P}_{t\text{-}\Delta t}^{\omega\omega} + \mathbf{Q}_t^\omega \quad . \tag{15}$$

If we had a function $\mathbf{e}_t = f(\mathbf{e}_{t-\Delta t})$, we could replace in (14), and perhaps obtain a relation similar to (15). However, we can not find a closed solution for (13).

In some works, an approximate solution for the $\overline{\mathbf{e}}_t$ differential equation is used. We will see that this is not a good practice, because in that way we would despise the influence of $\mathbf{q}_t^\omega$ in $\mathbf{P}^{e\omega}$ and $\mathbf{P}^{ee}$.

Not being able to solve the $\mathbf{e}_t$ differential equation (13), we can find a differential equation for $\mathbf{P}_t$ using this differential equation for $\mathbf{e}_t$.

Starting from (14),

$$\frac{d\,\mathbf{P}}{dt} = \begin{pmatrix} \dot{\mathbf{P}}^{ee} & \dot{\mathbf{P}}^{e\omega} \\ \dot{\mathbf{P}}^{\omega e} & \dot{\mathbf{P}}^{\omega\omega}\end{pmatrix} \quad ,$$

with

$$\dot{\mathbf{P}}^{ee} = E\left[\,\dot{\mathbf{e}}\ \mathbf{e}^T\,\right] + E\left[\,\mathbf{e}\ \dot{\mathbf{e}}^T\,\right] \quad , \tag{16}$$

$$\dot{\mathbf{P}}^{e\omega} = E\left[\,\dot{\mathbf{e}}\ \Delta\boldsymbol{\omega}'^T\,\right] + E\left[\,\mathbf{e}\ \Delta\dot{\boldsymbol{\omega}}'^T\,\right] \quad . \tag{17}$$

After replacing (13), assuming that higher moments are negligible compared to second-order moments, and assuming the independence of the random variables $\mathbf{e}$ and $\Delta\dot{\boldsymbol{\omega}}'$, (16) and (17) can be approximated by

$$\begin{aligned}\dot{\mathbf{P}}^{ee} \approx &\left(\mathbf{P}^{e\omega}\right)^T - \left[\,\overline{\boldsymbol{\omega}}'\,\right]_\times \mathbf{P}^{ee} + \\ &+ \mathbf{P}^{e\omega} - \mathbf{P}^{ee}\,\left[\,\overline{\boldsymbol{\omega}}'\,\right]_\times^T \quad .\end{aligned} \tag{18}$$

$$\dot{\mathbf{P}}^{e\omega} \approx \mathbf{P}^{\omega\omega} - \left[\,\overline{\boldsymbol{\omega}}'\,\right]_\times \mathbf{P}^{e\omega} \quad . \tag{19}$$

3) Evolution equations for the covariance matrix: We are dealing with a system of inhomogeneous linear matrix equations. Generally, a system of this type is untreatable. However, in our case the equations are sufficiently decoupled. Given a solution for $\mathbf{P}^{\omega\omega}$, we can find an approximate solution for $\mathbf{P}^{e\omega}$. And with this solution, we can find an approximate solution for $\mathbf{P}^{ee}$.

Denoting $\boldsymbol{\Omega} = [\,\overline{\boldsymbol{\omega}}'_t\,]_\times$ we can write

$$\mathbf{P}^{e\omega}_{t|t\text{-}\Delta t} \approx e^{-\boldsymbol{\Omega}\,\Delta t}\left[\,\mathbf{P}^{e\omega}_{t\text{-}\Delta t|t\text{-}\Delta t} + \right. \\ \left. + \Delta t\;\mathbf{P}^{\omega\omega}_{t\text{-}\Delta t|t\text{-}\Delta t} + \frac{\Delta t}{2}\,\mathbf{Q}^{\omega}_{t}\,\right] , \quad (20)$$

$$\mathbf{P}^{ee}_{t|t\text{-}\Delta t} \approx e^{-\boldsymbol{\Omega}\,\Delta t}\left[\,\mathbf{P}^{ee}_{t\text{-}\Delta t|t\text{-}\Delta t} + \right. \\ + (\Delta t)^2\,\mathbf{P}^{\omega\omega}_{t\text{-}\Delta t|t\text{-}\Delta t} + \frac{(\Delta t)^2}{3}\,\mathbf{Q}^{\omega}_{t} + \\ + \mathbf{P}^{e\omega}_{t\text{-}\Delta t|t\text{-}\Delta t}\,\Delta t + \\ \left. + \left(\mathbf{P}^{e\omega}_{t\text{-}\Delta t|t\text{-}\Delta t}\right)^T\Delta t\right] e^{-\boldsymbol{\Omega}^T\,\Delta t} . \quad (21)$$

V. Prediction of the measure

A. Expected value of the measurement

1) Expected value of the gyroscope measurement: Taking the expected value on (10),

$$\overline{\boldsymbol{\omega}}^m_t = \overline{\boldsymbol{\omega}}'_t + \overline{\mathbf{r}}^{\omega}_t .$$

2) Expected value of the accelerometer measurement: Taking the expected value on (9), knowing that the $\mathbf{g}_t$ vector does not change, and assuming that the accelerations affecting our system, $\mathbf{a}_t$, does not depend on its orientation,

$$\overline{\mathbf{a}}^m_t = E\left[\,\mathbf{R}^T_t\,\right]\left(\,\overline{\mathbf{a}}_t - \overline{\mathbf{g}}_t\,\right) + \overline{\mathbf{r}}^{a}_t .$$

Using the unit quaternion describing the orientation of our system, this relation takes the form

$$\overline{\boldsymbol{a}}^m_t = \overline{\boldsymbol{q}}^*_t * \left(\,\overline{\boldsymbol{a}}_t - \overline{\boldsymbol{g}}_t\,\right) * \overline{\boldsymbol{q}}_t + \overline{\boldsymbol{r}}^{a}_t .$$

And if we use the rotation matrix constructed from this unit quaternion,

$$\overline{\mathbf{a}}^m_t = \overline{\mathbf{R}}^T_t\left(\,\overline{\boldsymbol{a}}_t - \overline{\boldsymbol{g}}_t\,\right) + \overline{\mathbf{r}}^{a}_t .$$

B. Covariance matrix of the measurement

The measurement is related with the state through the measurement equations:

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t, \mathbf{r}_t) .$$

We can approximate linearly the relationship around the expected values, $\overline{\mathbf{x}}_t$, and $\overline{\mathbf{r}}_t$, using a Taylor series

$$\mathbf{z}_t \approx \mathbf{h}(\overline{\mathbf{x}}_t, \overline{\mathbf{r}}_t) + \mathbf{H}_t\,(\mathbf{x}_t - \overline{\mathbf{x}}_t) + \mathbf{M}_t\,(\mathbf{r}_t - \overline{\mathbf{r}}_t) ,$$

being

$$\mathbf{H}_t = \left.\frac{\partial\,\mathbf{h}(\mathbf{x}_t, \mathbf{r}_t)}{\partial \mathbf{x}_t}\right|_{\substack{\overline{\mathbf{x}}_t \\ \overline{\mathbf{r}}_t}} , \quad \mathbf{M}_t = \left.\frac{\partial\,\mathbf{h}(\mathbf{x}_t, \mathbf{r}_t)}{\partial \mathbf{r}_t}\right|_{\substack{\overline{\mathbf{x}}_t \\ \overline{\mathbf{r}}_t}} , \quad (23)$$

the jacobian matrices evaluated on the expected value of the random variables.

Then, our prediction equation of the measurement covariance matrix is given by

$$\mathbf{S}_t = \mathbf{H}_t\,\mathbf{P}_t\,\mathbf{H}^T_t + \mathbf{M}_t\mathbf{R}_t\mathbf{M}^T_t .$$

1) Gyroscope block: The measurement equation for the gyroscope is linear. Using (10) and (23) we obtain

$$\mathbf{H}^{\omega}_t = \begin{pmatrix}\mathbf{0} & \mathbf{1}\end{pmatrix} .$$

2) Accelerometer block: For being consistent the Kalman filter formulation, the acceleration term, $\mathbf{a}_t$, should be part of the noise in the measurement. Then, measurement noise in our Kalman filter has two components:

- $\mathbf{r}_t$: the main measurement noise. This noise comes from the sensor.
- $\mathbf{a}_t$: non-gravitational accelerations acting on the system. These accelerations obstruct the measurement of the $\overline{\mathbf{g}}$ vector.

Recalling that we express the covariance matrix of the state in a chart, and knowing that doing $\boldsymbol{q}_t = \overline{\boldsymbol{q}}_t$ in the manifold, is equivalent to $\mathbf{e}_t = \mathbf{0}$ in the chart, we will have for the accelerometer measurement equation:

$$\mathbf{a}^m_t \approx \overline{\mathbf{a}}^m_t + \left.\frac{\partial\,\mathbf{h}_a(\mathbf{e}_t, \mathbf{a}_t)}{\partial \mathbf{e}_t}\right|_{\substack{\mathbf{e}_t=\mathbf{0} \\ \mathbf{a}_t=\overline{\mathbf{a}}_t}} \mathbf{e}_t + \\ + \left.\frac{\partial\,\mathbf{h}_a(\mathbf{e}_t, \mathbf{a}_t)}{\partial \mathbf{a}_t}\right|_{\substack{\mathbf{e}_t=\mathbf{0} \\ \mathbf{a}_t=\overline{\mathbf{a}}_t}} (\mathbf{a}_t - \overline{\mathbf{a}}_t) + \mathbf{r}^a_t - \overline{\mathbf{r}}^a_t .$$

Denoting $\overline{\mathbf{g}}^{\mathbf{R}}_t = \overline{\mathbf{R}}^T_t\,(\overline{\mathbf{a}} - \overline{\mathbf{g}})$, and after doing some calculus,

$$\left.\frac{\partial\,\mathbf{h}_a}{\partial \mathbf{a}_t}\right|_{\substack{\mathbf{e}_t=\mathbf{0} \\ \mathbf{a}_t=\overline{\mathbf{a}}_t}} = \overline{\mathbf{R}}^T_t ,$$

$$\left.\frac{\partial\,\mathbf{h}_a(\mathbf{e}_t)}{\partial \mathbf{e}_t}\right|_{\substack{\mathbf{e}_t=\mathbf{0} \\ \mathbf{a}_t=\overline{\mathbf{a}}_t}} = \begin{pmatrix} 0 & -\left(\overline{\mathbf{g}}^{\mathbf{R}}_t\right)_3 & \left(\overline{\mathbf{g}}^{\mathbf{R}}_t\right)_2 \\ \left(\overline{\mathbf{g}}^{\mathbf{R}}_t\right)_3 & 0 & -\left(\overline{\mathbf{g}}^{\mathbf{R}}_t\right)_1 \\ -\left(\overline{\mathbf{g}}^{\mathbf{R}}_t\right)_2 & \left(\overline{\mathbf{g}}^{\mathbf{R}}_t\right)_1 & 0 \end{pmatrix} .$$

3) Measurement covariance matrix: Assuming independence of all random variables involved in the measure, our prediction equation for the measure is

$$\mathbf{S}_t = \begin{pmatrix} \left[\overline{\mathbf{g}}^{\mathbf{R}}_t\right]_\times & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{pmatrix} \mathbf{P}_t \begin{pmatrix} \left[\overline{\mathbf{g}}^{\mathbf{R}}_t\right]_\times & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{pmatrix}^T + \\ + \begin{pmatrix} \overline{\mathbf{R}}^T_t\,\mathbf{Q}^a_t\,\overline{\mathbf{R}}_t + \mathbf{R}^a_t & \mathbf{0} \\ \mathbf{0} & \mathbf{R}^{\omega}_t \end{pmatrix} ,$$

where

- $\mathbf{Q}^a_t$ is the covariance matrix of the random variable $\mathbf{a}_t$.
- $\mathbf{R}^a_t$ is the covariance matrix that describes the noise in the accelerometer measurement, which is modeled by the random variable $\mathbf{r}^a_t$.
- $\mathbf{R}^{\omega}_t$ is the covariance matrix describing the noise in the gyroscope measurement, which is modeled by the random variable $\mathbf{r}^{\omega}_t$.

VI. Update

Although in the original Kalman filter algorithm just an update is necessary, the fact that our covariance matrix is expressed in a chart requires the computation of a second update.

A. Kalman update

Given the estimate of the covariance matrix of the state, $\mathbf{P}_{t|t\text{-}\Delta t}$, and the estimate of the measurement covariance matrix, $\mathbf{S}_{t|t\text{-}\Delta t}$, we are able to calculate the optimal Kalman gain,

$$\mathbf{K}_t \;=\; \mathbf{P}_{t|t\text{-}\Delta t}\,\mathbf{H}_t^T\,\mathbf{S}_{t|t\text{-}\Delta t}^{-1} \quad .$$

Having the gain, we can update the state distribution:

$$\mathbf{x}_{t|t} \;=\; \mathbf{x}_{t|t\text{-}\Delta t} \;+\; \mathbf{K}_t\,\left(\mathbf{z}_t \;-\; \mathbf{z}_{t|t\text{-}\Delta t}\right) \quad , \tag{24}$$

$$\mathbf{P}_{t|t} \;=\; (\mathbf{1}-\mathbf{K}_t\,\mathbf{H}_t)\;\mathbf{P}_{t|t\text{-}\Delta t}\,(\mathbf{1}-\mathbf{K}_t\,\mathbf{H}_t)^T \;+\; \mathbf{K}_t\,\mathbf{R}_t\,\mathbf{K}_t^T \quad . \tag{25}$$

Note that we do not use the update equation "$\mathbf{P}_{t|t} = (\mathbf{1}-\mathbf{K}_t\mathbf{H}_t)\,\mathbf{P}_{t|t\text{-}\Delta t}$" as it is only valid in the case of using the optimal Kalman gain. As in the EKF it does not generally occur, we can only ensure that our $\mathbf{P}_{t|t}$ matrix remains positive-definite, and therefore is a covariance matrix, using (25).

The new state distribution will be expressed in the chart centered on $\overline{\mathbf{x}}_{t|t-\Delta t}$. In this chart, $\overline{\mathbf{e}}_{t|t\text{-}\Delta t} = \mathbf{0}$, but $\overline{\mathbf{e}}_{t|t} \neq \mathbf{0}$.

B. Manifold update

In order to find the quaternion corresponding to the updated vector $\overline{\mathbf{e}}_{t|t}$ expressed in the chart, we must reverse the function $\varphi_{\overline{\mathbf{q}}_{t|t\text{-}\Delta t}}(\overline{\mathbf{e}}_{t|t})$ making use of the equation (5):

$$\begin{aligned}\overline{\mathbf{q}}_{t|t} \;&=\; \overline{\mathbf{q}}_{t|t\text{-}\Delta t} * \boldsymbol{\delta}(\overline{\mathbf{e}}_{t|t}) \;=\\ &=\; \overline{\mathbf{q}}_{t|t\text{-}\Delta t} * \frac{1}{\sqrt{4+|\overline{\mathbf{e}}_{t|t}|^2}}\begin{pmatrix}2\\ \overline{\mathbf{e}}_{t|t}\end{pmatrix} \quad .\end{aligned} \tag{26}$$

C. Chart update

After the Kalman update, the new state distribution is expressed in the chart centered on $\overline{\mathbf{x}}_{t|t\text{-}\Delta t}$. We must update the covariance matrix expressing it in the chart centered on the updated state, $\overline{\mathbf{x}}_{t|t}$. To do this we must use the concept of transition maps, that for our charts take the form of (6). Being nonlinear this relation, we need to find a linear approximation to be able to transform the covariance matrix from a chart to another:

$$\mathbf{e}_p \;\approx\; \mathbf{e}_p(\overline{\mathbf{e}}_q) \;+\; \left.\frac{\partial\,(\mathbf{e}_p)_i}{\partial\,(\mathbf{e}_q)_\ell}\right|_{\mathbf{e}_q=\overline{\mathbf{e}}_q}\,(\,\mathbf{e}_q \;-\; \overline{\mathbf{e}}_q\,) \quad . \tag{27}$$

After deriving our transition map and evaluating in $\overline{\mathbf{e}}_{t|t}$, having identified the charts $\varphi_{\overline{\mathbf{p}}} = \varphi_{\overline{\mathbf{q}}_{t|t}}$ and $\varphi_{\overline{\mathbf{q}}} = \varphi_{\overline{\mathbf{q}}_{t|t\text{-}\Delta t}}$, we find out

$$\begin{aligned}\mathbf{e}_p(\mathbf{e}_q) \;&\approx\; \bar{\delta}_0\left(\bar{\delta}_0\,\mathbf{1} \;-\; \left[\boldsymbol{\delta}\right]_\times\right)\,\left(\mathbf{e}_q \;-\; \overline{\mathbf{e}}_{t|t}\right) \;=\\ &=\; \mathbf{G}_t\;\left(\mathbf{e}_q \;-\; \overline{\mathbf{e}}_{t|t}\right) \quad .\end{aligned}$$

Our update equations for the charts will be

$$\begin{aligned}\mathbf{P}^{ee}_{\overline{\mathbf{q}}_{t|t}} \;&=\; \mathbf{G}_t\;\mathbf{P}^{ee}_{\overline{\mathbf{q}}_{t|t\text{-}\Delta t}}\;\mathbf{G}_t^T \quad ,\\ \mathbf{P}^{e\omega}_{\overline{\mathbf{q}}_{t|t}} \;&=\; \mathbf{G}_t\;\mathbf{P}^{e\omega}_{\overline{\mathbf{q}}_{t|t\text{-}\Delta t}} \quad ,\\ \mathbf{P}^{\omega\omega}_{\overline{\mathbf{q}}_{t|t}} \;&=\; \mathbf{P}^{\omega\omega}_{\overline{\mathbf{q}}_{t|t\text{-}\Delta t}} \quad .\end{aligned}$$

VII. Experimental validation

A. Simulated experiment

1) Experiment setup: For testing our algorithm, we can think in a simple and intuitive simulation. Let us imagine that we can freely roam the surface of a torus, which is a manifold whose space can be described in $\mathbb{R}^3$ by

$$\begin{aligned}x(\theta,\phi) \;&=\; (\,R \;+\; r\,\cos\theta\,)\,\cos\phi \quad , &(28)\\ y(\theta,\phi) \;&=\; (\,R \;+\; r\,\cos\theta\,)\,\sin\phi \quad , &(29)\\ z(\theta,\phi) \;&=\; r\,\sin\theta \quad . &(30)\end{aligned}$$

We can define a path in the torus using a third parameter to set the other two:

$$\begin{aligned}\theta \;&=\; v_\theta\;t \quad , &(31)\\ \phi \;&=\; v_\phi\;t \quad . &(32)\end{aligned}$$

We will use the path defined by $v_\phi = 1\,\mathrm{rad/s}$ and $v_\theta = 3\,\mathrm{rad/s}$, and we will travel the path around the torus 3 times. This path can be seen in Figure 1.

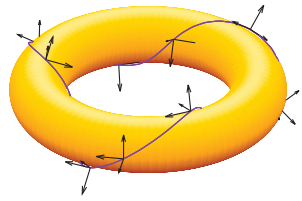


Fig. 1. Path followed on the torus in our simulation. Reference frames that define the orientation of the IMU can be observed.

Accelerations occurring in the IMU can be calculated by differentiating twice in (28)-(30) with respect to the t parameter, resulting

$$\begin{aligned}\ddot{x}(\theta,\phi) \;&=\; -(\,R \;+\; r\,\cos\theta\,)\,\cos\phi\,v_\phi^2 \;+\\ &\quad +\; 2\,r\,\sin\theta\,v_\theta\,\sin\phi\,v_\phi \;+\\ &\quad -\; r\,\cos\theta\,v_\theta^2\,\cos\phi \quad , &(33)\\ \ddot{y}(\theta,\phi) \;&=\; -(\,R \;+\; r\,\cos\theta\,)\,\sin\phi\,v_\phi^2 \;+\\ &\quad -\; 2\,r\,\sin\theta\,v_\theta\,\cos\phi\,v_\phi \;+\\ &\quad -\; r\,\cos\theta\,v_\theta^2\,\sin\phi \quad , &(34)\\ \ddot{z}(\theta,\phi) \;&=\; -\,r\,\sin\theta\,v_\theta^2 \quad . &(35)\end{aligned}$$

Now, we can define a reference frame for each point of the path. The axis of the reference frame will have the directions of the vectors

$$\left\{\;\frac{\partial\,\mathbf{x}}{\partial\theta}\;,\;\frac{\partial\,\mathbf{x}}{\partial\phi}\;,\;\frac{\partial\,\mathbf{x}}{\partial\theta}\times\frac{\partial\,\mathbf{x}}{\partial\phi}\;\right\} \quad . \tag{36}$$

After making the derivatives in (36), and choosing the direction of the vectors so that $\hat{\boldsymbol{z}}$ points outward the torus surface, our rotation matrix relating a vector measured in the IMU reference frame, with the same vector measured in the external reference frame will be

$$\mathbf{R} = \begin{pmatrix} | & | & | \\ \hat{\mathbf{x}} & \hat{\mathbf{y}} & \hat{\mathbf{z}} \\ | & | & | \end{pmatrix} = \begin{pmatrix} -\sin\theta\cos\phi & \sin\phi & \cos\theta\cos\phi \\ -\sin\theta\sin\phi & -\cos\phi & \cos\theta\sin\phi \\ \cos\theta & 0 & \sin\theta \end{pmatrix} . \quad (37)$$

Matrix (37) can be expressed as the product of 3 known rotation matrices:

$$\mathbf{R} = \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{pmatrix} . \quad (38)$$

Recognizing these matrices in (38), and using (1), we can find out the quaternion describing the orientation of our reference frame:

$$\boldsymbol{q} = \begin{pmatrix} \cos(\phi/2) \\ 0 \\ 0 \\ \sin(\phi/2) \end{pmatrix} * \begin{pmatrix} \cos(\theta/2) \\ 0 \\ -\sin(\theta/2) \\ 0 \end{pmatrix} * \begin{pmatrix} 0 \\ 1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{pmatrix} . \quad (39)$$

Having (39) we can obtain the $\dot{\boldsymbol{q}}$ quaternion:

$$\dot{\boldsymbol{q}} = \left(\dot{\boldsymbol{q}}_\phi * \boldsymbol{q}_\theta + \boldsymbol{q}_\phi * \dot{\boldsymbol{q}}_\theta\right) * \boldsymbol{q}_0 \quad , \quad (40)$$

where

$$\dot{\boldsymbol{q}}_\phi = \begin{pmatrix} -\sin(\phi/2) \\ 0 \\ 0 \\ \cos(\phi/2) \end{pmatrix} \frac{v_\phi}{2} \quad , \quad (41)$$

$$\dot{\boldsymbol{q}}_\theta = \begin{pmatrix} -\sin(\theta/2) \\ 0 \\ -\cos(\theta/2) \\ 0 \end{pmatrix} \frac{v_\theta}{2} \quad . \quad (42)$$

And with the $\dot{\boldsymbol{q}}$ quaternion we can use (7) to get the angular velocity:

$$\boldsymbol{\omega} = 2\, \boldsymbol{q}^* * \dot{\boldsymbol{q}} \quad . \quad (43)$$

With all, we can generate a succession of states $\begin{pmatrix} \boldsymbol{q}_r \\ \boldsymbol{\omega}_r \end{pmatrix}$, and measures $\begin{pmatrix} \mathbf{a}_t^m \\ \boldsymbol{\omega}_t^m \end{pmatrix}$.
Now, taking only the succession of measures, we can try to reconstruct the state of our system using the algorithm that we have reviewed, and then compare it with the real state $\begin{pmatrix} \boldsymbol{q}_r \\ \boldsymbol{\omega}_r \end{pmatrix}$. Our way to compare states will be through the definition of two errors:

$$e_\theta = 2\arccos\left((\boldsymbol{q}^* * \boldsymbol{q}_r)_0\right) \quad , \quad (44)$$

$$e_g = \frac{|\mathbf{g}'_r - \mathbf{g}'_e|}{|\mathbf{g}|} \quad . \quad (45)$$

The error e_θ in (44) is defined as the angle we have to rotate the estimated reference frame to transform it into the real reference frame. This gives us a measure of how well the algorithm estimates the whole orientation, including the heading.

The error e_g in (45) is defined as the relative error between the real $\mathbf{g}'_r$ vector (the gravity vector measured in the internal reference frame), and the estimated one, $\mathbf{g}'_e$. This gives us a measure of how well the algorithm estimates the direction of the ground.

2) Setting the algorithm values:

a) Characterization of process noise: The following relationships have been established without much rigor. Just trying to be reasonable. Still, after some testing, we found that the algorithm behaves similarly with other configurations, provided that they are not disproportionate.

$$\overline{\mathbf{q}}_t^\omega = \frac{1}{2}\left(\boldsymbol{\omega}_t^m - \boldsymbol{\omega}'_t\right) \quad , \quad (46)$$

$$\mathbf{Q}_t^\omega = \overline{\mathbf{q}}_t^\omega \left(\overline{\mathbf{q}}_t^\omega\right)^T + \mathbf{R}_t^\omega \quad , \quad (47)$$

$$\overline{\mathbf{a}}_t = \mathbf{0} \quad , \quad (48)$$

$$\mathbf{Q}_t^a = \frac{1}{2}\,\mathbf{1} \quad . \quad (49)$$

b) Characterization of measurement noise: The Kalman filter requires

$$\overline{\mathbf{r}}_t^a = \mathbf{0} \quad , \quad (50)$$

$$\overline{\mathbf{r}}_t^\omega = \mathbf{0} \quad , \quad (51)$$

in order to produce an unbiased estimation.

For the covariance matrices we will set

$$\mathbf{R}_t^a = \sigma^2\,\mathbf{1} \quad , \quad (52)$$

$$\mathbf{R}_t^\omega = \sigma^2\,\mathbf{1} \quad , \quad (53)$$

and we will compare how the error behaves as a function of the magnitude of the noise in the measurement.

3) Simulation: We will place our simulation in four different scenarios, whose details have been displayed in the table I. In this way we will test the performance of the algorithm in different situations.

Test	Sampling time (s)	σ^2
1	Ranging in $(10^{-3}, 10^{-1})$	Fixed to 10^{-1}
2	Fixed to 10^{-1}	Ranging in $(10^{-5}, 10^{-1})$
3	Ranging in $(10^{-3}, 10^{-1})$	Fixed to 10^{-5}
4	Fixed to 10^{-3}	Ranging in $(10^{-5}, 10^{-1})$

TABLE I
TESTS CONDUCTED IN OUR SIMULATION.

The results of Test 1 and 2 are not shown since the errors produced are too large. This suggests that both a good processor (small sampling time) as a good sensor (small variance) are required.

For Test 3 and 4 the time evolution of the error measures are plotted in Figures 2 - 5. In Figures 3 and 5 we observe how the error becomes smaller as we improve our IMU, having a good processor. In Figures 2 and 4 we observe how the error becomes smaller as we improve our processor, having a good IMU.

In Figures 2 and 3 we note that the error tends to increase over time when the variance or the sampling time are too big. This is because we have no reference for orientation in the plane perpendicular to the gravity vector. Adding measures from a magnetometer, or a camera could provide a more accurate estimate. Although the error (44) tends to grow over

time, in Figures 4 and 5 we note that the error (45) is very small. This means that although we do not get a good estimate of the overall orientation, we will have a good reference of the direction in which is the ground.

In case we want to use only the IMU, then we will have to use both a good processor and a good sensor to minimize the error.

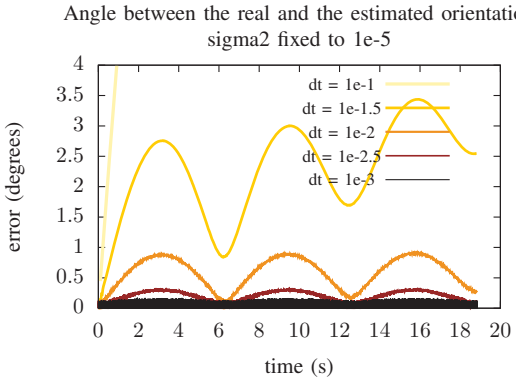


Fig. 2. Evolution of the e_θ error. Test 3.

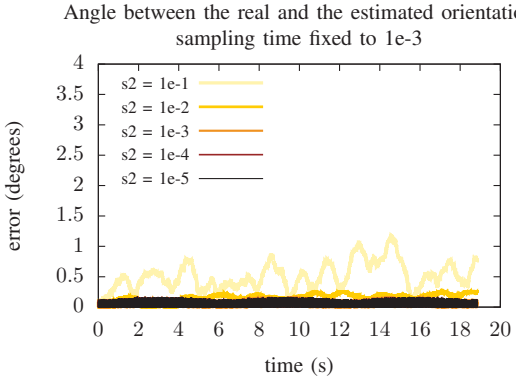


Fig. 3. Evolution of the e_θ error. Test 3.

B. Real experiment

1) Test bed: The algorithm has been implemented in a real system. It has been used a board containing a MPU6050 sensor. The processing is performed in the ATmega328P chip contained in an Arduino board. In this system the sampling time turns out to be about

$$\Delta t \quad \approx \quad 0.04\,\mathrm{s} \quad ,$$

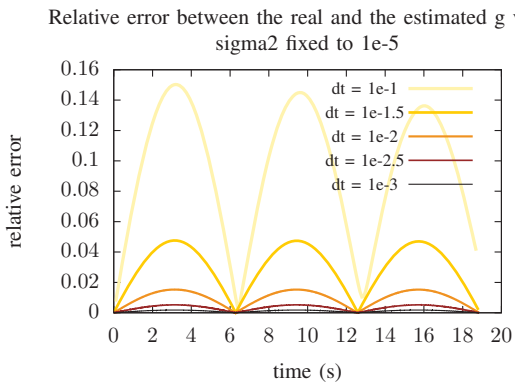


Fig. 4. Evolution of the e_g error. Test 4.

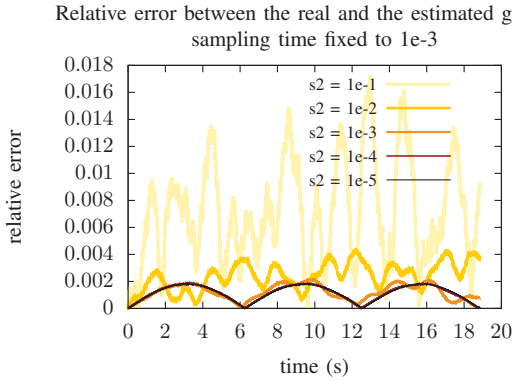


Fig. 5. Evolution of the e_g error. Test 4.

what means that the algorithm runs about 25 times per second. The sensor variance is approximately

$$\sigma_a^2 \quad \approx \quad 10^{-4}\,(\mathrm{g})^2 \quad , \tag{54}$$

$$\sigma_\omega^2 \quad \approx \quad 10^{-5}\,(\mathrm{rad/s})^2 \quad . \tag{55}$$

Figure 6 shows the assembled system, consisting in the MPU6050 sensor, the Arduino UNO, and a MTi sensor of Xsens.

2) Experiment results: We have described a series of movements with the system. The movements have been carried out in four phases. The dynamics of each phase has been more aggressive than that of the previous phase. We have tried to finish with the same orientation with which the system began. Both have been saved the sensors measurements and the estimated states which are returned by the algorithms. In Figures 7-8 these data are shown.

In Figure 7 we can see that both sensor acceleration measurements are very similar. This makes us think that the misalignment between the two sensors is small.

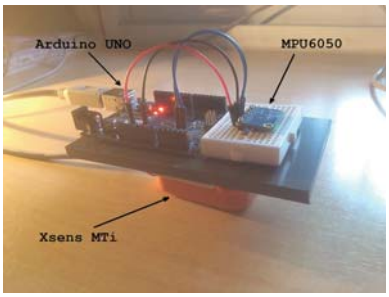


Fig. 6. Real system composed of an Arduino Uno, a MPU6050 chip, and a Mti sensor of Xsens.

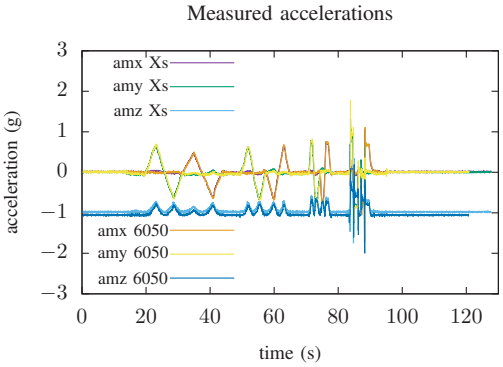


Fig. 7. Acceleration measurement during the real experiment.

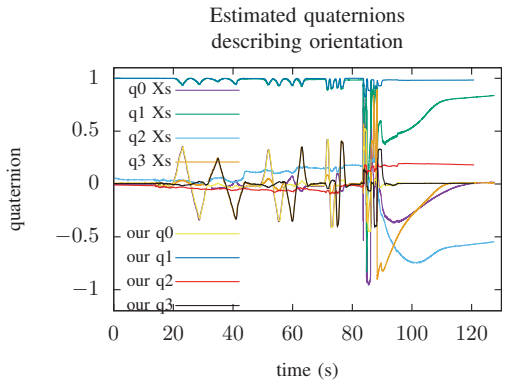


Fig. 8. Estimated quaternion describing orientation during the real experiment. The initial and final orientations are tried to match.

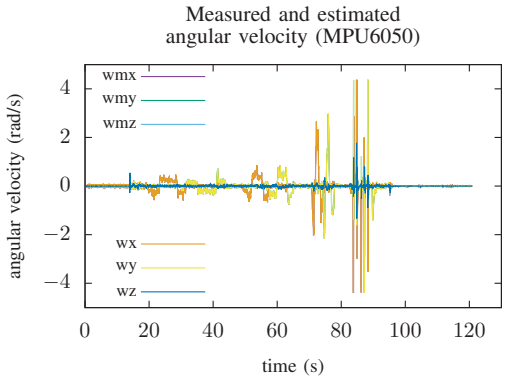


Fig. 9. Estimated and measured angular velocity during the real experiment.

In Figure 8 we note that both systems return a similar estimation of the orientation when the dynamics is not too aggressive. However, after some aggressive moves, the algorithm presented in this paper has a fast convergence. We also note the bias in the heading estimation of both algorithms when we look at "q2" quaternion component. The initial and final orientation should be the same, but we have no reference for orientation in the plane perpendicular to the gravity vector.

In Figure 9 we note that the measured angular velocity is very similar to the estimated angular velocity. Perhaps we could accept the gyroscope measure as the real angular velocity of our system. Maybe then we could get some advantage in processing speed, and therefore greater accuracy of our algorithm. But this is left for future research.

VIII. Conclusion

We have successfully used basic concepts of manifold theory for estimating orientations using quaternions as descriptors. A similar algorithm to the known as "Multiplicative Extended Kalman Filter" naturally arises in applying these concepts without having to redefine any aspect of the Extended Kalman Filter.

The orientation has been estimated using measurements from an IMU, but the basic idea introduced in this work is applicable to any other type of sensor intended to estimate the orientation.

We also have tested the algorithm in a simulated experiment and in a real one, and we have found that both a precise sensor and a fast application of the algorithm are necessary.

References

[1] A. M. Sabatini, "Quaternion-based extended kalman filter for determining orientation by inertial and magnetic sensing," *Biomedical Engineering, IEEE Transactions on*, vol. 53, no. 7, pp. 1346–1356, 2006.

[2] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 3565–3572.

[3] F. M. Mirzaei and S. I. Roumeliotis, "A kalman filter-based algorithm for imu-camera calibration: Observability analysis and performance evaluation," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 1143–1156, 2008.

[4] F. Shkurti, I. Rekleitis, M. Scaccia, and G. Dudek, "State estimation of an underwater robot using visual and inertial information," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 5054–5060.

[5] J. K. Hall, N. B. Knoebel, and T. W. McLain, "Quaternion attitude estimation for miniature air vehicles using a multiplicative extended kalman filter," in *Position, Location and Navigation Symposium, 2008 IEEE/ION*. IEEE, 2008, pp. 1230–1237.
[6] E. J. Lefferts, F. L. Markley, and M. D. Shuster, "Kalman filtering for spacecraft attitude estimation," *Journal of Guidance, Control, and Dynamics*, vol. 5, no. 5, pp. 417–429, 1982.
[7] F. L. Markley, "Attitude error representations for kalman filtering," *Journal of guidance, control, and dynamics*, vol. 26, no. 2, pp. 311–317, 2003.

APPENDIX A
STATE PREDICTION

A. Evolution of the expected value of the state

1) Evolution in the expected value of the angular velocity: Taking the expected value in the equation (8),

$$\overline{\boldsymbol{\omega}}'_t \;=\; \overline{\boldsymbol{\omega}}'_{t\text{-}\Delta t} \;+\; \overline{\mathbf{q}}^{\omega}_t \quad . \tag{56}$$

2) Evolution in the expected value of the orientation: Taking the expected value in the equation (7),

$$E\left[\frac{d\,\boldsymbol{q}(t)}{dt}\right] \;=\; E\left[\boldsymbol{q}(t) * \boldsymbol{\omega}'(t)\right] \quad .$$

Assuming the random variables $\boldsymbol{q}(t)$ and $\boldsymbol{\omega}'(t)$ to be independent,[1]

$$\frac{d\,\overline{\boldsymbol{q}}(t)}{dt} \;=\; \overline{\boldsymbol{q}}(t) * \overline{\boldsymbol{\omega}}'(t) \quad . \tag{57}$$

If we assume that the expected value of the process noise, $\overline{\mathbf{q}}^{\omega}(\tau)$, is constant when $\tau \in (t-\Delta t, t]$, then it also will $\overline{\boldsymbol{\omega}}'(\tau)$. In such case, we will have the matrix differential equation

$$\dot{\overline{\boldsymbol{q}}}(\tau) \;=\; \check{\boldsymbol{\Omega}}\, \overline{\boldsymbol{q}}(\tau) \quad ,$$

with

$$\check{\boldsymbol{\Omega}} \;=\; \begin{pmatrix} 0 & -\overline{\omega}'_1 & -\overline{\omega}'_2 & -\overline{\omega}'_3 \\ \overline{\omega}'_1 & 0 & \overline{\omega}'_3 & -\overline{\omega}'_2 \\ \overline{\omega}'_2 & -\overline{\omega}'_3 & 0 & \overline{\omega}'_1 \\ \overline{\omega}'_3 & \overline{\omega}'_2 & -\overline{\omega}'_1 & 0 \end{pmatrix}(t) \quad .$$

This differential equation has the solution

$$\overline{\boldsymbol{q}}_t \;=\; e^{\check{\boldsymbol{\Omega}}\,\Delta t}\;\; \overline{\boldsymbol{q}}_{t\text{-}\Delta t} \quad .$$

We can also express this solution using the quaternion product, as

$$\begin{aligned} \overline{\boldsymbol{q}}_t \;&=\; \overline{\boldsymbol{q}}_{t\text{-}\Delta t} * \boldsymbol{q}_{\omega} \;=\; \\ &=\; \overline{\boldsymbol{q}}_{t\text{-}\Delta t} * \begin{pmatrix} \cos\left(\frac{\overline{\omega}'_t\,\Delta t}{2}\right) \\ \frac{\overline{\boldsymbol{\omega}}'_t}{\overline{\omega}'_t}\,\sin\left(\frac{\overline{\omega}'_t\,\Delta t}{2}\right) \end{pmatrix} \quad . \end{aligned} \tag{58}$$

B. Evolution of the state covariance matrix

1) Diferential equations for our chart:

a) Differential equation for $\boldsymbol{\delta}$: Using (3), (7), and (57),

$$\begin{aligned} \boldsymbol{q} \;&=\; \overline{\boldsymbol{q}} * \boldsymbol{\delta} \qquad \Longrightarrow \\ \Longrightarrow \qquad \dot{\boldsymbol{q}} \;&=\; \dot{\overline{\boldsymbol{q}}} * \boldsymbol{\delta} \;+\; \overline{\boldsymbol{q}} * \dot{\boldsymbol{\delta}} \qquad \equiv \\ \equiv \qquad \frac{1}{2}\,\boldsymbol{q} * \boldsymbol{\omega}' \;&=\; \frac{1}{2}\,\overline{\boldsymbol{q}} * \overline{\boldsymbol{\omega}}' * \boldsymbol{\delta} \;+\; \overline{\boldsymbol{q}} * \dot{\boldsymbol{\delta}} \qquad \Longrightarrow \end{aligned}$$

[1] These random variables are not really independent. However, if their variances were small enough we could neglect their covariance. The Kalman filter is responsible for minimizing the variance of such random variables, and we will rely on it doing a good job.

Isolating the $\dot{\boldsymbol{\delta}}$ quaternion,

$$\begin{aligned}
\dot{\boldsymbol{\delta}} &= \frac{1}{2}\underbrace{\overline{\boldsymbol{q}}^* * \boldsymbol{q}}_{\boldsymbol{\delta}} * \boldsymbol{\omega}' - \frac{1}{2}\underbrace{\overline{\boldsymbol{q}}^* * \overline{\boldsymbol{q}}}_{\mathbf{1}} * \overline{\boldsymbol{\omega}}' * \boldsymbol{\delta} = \\
&= \frac{1}{2}\left[\boldsymbol{\delta} * \boldsymbol{\omega}' - \overline{\boldsymbol{\omega}}' * \boldsymbol{\delta}\right] = \\
&= \frac{1}{2}\left[\begin{pmatrix}\delta_0\\ \boldsymbol{\delta}\end{pmatrix} * \begin{pmatrix}0\\ \boldsymbol{\omega}'\end{pmatrix} - \begin{pmatrix}0\\ \overline{\boldsymbol{\omega}}'\end{pmatrix} * \begin{pmatrix}\delta_0\\ \boldsymbol{\delta}\end{pmatrix}\right] = \\
&= \frac{1}{2}\begin{pmatrix}-(\boldsymbol{\omega}' - \overline{\boldsymbol{\omega}}')\cdot\boldsymbol{\delta}\\ \delta_0\,(\boldsymbol{\omega}' - \overline{\boldsymbol{\omega}}') - (\boldsymbol{\omega}' + \overline{\boldsymbol{\omega}}')\times\boldsymbol{\delta}\end{pmatrix} = \\
&= \frac{1}{2}\begin{pmatrix}-\Delta\boldsymbol{\omega}'\cdot\boldsymbol{\delta}\\ \delta_0\,\Delta\boldsymbol{\omega}' - (2\,\overline{\boldsymbol{\omega}}' + \Delta\boldsymbol{\omega}')\times\boldsymbol{\delta}\end{pmatrix} . \qquad (59)
\end{aligned}$$

b) Differential equations for $\mathbf{e}$ *on the chart:* Using (4) and (59),

$$\begin{aligned}
\dot{\mathbf{e}} &= 2\,\frac{\dot{\boldsymbol{\delta}}\,\delta_0 - \dot{\delta}_0\,\boldsymbol{\delta}}{\delta_0^2} = \\
&= \Delta\boldsymbol{\omega}' - (2\,\overline{\boldsymbol{\omega}}' + \Delta\boldsymbol{\omega}')\times\frac{\mathbf{e}}{2} + \left[\Delta\boldsymbol{\omega}'\cdot\frac{\mathbf{e}}{2}\right]\frac{\mathbf{e}}{2} ,
\end{aligned}$$

or in matrix form,

$$\dot{\mathbf{e}} = \Delta\boldsymbol{\omega}' - [\,2\,\overline{\boldsymbol{\omega}}' + \Delta\boldsymbol{\omega}'\,]_\times\,\frac{\mathbf{e}}{2} + \frac{\mathbf{e}}{2}\,\frac{\mathbf{e}^T}{2}\,\Delta\boldsymbol{\omega}' . \qquad (60)$$

2) Differential ecuations for the covariance matrix:

a) Evolution equation for $\mathbf{P}^{\omega\omega}$*:*

$$\begin{aligned}
\mathbf{P}_t^{\omega\omega} &= E\left[\,\Delta\boldsymbol{\omega}'_t\,\Delta\boldsymbol{\omega}'^T_t\,\right] = \\
&= E\left[\,(\boldsymbol{\omega}'_t - \overline{\boldsymbol{\omega}}'_t)\,(\boldsymbol{\omega}'_t - \overline{\boldsymbol{\omega}}'_t)^T\,\right] = \\
&= E\left[\,(\Delta\boldsymbol{\omega}'_{t\text{-}\Delta t} + \Delta\mathbf{q}_t^{\omega})\,(\Delta\boldsymbol{\omega}'_{t\text{-}\Delta t} + \Delta\mathbf{q}_t^{\omega})^T\,\right] = \\
&= \underbrace{E\left[\,\Delta\boldsymbol{\omega}'_{t\text{-}\Delta t}\,\Delta\boldsymbol{\omega}'^T_{t\text{-}\Delta t}\,\right]}_{\mathbf{P}^{\omega\omega}_{t\text{-}\Delta t}} + E\left[\,\Delta\boldsymbol{\omega}'_{t\text{-}\Delta t}\,\Delta\mathbf{q}_t^{\omega T}\right] + \\
&\quad + E\left[\,\Delta\mathbf{q}_t^{\omega}\,\Delta\boldsymbol{\omega}'^T_{t\text{-}\Delta t}\,\right] + \overbrace{E\left[\,\Delta\mathbf{q}_t^{\omega}\,\Delta\mathbf{q}_t^{\omega T}\,\right]}^{\mathbf{Q}_t^{\omega}} .
\end{aligned}$$

Assuming the random variables $\boldsymbol{\omega}'_{t\text{-}\Delta t}$ and $\mathbf{q}_t^{\omega}$ to be independents, the covariance between their errors is null. In such case,

$$\mathbf{P}_t^{\omega\omega} = \mathbf{P}_{t\text{-}\Delta t}^{\omega\omega} + \mathbf{Q}_t^{\omega} . \qquad (61)$$

b) Differential equation for $\mathbf{P}^{ee}$*:* Replacing (13) in (16) we obtain,

$$\begin{aligned}
\dot{\mathbf{P}}^{ee} &= E\Big[\,\Delta\boldsymbol{\omega}'\,\mathbf{e}^T\Big] + \frac{1}{4}\,E\Big[\,\mathbf{e}\,\mathbf{e}^T\Delta\boldsymbol{\omega}'\,\mathbf{e}^T\Big] + \\
&\quad - [\,\overline{\boldsymbol{\omega}}'\,]_\times\,E\Big[\,\mathbf{e}\,\mathbf{e}^T\Big] - \frac{1}{2}\,E\Big[\,[\Delta\boldsymbol{\omega}']_\times\,\mathbf{e}\,\mathbf{e}^T\Big] + \\
&\quad + E\Big[\,\mathbf{e}\,\Delta\boldsymbol{\omega}'^T\Big] + \frac{1}{4}\,E\Big[\,\mathbf{e}\,\Delta\boldsymbol{\omega}'^T\,\mathbf{e}\,\mathbf{e}^T\Big] + \\
&\quad - E\Big[\,\mathbf{e}\,\mathbf{e}^T\Big]\,[\,\overline{\boldsymbol{\omega}}'\,]_\times^T - \frac{1}{2}\,E\Big[\,\mathbf{e}\,\mathbf{e}^T\,[\Delta\boldsymbol{\omega}']_\times^T\Big] .
\end{aligned}$$

Here we can see the consequences of treating a nonlinear system. The evolution in the covariance matrix $\mathbf{P}^{ee}$, which is composed by moments of second order, is affected by the higher moments of the distribution.

To find the evolution equations of the covariance matrix we would need information about the moments of order 3 and 4. These may depend of moments of order higher than them. Knowing all the moments of a distribution would mean to know all statistical information. What we can assume and expect is that higher moments to be negligible compared to second-order moments. In that case we can write,

$$\begin{aligned}
\dot{\mathbf{P}}^{ee} &\approx E\Big[\,\Delta\boldsymbol{\omega}'\,\mathbf{e}^T\Big] - [\,\overline{\boldsymbol{\omega}}'\,]_\times\,E\Big[\,\mathbf{e}\,\mathbf{e}^T\Big] + \\
&\quad + E\Big[\,\mathbf{e}\,\Delta\boldsymbol{\omega}'^T\Big] - E\Big[\,\mathbf{e}\,\mathbf{e}^T\Big]\,[\,\overline{\boldsymbol{\omega}}'\,]_\times^T = \\
&= \left(\mathbf{P}^{e\omega}\right)^T - [\,\overline{\boldsymbol{\omega}}'\,]_\times\,\mathbf{P}^{ee} + \\
&\quad + \mathbf{P}^{e\omega} - \mathbf{P}^{ee}\,[\,\overline{\boldsymbol{\omega}}'\,]_\times^T . \qquad (62)
\end{aligned}$$

c) Differential equation for $\mathbf{P}^{e\omega}$*:* Replacing (13) in (17) we obtain,

$$\begin{aligned}
\dot{\mathbf{P}}^{e\omega} &= E\Big[\,\Delta\boldsymbol{\omega}'\,\Delta\boldsymbol{\omega}'^T\Big] + \frac{1}{4}\,E\Big[\,\mathbf{e}\,\mathbf{e}^T\,\Delta\boldsymbol{\omega}'\,\Delta\boldsymbol{\omega}'^T\Big] + \\
&\quad - [\,\overline{\boldsymbol{\omega}}'\,]_\times\,E\Big[\,\mathbf{e}\,\Delta\boldsymbol{\omega}'^T\Big] + \\
&\quad - \frac{1}{2}\,E\Big[\,[\Delta\boldsymbol{\omega}']_\times\,\mathbf{e}\,\Delta\boldsymbol{\omega}'^T\Big] + E\Big[\,\mathbf{e}\,\Delta\widetilde{\boldsymbol{\tau}}'^T\Big] .
\end{aligned}$$

Assuming the independence of the random variables $\mathbf{e}$ and $\Delta\widetilde{\boldsymbol{\tau}}'$, and neglecting the higher order moments,

$$\begin{aligned}
\dot{\mathbf{P}}^{e\omega} &\approx E\Big[\,\Delta\boldsymbol{\omega}'\,\Delta\boldsymbol{\omega}'^T\Big] - [\,\overline{\boldsymbol{\omega}}'\,]_\times\,E\Big[\,\mathbf{e}\,\Delta\boldsymbol{\omega}'^T\Big] = \\
&= \mathbf{P}^{\omega\omega} - [\,\overline{\boldsymbol{\omega}}'\,]_\times\,\mathbf{P}^{e\omega} . \qquad (63)
\end{aligned}$$

3) Evolution equations for the covariance matrix: Denoting $\boldsymbol{\Omega} = [\,\overline{\boldsymbol{\omega}}'_t\,]_\times$ our differential equations take the form:

$$\mathbf{P}_t^{\omega\omega} = \mathbf{P}_{t\text{-}\Delta t}^{\omega\omega} + \mathbf{Q}_t^{\omega} , \qquad (64)$$

$$\dot{\mathbf{P}}^{e\omega}(\tau) \approx \mathbf{P}^{\omega\omega}(\tau) - \boldsymbol{\Omega}\,\mathbf{P}^{e\omega}(\tau) , \qquad (65)$$

$$\begin{aligned}
\dot{\mathbf{P}}^{ee}(\tau) &\approx \left(\mathbf{P}^{e\omega}(\tau)\right)^T - \boldsymbol{\Omega}\,\mathbf{P}^{ee}(\tau) + \qquad (66)\\
&\quad + \mathbf{P}^{e\omega}(\tau) - \mathbf{P}^{ee}(\tau)\,\boldsymbol{\Omega}^T . \qquad (67)
\end{aligned}$$

a) Evolution equation for $\mathbf{P}^{e\omega}$*:*

First of all, let us consider the homogeneous equation of (65),

$$\dot{\mathbf{P}}^{e\omega}(\tau) = -\boldsymbol{\Omega}\,\mathbf{P}^{e\omega}(\tau) . \qquad (68)$$

The solution for (68) is

$$\mathbf{P}^{e\omega}(\tau) = e^{-\boldsymbol{\Omega}\,\tau}\,\mathbf{P}^{e\omega}(0) .$$

To find the solution to the inhomogeneous differential equation, we use the *variation of constants method*:

$$\begin{aligned}\mathbf{P}^{e\omega}(\tau) &= e^{-\mathbf{\Omega}\,\tau}\;\mathbf{C}(\tau) \qquad \Longrightarrow \\ \dot{\mathbf{P}}^{e\omega}(\tau) &= -\,\mathbf{\Omega}\,e^{-\mathbf{\Omega}\,\tau}\;\mathbf{C}(\tau) \;+\; e^{-\mathbf{\Omega}\,\tau}\;\dot{\mathbf{C}}(\tau) \;= \\ &= -\,\mathbf{\Omega}\,\mathbf{P}^{e\omega}(\tau) \;+\; e^{-\mathbf{\Omega}\,\tau}\;\dot{\mathbf{C}}(\tau) \quad .\end{aligned}$$

Identifying the terms in the differential equation we obtain the following relation:

$$\begin{aligned}e^{-\mathbf{\Omega}\,\tau}\;\dot{\mathbf{C}}(\tau) &= \mathbf{P}^{\omega\omega}(\tau) \qquad \Longrightarrow \\ \Longrightarrow \qquad \dot{\mathbf{C}}(\tau) &= e^{\mathbf{\Omega}\,\tau}\;\mathbf{P}^{\omega\omega}(\tau) \quad . \end{aligned} \tag{69}$$

To solve this last differential equation, it is necessary to propose a continuous evolution equation for $\mathbf{P}^{\omega\omega}$. The simplest option, that still satisfies the equation of discrete evolution (15) is the linear function

$$\mathbf{P}^{\omega\omega}(\tau) = \mathbf{P}^{\omega\omega}(0) \;+\; \frac{\tau}{\Delta t}\,\mathbf{Q}_t^{\omega} \quad .$$

Having defined this continous evolution equation, (69) is transformed into

$$\begin{aligned}\dot{\mathbf{C}}(\tau) &= e^{\mathbf{\Omega}\,\tau}\;\left[\,\mathbf{P}^{\omega\omega}(0) \;+\; \frac{\tau}{\Delta t}\,\mathbf{Q}_t^{\omega}\,\right] \;= \\ &= \left(\sum_{n=0}^{\infty}\frac{\mathbf{\Omega}^n\,\tau^n}{n!}\right)\;\left[\,\mathbf{P}^{\omega\omega}(0) \;+\; \frac{\tau}{\Delta t}\,\mathbf{Q}_t^{\omega}\,\right] \quad .\end{aligned} \tag{70}$$

Integrating (70),

$$\begin{aligned}\mathbf{C}(\tau) &= \left(\sum_{n=0}^{\infty}\frac{\mathbf{\Omega}^n\,\tau^{n+1}}{(n+1)!}\right)\;\mathbf{P}^{\omega\omega}(0) \;+ \\ &+\; \frac{1}{\Delta t}\;\left(\sum_{n=0}^{\infty}\frac{\mathbf{\Omega}^n\,\tau^{n+2}}{n!\,(n+2)}\right)\;\mathbf{Q}_t^{\omega} \;+\; \mathbf{C}' \quad .\end{aligned}$$

The constant $\mathbf{C}'$ is determined by the initial conditions $\mathbf{P}^{e\omega}(0) = \mathbf{C}(0)$. With this in mind,

$$\begin{aligned}\mathbf{P}^{e\omega}(\tau) &= e^{-\mathbf{\Omega}\,\tau}\;\left[\;\left(\sum_{n=0}^{\infty}\frac{\mathbf{\Omega}^n\,\tau^{n+1}}{(n+1)!}\right)\;\mathbf{P}^{\omega\omega}(0) \;+\right. \\ &\left.+\;\frac{1}{\Delta t}\;\left(\sum_{n=0}^{\infty}\frac{\mathbf{\Omega}^n\,\tau^{n+2}}{n!\,(n+2)}\right)\;\mathbf{Q}_t^{\omega} \;+\; \mathbf{P}^{e\omega}(0)\right] \quad .\end{aligned} \tag{71}$$

Knowing that we have the information at $t-\Delta t$, and we want to update the information at t, our equation becomes

$$\begin{aligned}\mathbf{P}^{e\omega}_{t|t\text{-}\Delta t} &= e^{-\mathbf{\Omega}\,\Delta t}\;\left[\;\left(\sum_{n=0}^{\infty}\frac{\mathbf{\Omega}^n\;(\Delta t)^{n+1}}{(n+1)!}\right)\;\mathbf{P}^{\omega\omega}_{t\text{-}\Delta t|t\text{-}\Delta t} \;+\right. \\ &\left.+\;\left(\sum_{n=0}^{\infty}\frac{\mathbf{\Omega}^n\;(\Delta t)^{n+1}}{n!\,(n+2)}\right)\;\mathbf{Q}_t^{\omega} \;+\; \mathbf{P}^{e\omega}_{t\text{-}\Delta t|tdt}\right] \quad .\end{aligned} \tag{72}$$

Finally, knowing that calculating infinite sums would take a lot, we can truncate in the first term, and write

$$\begin{aligned}\mathbf{P}^{e\omega}_{t|t\text{-}\Delta t} &\approx e^{-\mathbf{\Omega}\,\Delta t}\;\left[\;\mathbf{P}^{e\omega}_{t\text{-}\Delta t|t\text{-}\Delta t} \;+\right. \\ &\left.+\;\Delta t\;\mathbf{P}^{\omega\omega}_{t\text{-}\Delta t|t\text{-}\Delta t} \;+\; \frac{\Delta t}{2}\;\mathbf{Q}_t^{\omega}\;\right] \quad .\end{aligned} \tag{73}$$

b) Evolution equation for $\mathbf{P}^{ee}$:
In this case we have the homogeneous equation

$$\dot{\mathbf{P}}^{ee}(\tau) \approx -\,\mathbf{\Omega}\,\mathbf{P}^{ee}(\tau) \;-\; \mathbf{P}^{ee}(\tau)\,\mathbf{\Omega}^T \quad ,$$

whose solution is

$$\mathbf{P}^{ee}(\tau) = e^{-\mathbf{\Omega}\,\tau}\;\mathbf{P}^{ee}(0)\;e^{-\mathbf{\Omega}^T\,\tau} \quad .$$

Using the *variation of constants method*,

$$\begin{aligned}\mathbf{P}^{ee}(\tau) &= e^{-\mathbf{\Omega}\,\tau}\;\mathbf{C}(\tau)\;e^{-\mathbf{\Omega}^T\,\tau} \qquad \Longrightarrow \\ \Longrightarrow \qquad \dot{\mathbf{P}}^{ee}(\tau) &= -\mathbf{\Omega}\,\mathbf{P}^{ee}(\tau) \;+ \\ &+\; e^{-\mathbf{\Omega}\,\tau}\;\dot{\mathbf{C}}(\tau)\;e^{-\mathbf{\Omega}^T\tau} \;+ \\ &-\; \mathbf{P}^{ee}(\tau)\;\mathbf{\Omega}^T \quad .\end{aligned}$$

Identifying terms, we deduce the relation

$$\dot{\mathbf{C}}(\tau) = e^{\mathbf{\Omega}\,\tau}\;\left[\,\mathbf{P}^{e\omega}(\tau) \;+\; \left(\mathbf{P}^{e\omega}(\tau)\right)^T\,\right]\;e^{\mathbf{\Omega}^T\,\tau} \quad .$$

After substituting the expression for $\mathbf{P}^{e\omega}$, integrate with respect to time, and truncating in the first term of the infinite sums, the solution we want is given by

$$\begin{aligned}\mathbf{P}^{ee}_{t|t\text{-}\Delta t} &= e^{-\mathbf{\Omega}\,\Delta t}\;\left[\;\mathbf{P}^{ee}_{t\text{-}\Delta t|t\text{-}\Delta t} \;+\right. \\ &+\; (\Delta t)^2\;\mathbf{P}^{\omega\omega}_{t\text{-}\Delta t|t\text{-}\Delta t} \;+\; \frac{(\Delta t)^2}{3}\;\mathbf{Q}_t^{\omega} \;+ \\ &+\; \mathbf{P}^{e\omega}_{t\text{-}\Delta t|t\text{-}\Delta t}\;\Delta t \;+ \\ &\left.+\;\left(\mathbf{P}^{e\omega}_{t\text{-}\Delta t|t\text{-}\Delta t}\right)^T\;\Delta t\right]\;e^{-\mathbf{\Omega}^T\,\Delta t} \quad .\end{aligned}$$

APPENDIX B
PREDICTION OF THE MEASURE

A. Expected value of the measurement

1) Expected value of the accelerometer measurement: Taking the expected value on (9),

$$\bar{\mathbf{a}}_t^m = E\left[\,\mathbf{R}_t^T\,\left(\,\mathbf{a}_t \;-\; \bar{\mathbf{g}}_t\,\right)\,\right] \;+\; \bar{\mathbf{r}}_t^a \quad .$$

The $\mathbf{g}_t$ vector does not change. If we also assume that the accelerations affecting our system, $\mathbf{a}_t$, does not depend on its orientation,

$$\bar{\mathbf{a}}_t^m = E\left[\,\mathbf{R}_t^T\,\right]\;\left(\,\bar{\mathbf{a}}_t \;-\; \bar{\mathbf{g}}_t\,\right) \;+\; \bar{\mathbf{r}}_t^a \quad .$$

One might be tempted to try to find the expected value of the matrix $\mathbf{R}_t$ written as a function of the $\boldsymbol{q}_t$ quaternion, but then we would run into the problem of the computation of expected values such as $E[q_1^2]$ or $E[q_1\,q_2]$. These expected

values are defined in the manifold, and are what we try to avoid defining covariance matrices in the charts.
What we seek is not the expected value of the rotation matrix, but something like the "expected transformation". Then, using the quaternion describing the orientation of our system, this expression is equivalent to

$$\overline{\mathbf{a}}_t^m \;=\; \overline{\boldsymbol{q}}_t^* * \left(\overline{\boldsymbol{a}}_t \,-\, \overline{\boldsymbol{g}}_t\right) * \overline{\boldsymbol{q}}_t \;+\; \overline{\mathbf{r}}_t^a \quad .$$

B. Covariance matrix of the measurement

1) Accelerometer block: Knowing that

$$\begin{aligned}\mathbf{h}_a(\mathbf{e}_t, \mathbf{a}_t) \;&=\; \boldsymbol{q}^* * (\boldsymbol{a}_t - \overline{\boldsymbol{g}}_t) * \boldsymbol{q} \;=\\ &=\; \boldsymbol{\delta}^* * \overline{\boldsymbol{q}}^* * (\mathbf{a}_t - \overline{\mathbf{g}}_t) * \overline{\boldsymbol{q}} * \boldsymbol{\delta} \;=\\ &=\; \left(\mathbf{R}_t^{\delta}\right)^T \overline{\mathbf{R}}_t^T \;(\mathbf{a}_t - \overline{\mathbf{g}}_t) \quad ,\end{aligned}$$

we will have,

$$\left.\frac{\partial\, \mathbf{h}_a}{\partial \mathbf{a}_t}\right|_{\substack{\mathbf{e}_t=\mathbf{0}\\ \mathbf{a}_t=\overline{\mathbf{a}}_t}} = \overline{\mathbf{R}}_t^T \quad ,$$

$$\left.\frac{\partial\, \mathbf{h}_a}{\partial \mathbf{e}_t}\right|_{\substack{\mathbf{e}_t=\mathbf{0}\\ \mathbf{a}_t=\overline{\mathbf{a}}_t}} = \left.\frac{\partial \left(\mathbf{R}_t^{\delta}\right)^T}{\partial \boldsymbol{\delta}_t}\right|_{\boldsymbol{\delta}=\mathbf{1}} \left.\frac{\partial\, \boldsymbol{\delta}_t}{\partial \mathbf{e}_t}\right|_{\mathbf{e}_t=\mathbf{0}} \overline{\mathbf{R}}_t^T \;(\overline{\mathbf{a}}_t - \overline{\mathbf{g}}_t) \quad .$$

Let us note that $\frac{\partial\, \boldsymbol{\delta}_t}{\partial \mathbf{e}_t} \in \mathbb{R}^{4\times 3}$. However, the term $\frac{\partial \left(\mathbf{R}_t^{\delta}\right)^T}{\partial \boldsymbol{\delta}_t}$ is a 1×4 matrix, whose elements are again 3×3 matrices.

a) First term:
Given the expression for the rotation matrix corresponding to quaternion $\boldsymbol{\delta}$,

$$\left(\mathbf{R}^{\delta}\right)^T = \begin{pmatrix} 1-2\delta_2^2-2\delta_3^2 & 2(\delta_1\delta_2+\delta_3\delta_0) & 2(\delta_1\delta_3-\delta_2\delta_0)\\ 2(\delta_1\delta_2-\delta_3\delta_0) & 1-2\delta_1^2-2\delta_3^2 & 2(\delta_2\delta_3+\delta_1\delta_0)\\ 2(\delta_1\delta_3+\delta_2\delta_0) & 2(\delta_2\delta_3-\delta_1\delta_0) & 1-2\delta_1^2-2\delta_2^2 \end{pmatrix} \quad ,$$

we can derive and evaluate at $\boldsymbol{\delta}=\mathbf{1}=(1,0,0,0)$, resulting

$$\left.\frac{\partial \left(\mathbf{R}^{\delta}\right)^T}{\partial \delta_0}\right|_{\boldsymbol{\delta}=\mathbf{1}} = \begin{pmatrix} 0&0&0\\0&0&0\\0&0&0\end{pmatrix} \quad ,$$

$$\left.\frac{\partial \left(\mathbf{R}^{\delta}\right)^T}{\partial \delta_1}\right|_{\boldsymbol{\delta}=\mathbf{1}} = \begin{pmatrix} 0&0&0\\0&0&2\\0&-2&0\end{pmatrix} \quad ,$$

$$\left.\frac{\partial \left(\mathbf{R}^{\delta}\right)^T}{\partial \delta_2}\right|_{\boldsymbol{\delta}=\mathbf{1}} = \begin{pmatrix} 0&0&-2\\0&0&0\\2&0&0\end{pmatrix} \quad ,$$

$$\left.\frac{\partial \left(\mathbf{R}^{\delta}\right)^T}{\partial \delta_3}\right|_{\boldsymbol{\delta}=\mathbf{1}} = \begin{pmatrix} 0&2&0\\-2&0&0\\0&0&0\end{pmatrix} \quad .$$

b) Second term:
For the charts defined by (4), we can recover the quaternion by (5). Then,

$$\frac{\partial\, \boldsymbol{\delta}}{\partial \mathbf{e}} \;=\; \frac{1}{(4+|\mathbf{e}|^2)^{3/2}} \begin{pmatrix} -2\,e_1 & -2\,e_2 & -2\,e_3\\ 4+e_2^2+e_3^2 & -e_1\,e_2 & -e_1\,e_3\\ -e_2\,e_1 & 4+e_1^2+e_3^2 & -e_2\,e_3\\ -e_3\,e_1 & -e_3\,e_2 & 4+e_1^2+e_2 \end{pmatrix} \quad .$$

And evaluating at $\mathbf{e}=\mathbf{0}$,

$$\left.\frac{\partial\, \boldsymbol{\delta}}{\partial \mathbf{e}}\right|_{\mathbf{e}=\mathbf{0}} \;=\; \frac{1}{2}\begin{pmatrix} 0&0&0\\1&0&0\\0&1&0\\0&0&1\end{pmatrix} \quad .$$

c) The matrix:
Denoting $\overline{\mathbf{g}}_t^{\mathbf{R}} = \overline{\mathbf{R}}_t^T(\overline{\mathbf{a}} - \overline{\mathbf{g}})$, and computing the matrix products,

$$\left.\frac{\partial\, \mathbf{h}_a(\mathbf{e}_t)}{\partial \mathbf{e}_t}\right|_{\substack{\mathbf{e}_t=\mathbf{0}\\ \mathbf{a}_t=\overline{\mathbf{a}}_t}} = \begin{pmatrix} 0 & -\left(\overline{\mathbf{g}}_t^{\mathbf{R}}\right)_3 & \left(\overline{\mathbf{g}}_t^{\mathbf{R}}\right)_2\\ \left(\overline{\mathbf{g}}_t^{\mathbf{R}}\right)_3 & 0 & -\left(\overline{\mathbf{g}}_t^{\mathbf{R}}\right)_1\\ -\left(\overline{\mathbf{g}}_t^{\mathbf{R}}\right)_2 & \left(\overline{\mathbf{g}}_t^{\mathbf{R}}\right)_1 & 0 \end{pmatrix} \quad .$$

APPENDIX C
UPDATE

A. Chart update

Differentiating our transition map,

$$\begin{aligned}\frac{\partial\, (\mathbf{e}_p)_i}{\partial\, (\mathbf{e}_q)_\ell} \;&=\; 2\,\frac{\overline{\delta}_0\,\mathbf{1} \,-\, \left[\boldsymbol{\delta}\right]_\times}{2\,\overline{\delta}_0 \,+\, \overline{\boldsymbol{\delta}}\cdot\mathbf{e}_q} \,+\\ &\quad -\, 2\,\frac{\overline{\delta}_0\,\mathbf{e}_q \,-\, 2\,\overline{\boldsymbol{\delta}} \,-\, \overline{\boldsymbol{\delta}}\times\mathbf{e}_q}{\left(2\,\overline{\delta}_0 \,+\, \overline{\boldsymbol{\delta}}\cdot\mathbf{e}_q\right)^2}\,\overline{\boldsymbol{\delta}}^T \quad .\end{aligned}$$

Now, if $\varphi_{\overline{\boldsymbol{p}}} = \varphi_{\overline{\boldsymbol{q}}_{t|t}}$, and if $\varphi_{\overline{\boldsymbol{q}}} = \varphi_{\overline{\boldsymbol{q}}_{t|t-\Delta t}}$, and we are interested in expressing the covariance matrix in the chart $\varphi_{\overline{\boldsymbol{q}}_{t|t}}$, from the covariance matrix expressed in the chart $\varphi_{\overline{\boldsymbol{q}}_{t|t-\Delta t}}$, where $\overline{\mathbf{e}}_q = \overline{\mathbf{e}}_{t|t}$, then we must keep in mind that

$$\begin{aligned}\mathbf{e}_p(\overline{\mathbf{e}}_{t|t}) \;&=\; 2\,\frac{\overline{\delta}_0\,\overline{\mathbf{e}}_{t|t} \,-\, 2\,\overline{\boldsymbol{\delta}} \,-\, \overline{\boldsymbol{\delta}}\times\overline{\mathbf{e}}_{t|t}}{2\,\overline{\delta}_0 \,+\, \overline{\boldsymbol{\delta}}\cdot\overline{\mathbf{e}}_{t|t}} \;=\\ &=\; \varphi_{\mathbf{1}}\Big(\overline{\boldsymbol{\delta}}^* * \underbrace{\boldsymbol{\delta}(\overline{\mathbf{e}}_{t|t})}_{\overline{\boldsymbol{\delta}}}\Big) \;=\; \mathbf{0} \quad .\end{aligned}$$

Then replacing in (27),

$$\mathbf{e}_p(\mathbf{e}_q) \;\approx\; 2\,\frac{\overline{\delta}_0\,\mathbf{1} \,-\, \left[\overline{\boldsymbol{\delta}}\right]_\times}{2\,\overline{\delta}_0 \,+\, \overline{\boldsymbol{\delta}}\cdot\overline{\mathbf{e}}_{t|t}}\,\left(\mathbf{e}_q \,-\, \overline{\mathbf{e}}_{t|t}\right) \quad .$$

Finally, knowing that

$$2\,\overline{\delta}_0 \,+\, \overline{\boldsymbol{\delta}}\cdot\overline{\mathbf{e}}_{t|t} \;=\; \underbrace{\left(\overline{\boldsymbol{\delta}}^* * \boldsymbol{\delta}(\overline{\mathbf{e}}_{t|t})\right)_0}_{1}\,\sqrt{4 \,+\, |\overline{\mathbf{e}}_{t|t}|^2} \quad ,$$

we have

$$\begin{aligned}\mathbf{e}_p(\mathbf{e}_q) \;&\approx\; \frac{2}{\sqrt{4+|\overline{\mathbf{e}}_{t|t}|^2}}\left(\overline{\delta}_0\,\mathbf{1} - \left[\overline{\boldsymbol{\delta}}\right]_\times\right)\left(\mathbf{e}_q - \overline{\mathbf{e}}_{t|t}\right) \;=\\ &=\; \overline{\delta}_0\left(\overline{\delta}_0\,\mathbf{1} \,-\, \left[\overline{\boldsymbol{\delta}}\right]_\times\right)\left(\mathbf{e}_q \,-\, \overline{\mathbf{e}}_{t|t}\right) \quad .\end{aligned}$$

Predicting Battery Level Analysing the Behaviour of Mobile Robot

Pragna Das and Lluís Ribas-Xirgo

***Abstract*—In automated manufacturing systems based on internal transportation, there is much importance of battery powered Automated Guided Vehicles or Mobile Robots. In general, while the whole multi-robot system is functioning, there is no possibility to infer about the current battery capacity of each mobile robot. However, the current battery capacity is important as a system information, to make more efficient decisions for the functioning of the robots. This work proposes a method to predict the battery capacities of each robot during run-time of the system. Also, it is inferred in this work that these prediction values of battery power of each robot is beneficial to have a cost effective system performance, in multi-robot scenario.**

***Index Terms*—Battery powered Automated Guided Vehicles, Internal Transportation Systems, Flexible Manufacturing Units, Logistics, Estimation, Cost Efficient Performance, Multi-Agent Systems**

I. Introduction

IN internal transportation system based flexible manufacturing plants, warehouses and logistic hubs, there is a necessity to have adaptable and evolving control system. In general, conveyor belts were the common option for internal transportation. Due to technical advancement and sophisticated machinery, mass production approaches are changed to small lot sizes, leading to constant changes in the production operation sequences and increase in the complexity of factory floors. To adopt to the changes in production style evolving with the sophisticated control architecture and strategies, conveyors were replaced mostly by Automated Guided Vehicles (AGVs) [1]. The un-manned vehicles operating on the factory floor for material transportation or as mobile production platform are generally termed as AGVs. They are mostly managed by humans relying on sophisticated control systems. With the use of AGVs, the production process has become more flexible with decrease in production time and costs. Also, AGVs consume less space and can accommodate new routes with relative ease [2]. So, these increase the demand of AGVs in internal transportation system and automation industry in general [3].

There has been extensive research work on these application areas of embedded systems using AGVs based on computational intelligence for internal transportation system [4], [5]. For applications of transportation and material handling in manufacturing and logistics plants using AGVs, the trends are to use driver-less diesel or electric powered AGVs. However,

Pragna Das is in Autonomous University of Barcelona

E-mail: pragna.das@uab.cat
Lluís Ribas-Xirgo is in Autonomous University of Barcelona

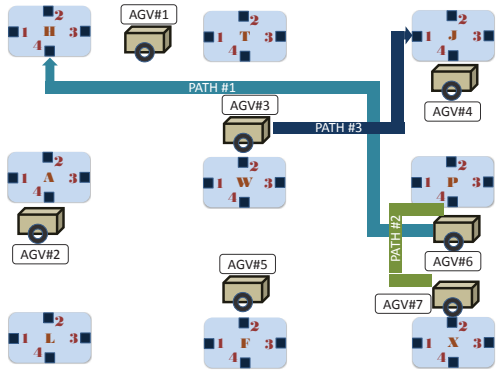


Fig. 1: An Example Scenario

according to [6], the use of battery powered AGVs have economic, technical and ecological advantages over diesel-electric powered AGVs in applications of transportation and material handling. In [6] *Schmidt et. al.* has investigated the commercial viability of Battery powered AGVs (B-AGVs) and has concluded that the battery mobility is economically beneficial because the charging and maintenance costs of a group of B-AGVs are significantly lower. Moreover, the cost of maintaining B-AGVs can be instrumental in requiting the higher investment costs of procuring charging infrastructure and spare batteries.

In manufacturing units and logistics hubs, the use of batteries play a major role in the overall AGV performance as a team. There are various scenarios where the knowledge of when to re-charge the batteries of any particular AGV or the current battery level during run-time of the system can be utilised efficiently to optimally use the performing cost of the system.

To have a comprehensive idea, we provide the following example: Consider Fig 1, where an illustration of a system of multiple transportation AGVs are shown. For example, let AGV#6 has to travel from Port P#4 to Port H#4 along PATH#1. There can be a situation where AGV#6 has exhausted the battery in its previous task(s) so much that it will take a very long time now to load at Port P#4 and traverse the distance from Port P#4 to Port H#4 taking the load. Hence, it can be a wiser decision to deploy AGV#7 which has more battery power than AGV#6 to take Path#2 to load at Port P#4 and traverse the distance to Port H#4 to carry that load. Moreover, AGV#3 taking the PATH#3 can now have a collision with AGV#6 as AGV #6 is delayed.

The above situations are arising due to the lack of knowledge about the current battery level at run-time leading to incorrect prediction about performance capability of the AGVs. In this work we investigate on the possibility of obtaining this information and effectively using them to take more optimal control decisions.

Though, charging the batteries at regular time interval of time (essential for the longevity of the batteries) can help to mitigate the above situation, frequent charging increases the cost of operation of the AGVs [7]. Moreover, even with charging at regular interval, the above situations cannot be entirely avoided to serve the need of production systems with growing flexibility.

Normally state of the art transportation systems [8] take transport decisions irrespective of the battery efficiency level of the AGVs and hence the decisions for the task allocation and movement can eventually lead to increase of overall cost of performing the tasks.

However it will be beneficial, in terms of cost, when continuously battery profile is used for taking control decisions. But, for this decisions to take, we need to know how much is the level of battery efficiency at the onset of performing the task. However, there is no direct method to measure the battery efficiency of the AGVs after completing each task.

Though the above example is used to demonstrate the fact that battery information is indeed useful for effective decision making of AGVs in case of path planning but this is not limited to scenarios arising out of path planning. Similar examples can be thought of in diverse application domains (e.g heavy weight lifting, enabling a handle, carrying material, *et cetera*). So, in this work our focus is not on effective path planning or any particular co-operative or co-ordinating function of multi-robot systems, rather on estimating the influence of battery level on performance of AGVs in general and how the battery level can be estimated by observing the AGV's performance. We also investigated how this information (battery discharge profile) modifies the system for more effective control.

So, the goal of this work is two fold:

1) We propose a scheme to estimate the battery level by parametrization of the performance of AGVs on battery level
2) We propose that this information can be efficiently used to control the system of AGVS

The rest of the paper is arranged as follows: First we describe the related research works focusing on same or similar problem and analysing them in Section II, then formally describe the problem in Section III, then in Section IV, we describe the experimentation platform used in this work. Section V re-formulates the problem in the view of our architecture. Some general methods for identifying and estimating are described in Section VI. In the following sections we present the experimental verification of our proposal before concluding with a discussion and further scope.

II. BACKGROUND

The recent research contributions in the area of battery power of AGVs where their re-charging problems and power management are addressed, are diversified into four major directions. One direction is estimating on-line the state of charge of the batteries of individual AGVs using Extended Kalman Filters [9], [10]. Second area of investigation is overall battery life cycle and power management [7] where simulations are done to evaluate battery related operational costs for individual AGVs under different functioning modes and to help develop a proper battery management strategy. The third major research area is the problems related to design and optimisation of charging and re-charging stations and their modes [11]. The fourth direction is tracking and prediction of real time robotic power for optimization of transportation of mobile robots or AGVs [12], which is similar to the problem which we are focused in this work.

From the research work on other three research directions, we can grasp that the problem of real time battery power prediction for optimised control of operations of AGVs or mobile robots is still less investigated as a research topic. We can site only one state of the art proposal in this direction. However as explained in Section I, this problem needs attention of research and there are major problems to be solved in this aspect.

In [12], *Liu, Stoll, Junginger and Thurow* have investigated on predicting the battery power of the mobile robots when the transportation request is made on an automated laboratory set-up. They have proposed a framework to measure the original power voltages and subsequently predicting them offline. When a request of transportation for the AGV is made, these offline predicted results are used to decide for the assignment of tasks.

Our work is different from the work of *Liu, Stoll, Junginger and Thurow* in the procedure of predicting the battery voltages and using them efficiently to make control decisions regarding the operation of AGVs. We are investigating on predicting the battery voltages on-line during the operation of the multi-robot system so that these predicted results can be used during the run time of the system at the onset of each task or destination traversal. Hence, this paper is focused on the on-line tracking and predicting of battery power of the AGVs in multi-robot systems used for automated transportation.

III. PROBLEM DEFINITION

From the controller perspective in AGV based transportation systems, there is a continuous process of fulfilling the order of assigned tasks to be completed properly. For the seamless flow of performing one task after another, the controllers are designed for a kind of mechanism which accomplish the decision-making based on current scenario of doing job and the status of the environment. However, while controlling and even allocating tasks and interleaving one or multiple task(s) and movement(s) in multi-robot transportation systems, the capability of performing the "to be assigned task or movement" before the beginning of it, is not considered. In our current work, we focus on this problem.

Let us consider a set of observed data for the time required for completing a distance traversal task by an AGV, which is considered as an example of the the task (Fig 2). The data is observed till the battery drains out completely and the AGV stops moving.

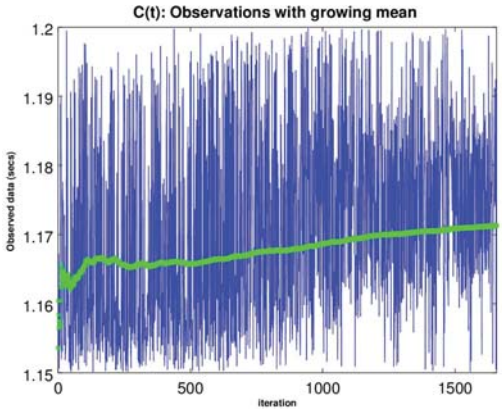


Fig. 2: Observed data for Distance Traversal

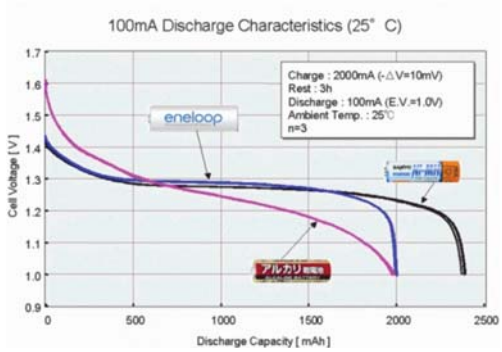


Fig. 3: Battery Discharge Profile

It is observable in Fig 2 that the values of the time required to reach a target distance (in general terms, performance ability) follow a steady rise during the initial stage, then follow a constant rising value for a span of time and then again rises up to a big magnitude before the robot finally stops moving. The green line shows the growing mean values over the entire course of traversing a particular distance till the battery exhausts completely.

In our current work, we are using ENELOOP batteries for operation of our in-house made AGVs. The battery discharge profile of ENELOOP batteries is illustrated in Figure 3.

Hence, the plot in Fig 2 shows that the task of reaching a target distance is influenced by the battery discharge profile (Figure 3), where the state of charge of the batteries used in the work is in parity with the observation data of traversal time. When observing the green line in Fig 2, it is evident that the growing mean of the values of traversal time is just inverse of the discharge profile of the batteries. Hence, the battery's state of charge can be predicted from the observed data of performing any task in a multi-robot system. Therefore, Fig 2 depicts that a battery discharge profile directly influences the performance of individual AGVs and also the overall system of AGVs [9].

If we can predict while completing one task, how the individual AGV and the system will behave during the next task, then dynamically better decision can be taken for the next assignment of task.

In our current work, we are investigating on the prediction of the battery capability of each AGV in transportation system at the onset of each task or each movement so that the controller is able to know whether the next assigned task or next movement will be accomplished successfully to the best possible way. We propose to analyse the current behaviour of the AGV during completion of the current task to understand its battery power for the next assigned task.

After we can successfully derive the predictions for battery power from the history and current performance capability of the AGVs, the system controller will be able to correctly decide on next assignment of task or next movement decision. This flexibility of decision of the controller is not the current concern of our work, however our current work leads to this direction.

IV. PLATFORM DESCRIPTION

To avoid a gap between the abstract idea and the actual implementation of the system, prototyping proves to be a beneficial approach for researches and other professionals [13]. Hence, to investigate on how to estimate the current battery power on-line, as suggested previously, a scaled prototype platform is required which consists of AGVs in the laboratory set-up. Although there are many techniques like PLC controllers, Petri-Net solutions for designing the control architecture of AGVs, Null-Space-Behavioral Control, we use Agent-Based Modelling as the basic control architecture for the individual AGVs for factors like better co-ordination and re-configurability, optimization, consensus and synchronization.

So in our experimentation platform, the individual AGVs are guided and monitored by their respective controllers. Also, the individual controllers of each AGV are equipped with mechanism to share data among other AGVs' controllers. In this work, a stacked controller model has been used to build a robust robotic system. Experiments will be performed on a realistic scenario that emulates a common workshop, with work and storage cells and pathways in between.

A. Scale prototype

The scaled prototype is made of a environment suited for material transportation, constructed out of mobile robots enabled to move in pathways in between to reach ports for loading and unloading. This is a typical replication of a transportation shop floor. As for the transportation agents, hobby robots made from the Boebot [14] are being used. The environment for the AGVs to navigate has been developed using uniform sized inexpensive boxes to build up labyrinth like pathways. There are marks in the boxes which can be identified as ports where the loading and unloading has to be done by the transportation agents (AGVs). The simulation of the above explained transportation floor is done in the V-Rep software also. In Figure 4, both the real and simulated platforms are shown.

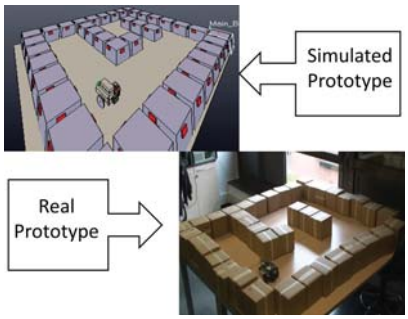


Fig. 4: The prototype platform

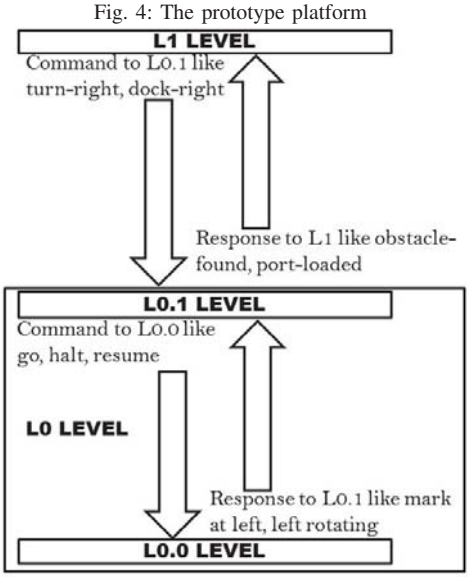


Fig. 5: Architecture of the controller

B. Controller architecture

We describe the controller architecture of a single AGV in this work as the controllers are similar for all AGVs in multi-agent systems. Moreover, there are scopes in our controller for communicating and sharing information between the AGVs [15]. This aspect provides an opportunity for sharing the status and knowledge of one AGV with another. The control structure consists of two major control layers (Figure 5). The top most layer is $L1$ level and the $L0$ level is below it. The $L0$ level is divided into two sub levels $L0.1$ and $L0.0$ levels respectively. The $L0$ level and $L1$ both functions on each of the transportation agents (AGVs) individually. Here, the $L0.0$ level can only communicate with the $L0.1$ level and can not have any direct communication with the $L1$ level. The $L0.1$ level is the intermediate level which communicates with both $L0.0$ level and $L1$ level. The control levels ($L1$ and $L0$) functions in co-ordination with each other to accomplish the decision making, guidance and monitoring of the AGVs in the environment [16].

The functions of controlling are more simplified in $L0.1$ and $L0.0$ levels. The $L1$ level is engaged in controlling more

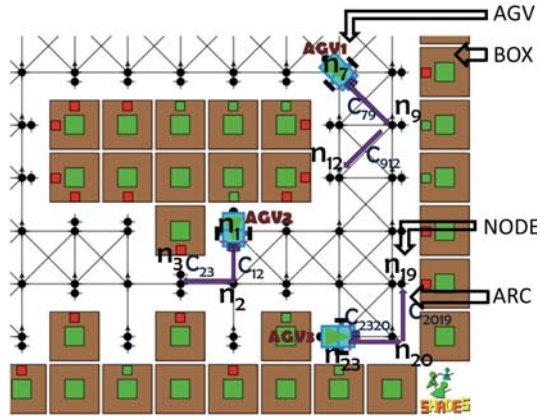


Fig. 6: The transportation floor

complex functions like path planning and finding destination poses for each of the robots. In this architecture, the inter-agent communications are done in the highest level ($L1$). Now, each of the robots in our scaled prototype platform have the $L0.0$ (implemented in the micro-controller board of the robot) and $L0.1$ (implemented in the mounted RaspberryPi in the robot) levels together implemented in them and each of the $L1$ is implemented in a personal computer (PC) with Internet access which has established connection between the different devices. Netlogo is used to synchronize the events happening in the real platform and the simulation at the same time.

C. Model

The transportation floor can be described in a topological view of the plant, which is a graph with nodes and arcs tagged with costs (time, energy).

The shop floor is designed as a graph whose simulation is done in a software. Figure 6 depicts this simulated shop floor. It is observed in Figure 6 that the nodes like n_3 denotes a valid port and n_2 denotes a bifurcation point where four possible arcs can be taken. The connecting lines (in violet color) between the two nodes n_3 and n_2 denotes an arc (a_{32}).

V. RE-FORMULATION OF THE PROBLEM IN THE LIGHT OF THE PROTOTYPE

Considering Figure 6, traversal of each arc in the shop floor to reach at destination port involves some cost expenditure in terms of energy exhaustion, dynamic obstacle, condition of the floor, load it carries, *et cetera*. The time to traverse an arc by an AGV (Figure 6) is thus conceptualised as some cost incurred for that robot. In terms, the battery charge is also expended to traverse the distance. Hence, we formulate this cost incurred in traversing an arc in the shop floor as $C(t, e, b, f, o)$, which is time-varying, where t is for time, b is for battery state of charge, e is for tire condition, f is for frictional force of the floor, o is for obstacle dependencies. $C(t, e, b, f, o)$ is time-varying from the perspective that at a particular instance of the time, the cost expenditure of that particular arc is dependent on battery discharge, condition of the floor or any dynamic

obstacle. Hence, for the arc a_{32} (denoted by violet line in Figure 6), the cost incurred is denoted by $C_{32}(t, e, b, f, o)$. Also, for the same robot, for the arc a_{12} (denoted by violet line in Figure 6), the cost incurred will be $C_{12}(t, e, b, f, o)$.

Now, a particular path will comprise of one or more arcs. Thus, for a particular path traversal, there are one or more of $C(t, e, b, f, o)$ for a particular AGV. Our work focuses on estimating all of such $C(t, e, b, f, o)$ to get the next prediction on the next time instance on-line. We call this cost incurred as a cost parameter. These estimations of cost incurred for each arc will designate the current battery discharge level. So, we can have a prediction of the battery charge required to accomplish the next task or to reach the next destination. Also, we can recursively update all the prediction values for all the $C(t, e, b, f, o)$ estimates till that time. This will eventually lead to implement a better control decisions like, which path to take or which task to assign.

VI. APPROACHES FOR IDENTIFICATION AND ESTIMATION

In our work, we are focused on estimation and prediction of on-line battery voltage power from the estimations of real-time performance capability of the AGVs. We interpret this performance capability of AGVs in the light of the cost incurred in completing one task or one movement as cost parameter.

In state of the art proposals, mostly the robot manipulator parameters [17], variance parameters for localisation [18] or parameters on model-based mobile robot controller [19] are investigated. We are estimating time-varying cost parameters which are different from the above parameters, in case of robotic systems.

For parameter identification and estimation, the standard methods include Least-Square Estimator approach [20], Least-Square Moving Window (LSMW) method [21], Recursive Least Square (RLS) method [21], Kalman Filtering (KF) [22], *et cetera*.

In [21], a LSMW method is suggested. Here, the data set (X, Y) of length L is such that,

$$Y = X\theta + W \tag{1}$$

where, X^T=(x_1,x_2,x_3,.....,x_L), Y^T=(y_1,y_2,y_3,.....,y_L) and W is the measurement noise.

Now, with a window size $l \in N$ such that $l < L$, the number of estimations of θ will be $L - l$ +1. The estimation is given by,

$$\hat{\theta}_i = X_i^{\#} Y_i \tag{2}$$

where,

$$X_i^{\#} = \left(X_i^T X_i\right)^{-1} X_i^T \tag{3}$$

and Y_i^T = (y_i, y_{i+1}, ..., y_{i+l-1}), X_i^T = (x_i, x_{i+1}, ..., x_{i+l-1}), i = 1,2,.....,$L - l + 1$ with the estimation error

$$\hat{e_i} = Y_i - X_i\hat{\theta}_i \tag{4}$$

Also, [21] has suggested a RLS algorithm based on both constant and variable forgetting factor. After the least square method, the estimate obtained at time t is

$$\hat{\theta}_t = \left(X_t^T X_t\right)^{-1} X_t^T Y_t \tag{5}$$

where, Y_t^T = (y_1, y_2, ..., y_t), X_t^T = (x_1, x_2, ..., x_t), the estimation of time $t + 1$ is calculated as

$$\left.\begin{aligned} \hat{\theta_{t+1}} &= \hat{\theta}_t + K_{t+1}\left(y_{t+1} - x_{t+1}^T\hat{\theta}_t\right) \\ P_{k+1} &= \frac{P_t}{\lambda + x_{t+1}^T P_t x_{t+1}} \\ K_{t+1} &= P_{t+1}x_{t+1} \end{aligned}\right\} \tag{6}$$

where, λ is the forgetting factor which needs to be carefully set and it is a design issue. According to [21], for time-varying λ a good approach is to set it to a function of estimation error $\hat{e_t}$ as follows:

$$\lambda = 1 - \alpha_1\left(\frac{1}{\pi}\arctan\left(\alpha_2\left(|\hat{e_t}| - \alpha_3\right)\right) + \frac{1}{2}\right) \tag{7}$$

where, α_1, α_2 and α_3 are all design parameters.

$$\lambda_t = \begin{cases} 1 - \frac{\alpha_3}{\pi} arctan(|R_t - 1|), if\ |R_t - 1| \geq \alpha_2; \\ \alpha_1 + \frac{1}{\pi}(1 - \alpha_1)(arctan(1 - |R_t - 1|))else; \end{cases}$$

$$R_t = \begin{cases} max(\frac{\theta_{t-k}^{ij}}{\theta_t^{ij}}, \frac{\theta_t^{ij}}{\theta_{t-k}^{ij}}), if \theta_{t-k}^{ij}\theta_t^{ij} \neq 0 \\ \infty else. \end{cases} \tag{8}$$

$\forall i \in (1, 2, ..., n)$, $\forall j \in (1, 2, .., m)$, with $k, \alpha_1, \alpha_2, \alpha_3$ tunable parameters, $\frac{1}{3} \leq \alpha_1 < 1$, $\alpha_2 \geq 0$, $0 \leq \alpha_3 \leq 2$, $k \in N$

We use equation 8, as the authors have suggested a time varying forgetting factor in it, which enables more accurate tracking than constant forgetting factor given in equation 7. Here also, for our application, set Y is our observed data and set X is the cost parameter variable to be estimated.

The KFs and the EKFs are also useful approach for estimation and optimisation. The KF is often used when the parameter is linearly time-varying, where the equation of the system model is given as:

$$\begin{aligned} x_{k+1} &= A_k x_k + B_k u_k + G w_k \\ y_k &= C_k x_k + v_k \end{aligned} \tag{9}$$

where, x is the parameter to be estimated and y is the observation of x. Also, $x(k) \in \mathbf{R^n}$, $u(k) \in \mathbf{R^n}$, $w(k) \in \mathbf{R^n}$, $v(k) \in \mathbf{R^r}$ and $y(k) \in \mathbf{R^r}$. Moreover, $w(k)$ and $v(k)$ are white, zero mean, Gaussian noise. The KF results from the recursive application of the prediction and the filtering cycle as given by the following equations:

$$\begin{aligned} \hat{x}(k+1 \mid k) &= A_k\hat{x}(k \mid k) + B_k u_k \\ \hat{P}(k+1 \mid k) &= A_k P(k \mid k) A_k^T \end{aligned} \tag{10}$$

$$\left.\begin{aligned} \hat{x}(k \mid k) &= \hat{x}(k \mid k-1) + K(k)[y(k) - C_k\hat{x}_{k|k-1}] \\ K(k) &= P(k \mid k-1)C_k^T[C_k P(k \mid k-1)C_k^T + R]^{-1} \\ P(k \mid k) &= [I - K(k)C_k P(k \mid k-1)] \end{aligned}\right\} \tag{11}$$

where, $\hat{x}(k+1)$ is the new estimation for the variable $x(k)$. Equation 10 gives the prediction cycle of the KF dynamics and equation 11 gives the filtering cycle. Here, $K(k)$ is the KF gain. In our work, $C(t, e, b, f, o)$ is the parameter which is to be estimated. Thus, x in the general KF equation, is $C(t, e, b, f, o)$ in our application.

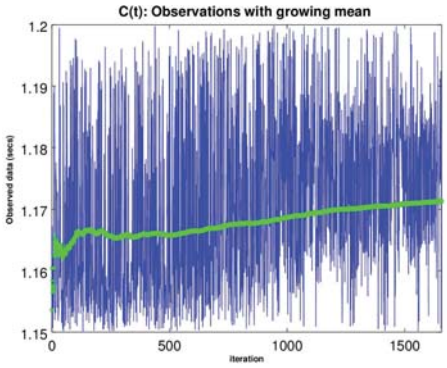


Fig. 7: The observed $C_0\,(t, e, f)$

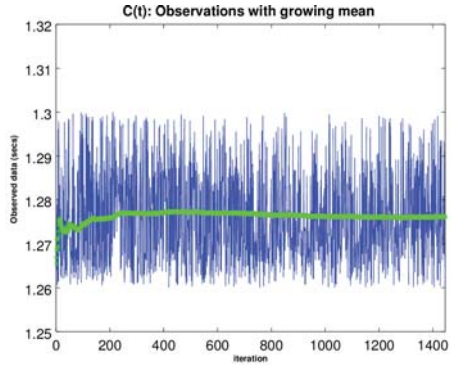


Fig. 8: Observation of $C_{79}\,(t, e, f)$

VII. APPLICATION OF THE ESTIMATION METHODS

A. Experimental setup

The prototype platform, described in section IV is the basic experimental set-up for our work to validate our proposal, where the AGVs perform specific tasks given to them, like traversing from one port to another as carriage, loading and unloading at designated ports, *et cetera*.

For measuring the cost variable of the system and to understand the current battery power on-line (as formulated in section V and explained in section VI), the platform is conceptualized consisting of three AGVs, each of them have the $L0.0$ and $L0.1$ levels implemented and all of them are individually controlled by their $L1$ level. For predicting the current battery level from the cost variable estimation, the effect of tire conditions and effect of the frictional force of the floor are considered. Thus, $C(t, e, b, f, o)$ (section V) is re-formulated as $C(t, e, f)$.

For a certain instance, as in Figure 6, let AGV_1 is required to go from n_7 to n_{12} and there are 2 arcs in that path. So there are 2 cost parameters for AGV_1 namely, $C_{79}\,(t, e, f)$ and $C_{912}\,(t, e, f)$. Similar is the case for both the other AGVs (AGV_2 and AGV_3).

For the time being, the estimation methods discussed in Section VI is applied to estimate one such cost variable to validate whether the standard estimation methods, generally used for position or mechanical parameter estimation, actually works for time variable estimations so that the battery efficiency at the run-time can be known. Let that cost variable be denoted as $C_0\,(t, e, f)$.

B. Experiment Conducted

The $C_0\,(t, e, f)$ is measured till the battery of the mobile robot drains out and the robot comes to complete halt.

The plot of the observed (measured) $C_0\,(t, e, f)$ over time is given in Figure 7. Several epochs of tests are done to measure the same $C_0\,(t, e, f)$ in order to ensure that we are measuring the right data. We can observe that the values of $C_0\,(t, e, f)$ is influenced by the battery discharge profile which is given in Figure 3, where the state of charge of the batteries used in the platform is inverse of this profile. This observation was formulated in Section III.

Hence, the battery's state of charge can be predicted from the observed data of $C_0\,(t, e, f)$ as the battery's state of charge influences the values of $C_0\,(t, e, f)$. We show observation of another cost variable ($C_{79}\,(t, e, f)$) in Figure 8, where the similar dependency of cost variables with battery charge is exhibited.

Thereafter, we apply the three different estimation methods explained in Section VI to both the cost variables plotted in Figure 7 and Figure 8 to obtain on-line predictions of these cost variables which denotes the battery power in real time during the task performances of AGVs. We deploy LSMW method [21] first because it is a naive, inexpensive approach to estimate variables on-line. We also deploy the RLS algorithm proposed in [21] with time-varying forgetting factor (equation 8) as it helps in accurate tracking. Thereafter, we deploy KF method to further enhance the accuracy.

C. Results of the estimation methods deployed

On all the plots, the blue thick line demonstrates the estimated values and the green dashed line demonstrates the observations. Also, the left plot is for the estimated results for $C_0\,(t, e, f)$ and the right plot is for the results on $C_{79}\,(t, e, f)$.

At first, when LSMW method is implemented to find the estimation, the window size was taken as 10.

In Figure 9, the results of estimates on two different cost variables, $C_0\,(t, e, f)$ and $C_{79}\,(t, e, f)$, obtained by LSMW method is demonstrated. We can observe that the mean of the estimation error is of the order of 10^{-3}.

The error level is not being further reduced by applying the Recursive Least Square (RLS) method, whose estimation results are shown in Figure 10, with the order of the error being 10^{-3}.

The results of estimation done by the Kalman filtering is given by the Figure 11. In Figure 11, the estimation error is of the order of 10^{-5}. Hence, we can infer that the Kalman Filtering method provides the most suitable estimation for the cost parameter to derive the knowledge of current battery capacity during operation of the robotic system.

All the computations done are enough short in time to be obtained in real time.

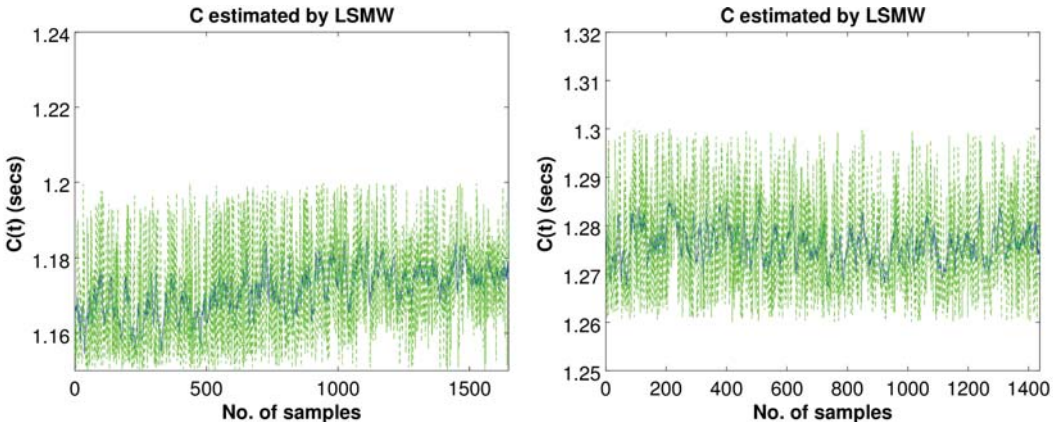


Fig. 9: LSMW estimation results $C_0\,(t, e, f)$ and $C_{79}\,(t, e, f)$

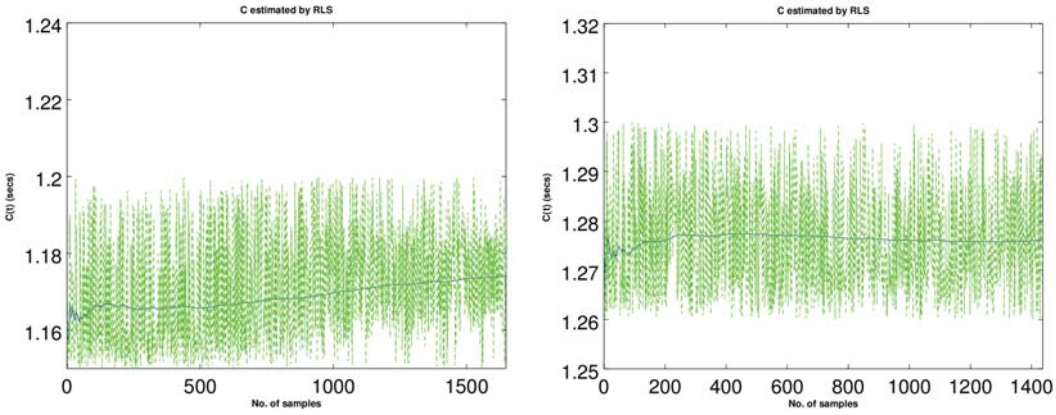


Fig. 10: RLS estimation results on $C_0\,(t, e, f)$ and $C_{79}\,(t, e, f)$

VIII. DISCUSSION OF THE RESULTS

We present the estimation results on two different cost parameters, namely, $C_0\,(t, e, f)$ and $C_{79}\,(t, e, f)$. From the predicted results of Kalman Filtering (Figure 11), RLS method (Figure 10) and LSMW method (Figure 9), we can observe that the predicted values of both $C_0\,(t, e, f)$ and $C_{79}\,(t, e, f)$ cost variables are in parity of the battery discharge profile of the ENELOOP batteries. In comparison, we can observe in Figure 11 that Kalman Filtering provides the best estimates for estimating the time-varying cost variables in order to predict the performance capability of individual AGVs.

Therefore, from the predicted values of the different cost variables like $(C_0\,(t, e, f))$ and so on, the real time state of charge of the batteries is obtained for a particular AGV. These predictions are obtained on-line during the run-time of the multi-robot prototype system. Henceforth, these on-line predicted values for the state of charge of the batteries can be efficiently used during the task allocation or deciding the path to traverse for any AGV in the system. Also, during the onset of a particular task or movement these predicted values for the charge level of the batteries can be used by the controller to decide whether to start the task or the movement.

Moreover as highlighted in Section III, when the prediction can be achieved about the battery efficiency of completing one task of the individual AGV, then dynamically better decision can be taken for the next assignment of task for each AGV. This will eventually lead to a cost effective controller for each AGV and the system as a whole.

IX. CONCLUSION AND FUTURE SCOPE

The prediction of on-line battery efficiency of individual AGV is done from the approach of estimating the cost incurred in performing and completing one task assigned to the individual AGVs in the multi-robot system implemented in transportation and automation industry. The most suitable estimation method for prediction is proposed after deploying three most usable estimation methods. Moreover, it is proposed that when the correct battery capability will be known, the high level decisions of each AGV will be more cost effective which will make the whole system performance better. It is evident from the results that Kalman Filtering provides the best method for estimating the proposed cost variables. The

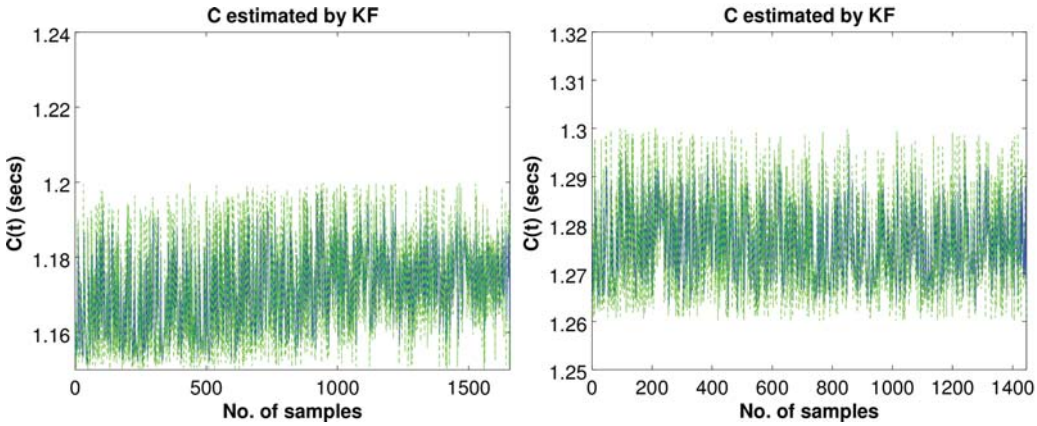


Fig. 11: KF estimation results on $C_0\left(t,e,f\right)$ and $C_{79}\left(t,e,f\right)$

control architecture of the prototype platform used in this work enables each AGV in a multi-robot system to share and distribute information among each other. Thus, the prediction values of battery capacity at the run-time provides a direction to make a more profit based control for all the AGVs as a whole for optimised assignments of tasks and movement for the mobile robots.

REFERENCES

[1] L. Ribas-Xirgo, J. M. Moreno-Villafranca, and I. F. Chaile, "On using automated guided vehicles instead of conveyors," in *Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on*. IEEE, 2013, pp. 1–4.

[2] D. Herrero-Perez and H. Martinez-Barbera, "Modeling distributed transportation systems composed of flexible automated guided vehicles in flexible manufacturing systems," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 2, pp. 166–180, 2010.

[3] L. Schulze and A. Wüllner, "The approach of automated guided vehicle systems," in *IEEE International Conference on Service Operations and Logistics, and Informatics 2006. SOLI'06*. IEEE, 2006, pp. 522–527.

[4] A. Malinowski and H. Yu, "Comparison of embedded system design for industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 2, pp. 244–254, 2011.

[5] W. Xing, L. Peihuang, C. Qixiang, Z. Chidong, S. Ke, and J. Chen, "Design and control of material transport system for automated guided vehicle," in *2012 UKACC International Conference on Control (CONTROL)*, Sept 2012, pp. 765–770.

[6] J. Schmidt, C. Meyer-Barlag, M. Eisel, L. M. Kolbe, and H.-J. Appelrath, "Using battery-electric {AGVs} in container terminals — assessing the potential and optimizing the economic viability," *Research in Transportation Business & Management*, vol. 17, pp. 99 – 111, 2015, energy Efficiency in Maritime Logistics Chains. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2210539515000516

[7] T. Kawakami, Takehito and Shozo, *Design for Innovative Value Towards a Sustainable Society: Proceedings of EcoDesign 2011: 7th International Symposium on Environmentally Conscious Design and Inverse Manufacturing*. Dordrecht: Springer Netherlands, 2012, ch. Battery Life Cycle Management for Automatic Guided Vehicle Systems, pp. 403–408.

[8] K. Vivaldini, L. F. Rocha, N. J. Martarelli, M. Becker, and A. P. Moreira, "Integrated tasks assignment and routing for the estimation of the optimal number of agvs," *The International Journal of Advanced Manufacturing Technology*, vol. 82, no. 1, pp. 719–736, 2015.

[9] M. M. Oliveira, J. P. M. Galdames, K. T. Vivaldini, D. V. Magalhães, and M. Becker, "Battery state estimation for applications in intelligent warehouses," in *2011 IEEE International Conference on Robotics and Automation (ICRA)*, May 2011, pp. 5511–5516.

[10] K. C. T. Vivaldini, M. M. Oliveira, J. P. Galdames, J. A. Santos, D. V. Magalhães, and M. Becker, "Battery charge state estimate for a robotic forklift routing system," in *2013 IEEE International Conference on Industrial Technology (ICIT)*, Feb 2013, pp. 1222–1227.

[11] N. Mathew, S. L. Smith, and S. L. Waslander, "A graph-based approach to multi-robot rendezvous for recharging in persistent tasks," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2013, pp. 3497–3502.

[12] H. Liu, N. Stoll, S. Junginger, and K. Thurow, "A new approach to battery power tracking and predicting for mobile robot transportation using wavelet decomposition and anfis networks," in *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2014, pp. 253–258.

[13] A. Rodic, M. Jovanovic, S. Popic, and G. Mester, "Scalable experimental platform for research, development and testing of networked robotic systems in informationally structured environments experimental testbed station for wireless robot-sensor networks," in *2011 IEEE Workshop on Robotic Intelligence In Informationally Structured Space (RiiSS)*. IEEE, 2011, pp. 136–143.

[14] A. Lindsay, "Robotics with the boe-bot," in *Parallax Inc. United States*, 2004.

[15] L. Ribas-Xirgo and I. F. Chaile, "Multi-agent-based controller architecture for agv systems," in *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*. IEEE, 2013, pp. 1–4.

[16] A. Norouzi and C. A. Acosta, "An approach to design a robust software architecture and an intelligent model for multi-agent systems," in *AI & Robotics and 5th RoboCup Iran Open International Symposium (RIOS), 2013 3rd Joint Conference of*. IEEE, 2013, pp. 1–7.

[17] W. Rackl, R. Lampariello, and G. Hirzinger, "Robot excitation trajectories for dynamic parameter estimation using optimized b-splines," in *2012 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2012, pp. 2042–2047.

[18] G. Erinc, G. Pillonetto, and S. Carpin, "Online estimation of variance parameters: experimental results with applications to localization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS*. IEEE, 2008, pp. 1890–1895.

[19] M. M. Olsen and H. G. Petersen, "A new method for estimating parameters of a dynamic robot model," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 1, pp. 95–100, 2001.

[20] F. Flacco, A. De Luca, I. Sardellitti, and N. G. Tsagarakis, "On-line estimation of variable stiffness in flexible robot joints," *The International Journal of Robotics Research*, vol. 31, no. 13, pp. 1556–1577, 2012.

[21] Z. Wang, A. Peer, and M. Buss, "Fast online impedance estimation for robot control," in *IEEE International Conference on Mechatronics, 2009. ICM*. IEEE, 2009, pp. 1–6.

[22] V. A. Sujan and S. Dubowsky, "An optimal information method for mobile manipulator dynamic parameter identification," *Mechatronics, IEEE/ASME Transactions on*, vol. 8, no. 2, pp. 215–225, 2003.

A Proposal for the Design of a Semantic Social Path Planner using CORTEX

Pedro Núñez, Luis J. Manso, Pablo Bustos, Paulo Drews-Jr, Douglas G. Macharet

***Abstract*—Path planning is one of the basic and widely studied problems in robotics navigation, being its aim to determine the path from one coordinate location to another along a set of way-points. Traditionally, this problem has been addressed using the geometric world, that is, 2D or 3D coordinates from a geometric map. New generation of robots should be able to plan this path also taking into account social conventions, which is commonly called social navigation. This paper describes the ongoing work of a new proposal for the path planning problem where the semantic knowledge of the robot surrounding and different social rules are used to determine the best route from the robot to the target poses. In this work, a specific type of semantic map is described, which integrates both, geometrical information and semantic knowledge. The proposal uses CORTEX, an agent-based Robotics Cognitive Architecture which provides a set of different agents in the deliberative-reactive spectrum. The proposal is going to be tested in two different social robots within the NAVLOC project[1].**

***Index Terms*—Path planning, social navigation, human-robot interaction**

I. Introduction

SOCIAL robotics is witnessing a remarkable growth in the recent years. In a not very far future, social robots will perform everyday tasks in offices, hospitals homes or museums. These actual scenarios are usually complex and dynamic, that is, people, objects or other robots move around the robot and, therefore, the environment in an instant of time is not the same after some days, hours or even some minutes. In the social context where these robots are going to work, there exist different capabilities and skills that are expected, such as human or object avoiding collisions, localization, path planning, human-robot interaction or object recognition.

Most of the solutions proposed in the literature for these typical problems, especially for navigation tasks, are addressed using representations of the spatial structure of the environment. These solutions provide a robot with more or less efficient methods to carry out some tasks in static and simple scenarios. However, using only a spatial representation of the environment is difficult to navigate successfully. This tendency is now changing, and the scientific community is experiencing an increasing interest in so-called semantic solutions, which integrate semantic knowledge and geometrical information.

Pedro Núñez, Luis J. Manso and Pablo Bustos are with University of Extremadura.
E-mail: pnuntru@unex.es
Paulo Drews-Jr is with Federal University of Rio Grande.
Douglas G. Macharet is with Federal University of Minas Gerais

[1]Brazil-Spain cooperation on navigation and localization for autonomous robots on underwater and terrestrial environments

Recently, the term *social navigation* in robotics has been introduced as a way to relate human-robot interaction and the robot navigation in human scenarios. The main goal of social navigation is to develop methods in order to make robot behavior, in particular robot navigation, socially accepted [1]. In the case of the socially-aware path planning problem, which is expected to become an increasingly important task in next social robots generation [2], the robot should decide the best route to the target poses following social rules ($e.g.$, to gracefully approach people, or to wittily enter and exit from a conversation).

To determine the best social route from the robot to the target poses, this work proposes a new design of a path planning algorithm based on semantic knowledge of the environment robot and the use of socially accepted rules. Classical path planning approaches assume geometrical information from the environment, that is, they use spatial representation of the environment ($e.g.$, topological, feature-based or occupancy grid maps), and reason about the best route using this spatial memory. On the contrary, the semantic social path planning approach described in this paper firstly introduces a high-level and long-life knowledge captured by the robot from the environment, in a similar way that the human point-of-view, and after, it introduces socially accepted rules to the semantic knowledge during the planning task. The main advantages of the semantic approach are robustness, ease of human-robot communication and a more understandable access to robot's inner functioning for debugging. It can be more robust to localization errors than classical path planning algorithms based on low-level geometric descriptors, since it adds as a new source of information the position of known objects along the path. Moreover, using semantic way-points, robots and humans can share routes since their way-points would be described using objects and labels instead of two-dimensional coordinates given in an arbitrary reference frame.

New generations of social robots should be able to generate different routes during an interaction with humans and also exhibit proactive social behaviors. In this context, the cognitive architecture CORTEX used in this paper and defined in [3], is based on a collection of agents ($i.e.$, semi-autonomous functional units that collaborate by means of a common representation in their pursue of a common goal) that can operate anywhere in the deliberative-reactive spectrum. In this architecture there exist navigation, perceptual and human-robot interaction agents, among others, thereby facilitating the combined use of semantic knowledge and social rules.

The use of semantic knowledge, thus, may enable a social robot to work in a more intelligent and robust manner [5].

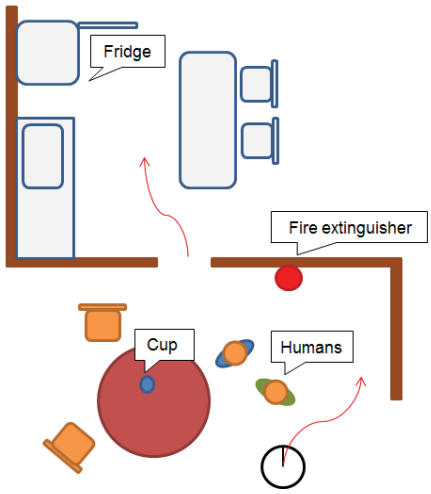


Fig. 1. Brief description of the semantic social path-planning problem. The robot has to choose the best route from the living-room to the kitchen by using its semantic knowledge (*e.g.*, cup, fire extinguisher and fridge) and social rules (*e.g.*, humans along the path)

Fig. 1 illustrates the problem to solve: the robot located in the living-room has to choose the best route from its pose to the target along a complex dynamic environment with people. In this study case, the robot's target is 'kitchen'. The robot has knowledge about different objects with label 'kitchen' (*e.g.*, fridge). Therefore, the best route has to be chosen from the 'living-room' to the 'kitchen'. Here, it uses perceptual agents in order to perceive the environment and detect objects from it (semantic knowledge), and also uses human-robot interaction agents to infer or apply social rules. Obviously, there exist navigation agents that allow the robot to navigate in a secure way.

What does a social robot need for a social semantic path planning algorithm? The next list summarizes the main items:

- A navigation system that smoothly integrates local and global constraints.
- Semantic description and recognition of objects and humans in the robot's surroundings.
- Automatic Speech Recognition (ASR) and Text-To-Speech (TTS) algorithms (Human-Robot Interaction), and optionally the capability to enhance dialogs with accompanying movements of the head and arms to express emotions.

This article is structured as follows: Section II provides a brief summary about similar works in this field of research. In Section III, a description of CORTEX cognitive architecture and the hybrid representation are made. Section IV provides the proposal description along the main agents involved. The description of the semantic social path planning is described in Section V. Finally, the main conclusions are detailed in Section VII.

II. Related Work

How autonomous robots move in human environments has a strongly effect on the perceived intelligence of the robotic system [6]. Independently of the physical capabilities of robots (navigation of wheeled robots are completely different to biped ones), social navigation started being extensively studied in the last years and several methods have been proposed from then. On one hand, some authors propose models of social conventions and rules by using cost functions [7], [8]. A typical solution is to propose a classic navigation method, adding social conventions and/or social constraints. In [7], for instance, the authors use a classical A* path planner in conjunction with social conventions, like to pass a person on the right. Other work as [8] uses potential fields and a proxemics model[2]. On the other hand, several authors use human intentions in order to model the social navigation [10]. In [10], authors propose a local navigation method and a human intention analysis using face orientation in order to modify the trajectory, which they called as Modified Social Force Model (MSFM) with three types of human intentions: avoid, unavoid and approach.

All the aforementioned methods need global path planners in order to choose the best route from the robot to the target and then, they apply social conventions and constraints to modify this path. Classical global path planners use a spatial representation of the robot's surrounding, that is, the path-planning problem requires a map of the environment. Numerous path-planning algorithm have been proposed in the literature, from classical Dijkstra or A* algorithms to other more complex systems. An interesting review of path planning algorithms was writing by LaValle et al. [11], who also propose the Rapidly-exploring Random Trees (RRT) method. This method, in conjunction with the Probabilistic Road Map algorithm (PRM) algorithm [12] is used in the cognitive architecture CORTEX [3].

Recently, several advances in semantic planning have been achieved. In fact, social robots that incorporate capabilities for task planning and storing some semantic knowledge in their maps are commonly used (*e.g.*, classification of spaces, such as rooms, corridors or garden, and labels of places and/or objects) [15], [5]. By using this semantic knowledge, robots are able to navigate or planning tasks. In several works is used voice interactions with robots in order to build the semantic map. This same problem is autonomously acquired by the robot in most recent works (see Kostavelis's survey [15]). In this paper, a 3D object detection and representation autonomous agent is used.

Finally, one of the main problem to solve is the cognitive architecture the robot is equipped with and also the kind of semantic and spatial information the robot has to store. Different proposals can be found in the literature, most of them by separately storing symbolic and metric information. Symbolic knowledge representation, such as the works proposed in [16] and [17], have been at the core of Artificial Intelligence since its beginnings. Recently, works that integrate

[2]Proxemics is defined as the study of humankinds perception and use of space [9]

spatial and symbolic knowledge in a unified representation are more frequent in the literature, and have a distinctive start in the *deep representation* concept introduced in [18]. Examples of these deep representations are [18], [19], [3]. In the proposal described in this paper, the robot uses the CORTEX architecture, which has demonstrated to be a robust and efficient agent-based architecture for robotics applications and is based on a set of agents interacting through a deep state representation [3].

III. Deep State Representation and Cortex

The concept of *deep representations* was initially described by Beetz et al. [18]: *representations that combine various levels of abstraction, ranging, for example, from the continuous limb motions required to perform an activity to atomic high-level actions, subactivities, and activities*. Deep representation advocates the integrated representation of robots knowledge at various levels of abstraction in a unique, articulated structure such as a graph. Based on this definition, in [3] it is proposed a new shared representation to hold the robots belief as a combination of symbolic and geometric information. This structure represents knowledge about the robot itself and the world around it. From an engineering point of view this representation is flexible and scalable. Formally, a deep state representation was defined as *a directed multi-labelled graph where nodes represent symbolic or geometric entities and edges represent symbolic and geometric relationships*.

The robotics cognitive architecture CORTEX is defined structurally as a configuration of software agents connected through a deep state representation that is called DSR, where an agent is defined in [3] as *a computational entity in charge of a well defined functionality, whether it be reactive, deliberative of hybrid, that interacts with other agents inside a well-defined framework, to enact a larger system*. The CORTEX architecture is implemented on top of the component-oriented robotics framework RoboComp [14]. In CORTEX, higher-level agents define the classic functionalities of cognitive robotics architectures, such as navigation, manipulation, person perception, object perception, dialoguing, reasoning, planning, symbolic learning or executing. The choice of these functionalities is by no means a closed issue in Cognitive Robotics but it is clearly a discussion outside the scope of this paper. These agents operate in a goal-oriented regime [16] and their goals can come from outside through the agent interface, and can also be part of the agent normal operation.

In Fig. 2 an overview of the DSR and its location within the cognitive architecture CORTEX is drawn. Different agents, such as navigation, person detector or planner are also shown. DSR is illustrated as a a graph where all the robot knowledge about its surrounding is represented. The next sections describe the design of the semantic social path planner using the cognitive architecture CORTEX, analyzing the agents involved and the relationships between them in three different cases of study.

IV. Agents

In order to plan the best social route from the robot pose to the target, different specific agents within CORTEX

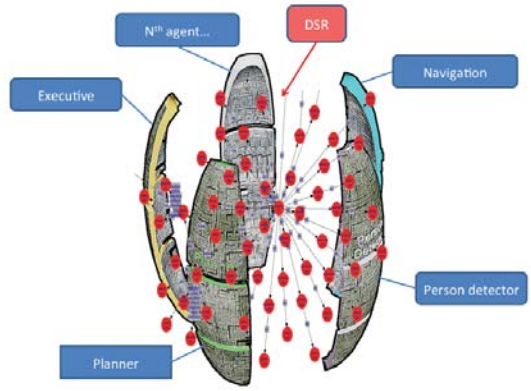


Fig. 2. An overview of the DSR and its location within the cognitive architecture CORTEX [3]

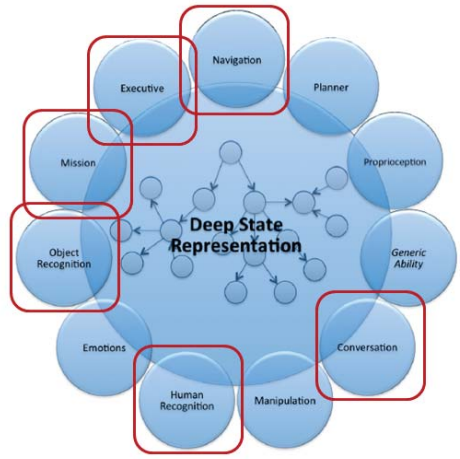


Fig. 3. Main agents within CORTEX involved in the semantic social path planning described in this proposal are highlighted in red.

are involved. First, the robot must have the capability of detecting objects in the path and updating the symbolic model accordingly. Additionally, the skill of detecting humans is also mandatory because robots need to know about humans to get commands, avoid collisions and provide feedback. The final, and most important agent for social navigation, is the one implementing the navigation algorithms that allow robots to navigate from a point to another in a secure and social manner. In the next subsections, a brief description of the main agents involved in the proposal is provided. These agents are highlighted in Fig. 3, which illustrates the current CORTEX cognitive architecture [3].

A. Object detection and representation

The object perception agent is in charge of recognizing and estimating the pose of objects and visual marks in the environment. For each object or mark detected it describes within the model (DSR) not only the pose but also its type.

These kind of elements are useful for the robot in several scenarios. First of all, the targets will usually be objects and qualitative positions (*e.g.*, in front of something, close to something, between two objects) taking the objects as reference instead of just some coordinates. Humans will seldom ask the robot to go to a coordinate because they do not necessarily need to know the reference frame used by the robots and, more importantly, because it is not comfortable for humans to provide targets in such a way.

Synthetic visual marks are detected using the AprilTags library [20]. Arbitrary visual marks will be detected using the OpenCV library [21] and 3D objects are currently being detected using an object recognition pipeline based on the PointClouds library [22]. The poses of the detected objects are referenced to known objects (in the DSR) that support them, such as a known table under a target cup. Once referenced, the kinematic relations embedded in DSR allow the computation of any object's pose from any reference frame easily.

B. Person detector

Person detector is the agent responsible for detecting and tracking the people in front of the robot. Humans do not usually enjoy its personal space being invaded by robots. The presence of humans in the robots' path or in their environment may determine changes in the navigation route in order to make it socially acceptable. In social navigation, a human can also interact with the robot and give it orders, or the robots, in their way to their target, might be able to communicate with people to provide information or ask for help.

The person detector agent acquires the information using an RGBD sensor. For each detected person the agent inserts in the DSR the pose of its torso, its upper limbs, and the head. The lower limbs are ignored because they do not provide as much social information as the head, the upper limbs and the torso do [3]. These elements can be used to infer the objects referenced by the humans when they point or look at them. The torso is used to avoid entering the personal space of humans and as an indicator of the possible directions in which they might walk.

C. Conversation

The conversation agent performs human-robot interaction (HRI). In social environments, HRI provides tools to the robot and/or human to communicate and collaborate. Therefore, this agent is used to include information in the model when humans tell robots about unknown objects and to properly acquire commands. Automatic Speech Recognition and Text-to-Speech algorithms allow robot to both send and receive information to/from humans in the environment during its social navigation.

D. Mission

This agent is used as a means to provide missions to the executive agent and to visualize the DSR. It has two graphic views. A graph-like view and a 3D geometric view [3].

E. Executive

The Executive is responsible of planning feasible plans to achieve the current mission, managing the changes made to the DSR by the agents as a result of their interaction with the world, and monitoring the execution of the plan. The active agents collaborate executing the actions in the plan steps as long as they consider them valid (it must be taken into account that agents might have a reactive part). Each time a structural change is included in the model, the Executive uses the domain knowledge, the current model, the target and the previous plan to update the current plan accordingly. The Executive might use different planners. Currently AGGL [4] and PDDL-based [23] planners are supported.

F. Navigation

Navigation is in charge of performing local navigation complying with social rules and including the location of the robot in the DSR. Global path planning is performed by the symbolic planner used by the executive.

Two poses are maintained by the robot: the pose obtained from the local odometry, and the pose provided by a localization algorithm based on external geometric laser features. Given their properties, each of these poses is useful for a particular purpose. Odometry provides good information relative to the robot's position in the short term, while localization provides good information for mid and long term positioning. Additionally, the space walked by the robot in the last seconds is also included.

Regarding localization algorithms, the navigation agent is algorithm-independent. It has been used with different algorithms showing different properties, which can be selected to fit different kinds of environments.

While it can be used with any local navigation system, the navigation agent has been only tested with the path-planning algorithm proposed in [24], an algorithm based on the elastic-band representation, with successful results. The proposal presented in this paper extends the geometrical path-planning to a social semantic algorithm, which is described in the next section.

V. Social Semantic Path planning in Cortex cognitive architecture

In this section the semantic social path planning algorithm is described. An overview of the system is shown in Fig. 4. As illustrated in the figure, the complete system is composed of a global semantic path planner followed by a local geometrical path planner. Both of them are affected by a social navigation model. The semantic path planner chooses the optimal route, that consists of a list of labeled objects in the map. Then, the robot plans a local geometrical navigation from its current pose to the next object in the list. This path is affected by the

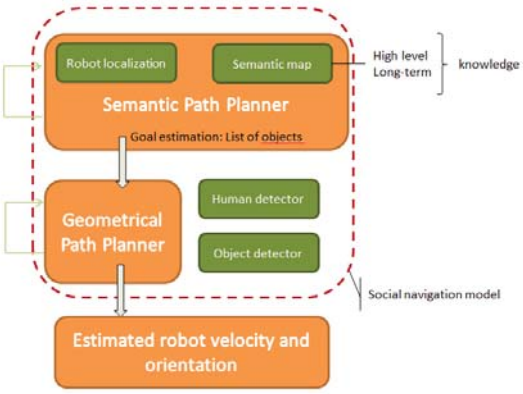


Fig. 4. The overall blocking diagram of the proposed system.

social navigation model, and if necessary, the local (or global) route is re-planned. All agents described in previous section are concurrently running in the navigation process.

A. Semantic Path Planning

Global path planning at a symbolic level is performed by the planner included in the executive. The semantic path planner is based on the use of a two-hierarchies architecture, similar to that one presented in [5]. Both, the spatial and semantic properties of each object within DSR allow the planner to choose the best global route. Let $O = \{o_1, o_2, ..., o_n\}$ being the list of n objects o_i within the semantic map of the robot, that is, its high level and long-term knowledge. Each object o_i is characterized by a duple $o_i = \{m_i, s_i\}$, where m_i is the metric representation of the object ($i.e.$, rotation and translation matrices from its parent node) and s_i is the semantic information associated to the object ($i.e.$, label). Each object has a parent node, which usually represents the room where the object is located. Rooms are also nodes of the graph, that are connected if they are sharing a door. Thus, the semantic path planning algorithm chooses the best route from the graph, that is, the list of rooms that the robot has to visit, and then, generates a list of j objects (waypoints) for the local planner, $\Gamma_R = \{o'_1, o'_2, ...o'_j\}$, being $o'_j \in O$. Thus, global navigation is achieved by using consecutive objects from Γ_R.

B. Geometrical path Planning

Once the robot is assigned the path Γ_R, the geometrical path-planner should accomplish the navigation between two consecutive objects, o'_{k-1} and o'_k. The geometrical path-planning algorithm of the proposal is based on the use of graphs as a representation of free space and of elastic-bands [24] as an adaptable representation of the current path. Elastic bands work as a glue filling the gap between the internal representation of the path and the constraints imposed by the world physics. In order to build a graph representing the overall free space, the probabilistic road map algorithm, PRM, is used [12] along with a preexisting map and a collision detection algorithm. To complete this schema, the RRT algorithm [11] is also included in the system to complete the paths when unconnected islands remain in the PRM graph or to connect the robot's current position and robot's final position with nodes in the graph. The object perception agent is directly involved in the path following process: when the robot detects the object o'_k, a new target is generated, being the new local route defined by the nodes o'_k and o'_{k+1}.

C. Social Navigation Model

In order to mathematically formulate the socially-acceptable path planning algorithm, let denote $H = \{H_1, H_2, ...H_n\}$ the set composed by n humans in the environment. Each human, H_i, in the DSR is represented by the pose of its torso, its upper limbs, and the head. These elements can be used for defining a personal space θ_i and a social interaction intention ρ_i. Both θ_i and ρ_i are detected by the human detector agent, and are included in the DSR as information associated to human H_i. On one hand, and similar to the work presented in [2], θ_i is defined as Gaussian Mixture Model of two 3D Gaussian functions, one for the front of the individual, and other for its rear part. By adjusting the covariance matrices of these two gaussians, one can modify the personal space model. On the other hand, ρ_i describes the different cases where a human wants or not to interact with the robot during the path: i) human does not want to interact ($i.e.$, human is considered as obstacle); ii) human wants to interact with the robot, and then, the robot has to approach human, interact and finish the communication. In this respect, depending of the ρ_i value, the final path may be modified. For instance, if the human is considered as obstacle, the graph in the geometrical local navigator has to be updated in order to avoid this new obstacle (see Fig. 5(a)). On the contrary, if the human wants to interact with the robot, a new object $o'_k = H_i$ is included in the list of nodes to reach, being H_i the next target (see Fig. 5(b)).

VI. Cases of study

Within the 'BS-NAVLOC' project, this paper proposes three different cases of study. All of them are examples of robots navigating in indoor environments, and the main goal is to demonstrate that the semantic social path planning algorithm proposed in this paper, using the CORTEX cognitive architecture, may be performed in different robotics platforms in a near future and with successful results. In this section, the ongoing work is presented, describing briefly the DSR and the relationships between the involved agents.

The proposal of semantic social path-planning algorithm is going to be tested in two different robots (Fig. 6). The first robot is the DOKBot, from VerLab at the University Federal of Minas Gerais, which consists of a Pioneer 2-AT robotics platform equipped with different perceptual sensors, such as laser, RGB cameras and microphones (see Fig. 6(a)). This robot was originally designed for semiautomatic telepresence purposes. The second autonomous system is Shelly, an anthropometric social robot that is currently being used in RoboLab,

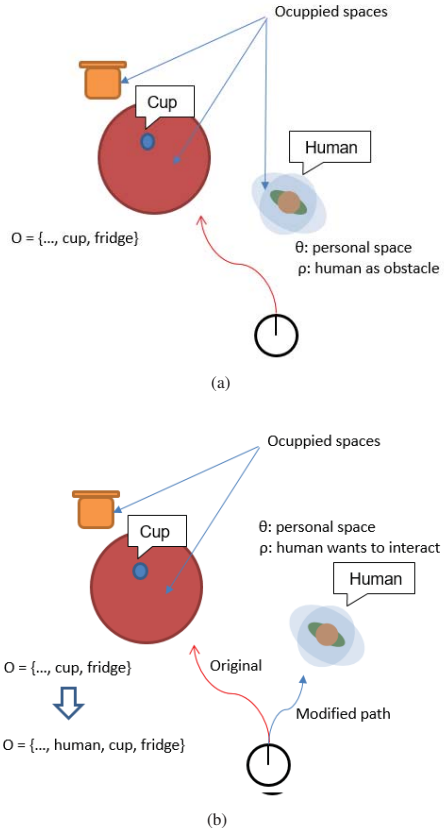
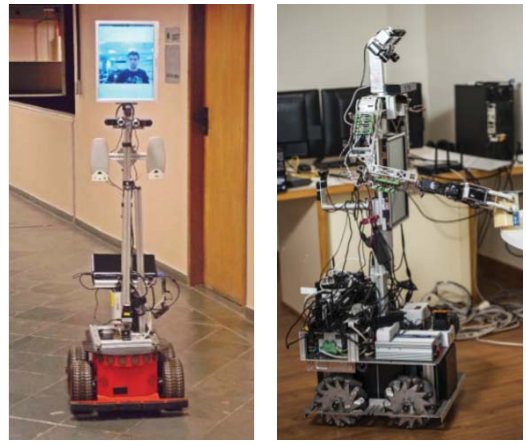


Fig. 5. Social navigation model proposed in this paper: a) human does not want to interact with the robot; and b) the human wants to interact.

at the University of Extremadura. This robot was designed to help in daily life tasks. It is composed of an omnidirectional base, two 7-DOF arms with two-fingered grippers and a RGB-D camera attached to a pan-tilt-yaw structure. It has another RGB-D camera on the upper part of the torso which is used to detect human bodies and a lidar for navigation. This robot is illustrated in Fig. 6(b).

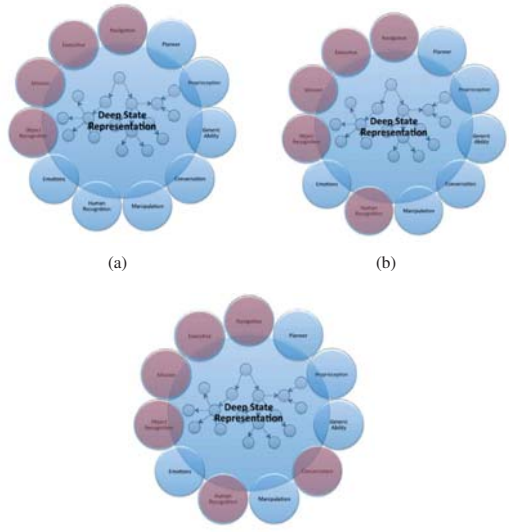
Next, the experimental scenarios are described. They have been designed from low-complexity to high-complexity levels:

- Semantic Path Planning. In this experiment, the robot chooses the best route from a room to another. In this scenario, there is not people in the path, and thus, only the semantic knowledge of the rooms is used. Fig. 7(a) shows the agents involved in this scenario.
- Semantic social path planning. This experimental scenario consists on a semantic social navigation. Similar to the previous case of study, the robot has to navigate between two different rooms in an indoor and human environment. In this respect, people walk or stand in the robot path, and thus, the robot has to modify the route in order to be socially accepted. The set of agents involved in this case of study is illustrated in Fig. 7(b).
- Semantic social path planning with HRI In this case of study, the robot first interacts with the human in order to know what is the next room to visit, and also, other humans interact with the autonomous agent during the path. In this HRI, the robot may modify partial or fully its route. Finally, in Fig. 7(c), the agents involved in CORTEX are highlighted.



(a) (b)

Fig. 6. The semantic social path-planner within CORTEX is going to be integrated in two different robots: a) DOKBot robot, from VerLab research group at the University Federal of Minas Gerais; b) Shelly robot, from RoboLab research group at the University of Extremadura.



(a) (b) (c)

Fig. 7. Agents involved in the cases of study described in this paper. a) semantic path planning; b) semantic social path planning; and c) semantic social path planning with HRI.

VII. Conclusions and future works

This paper presents the ongoing work, within the NAVLOC project, of a proposal for the design of a semantic social path-planning algorithm. The approach is based on the use of a global semantic path-planner in conjunction with a social navigation model. The theoretical proposal achieves the main goal of this kind of algorithm, that is, the robot is able to choose the best route from its current position to another position in a dynamic and complex scenario by using its high level knowledge and by applying social rules in order to be socially accepted. High functionality and robustness are guaranteed by using the cognitive architecture CORTEX and the Deep State Representation.

As it was aforementioned, this paper describes the ongoing work, where three different experimental scenarios are also described in order to test the proposed social navigation algorithm in future works. Currently, both spanish and brasilian researching teams, are working in integrating CORTEX in the two robots presented in this paper, Shelly and DOKbot.

Acknowledgments

This work has been partially supported by the MICINN Project TIN2015-65686-C5-5-R, by the Extremaduran Goverment project GR15120, by MEC project PHBP14/00083 and by CAPES-DGPU 7523/14-9.

References

[1] Lichtenthaler, C., Peters A., Griffiths, S. and Kirsch, A. Social Navigation - Identifying Robot Navigation Patterns in a Path Crossing Scenario, in Lecture Notes in Computer Science, Volume 8239, pp 84-93

[2] Gómez, J., Mavridis N. and Garrido, S. Social Path Planning: Generic Human-Robot Interaction Framework for Robotic Navigation Tasks, in Workshop on Cognitive Robotics Systems: Replicating Human Actions and Activities at IROS 2013.

[3] Luis Vicente Calderita, Deep State Representation: a Unified Internal Representation for the Robotics Cognitive Architecture CORTEX, PhD Thesis, University of Extremadura, 2015.

[4] L.J. Manso, P. Bustos, P. Bachiller and P. Núñez. "A Perception-aware Architecture for Autonomous Robots" In International Journal of Advanced Robotic Systems (ISSN 1729-8806), InTech, Vol. 12, No. 174, 2015. DOI: 10.5772/61742.

[5] Galido, C., Fernández-Madrigal, J.A., González J., and Saffiotti, A. Robot Task Planning using Semantic Maps in Robotics and Autonomous Systems, 56(11), pp. 955-966, 2008.

[6] Althaus, P.,Ishiguro, H., Kanda, T., Miyashita, T., Christensen, H.I.: Navigation for HumanRobot Interaction Tasks, in IEEE International Conference on Robotics and Automation, pp. 1894, 1989, 2004

[7] Kirby, R., Simmons, R., Forlizzi, J.: COMPANION: A Constraint-Optimizing Method for Person-Acceptable Navigation, in IEEE International Symposium on Robot and Human Interactive Communication, 2009

[8] Tranberg Hansen, S., Svenstrup, M., Andersen, H.J., Bak, T.: Adaptive Human Aware Navigation Based on Motion Pattern Analysis, in IEEE International Symposium on Robot and Human Interactive Communication, 2009

[9] Hall, E. "Proxemics", Current Anthropology, vol. 9, no. 2-3, pp. 83108, 1968.

[10] Photchara R., Mae, Y, Ohara K., Kojima M., and Arai T. Social navigation model based on human intention analysis using face orientation, in 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1682-1688, 201.

[11] LaValle, S.: Planning Algorithms. Cambridge University Press, Cambridge (2006).

[12] Kavraki, L.E., Svestka, P., Latombe, J.C., Overmars, M.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation 12(4), 566580 (1996).

[13] J. Fernandez-Madrigal, J. Gonzalez, Multi-Hierarchical Representation of Large-Scale Space, Int. Series on Microprocessor-based and Intell. Systems Eng., vol 24, Kluwer Academic Publishers, Netherlands, 2001

[14] L. Manso, P. Bachiller, P. Bustos, P. Núnez, R. Cintas and L.Calderita. RoboComp: a Tool based Robotics Framework, In Proceedings, SIMPAR Second International Conference on Simulation, Modeling and Programming for Autonomous Robots. pp 251-262. 2010.

[15] Kostavelis, I. and Gasteratos, A. Semantic mapping for mobile robotics tasks: A survey in Robotics and Autonomous Systems 66 (2015) pp. 86-103, 2014.

[16] Rusell, S., Norvig, P.: Articial Intelligence: A Modern Approach (3rd Edition). Pearson (2009)

[17] Poole, D., Mackworth, A.: Articial Intelligence: Foundations of Computational Agents. Cambridge University Press (2010), http://artint.info/index.html

[18] Michael Beetz, Dominik Jain, Lorenz Mösenlechner and Moritz Tenorth. "Towards performing everyday manipulation activities". In Journal of Robotics and Autonomous Systems, vol. 58, n. 9, pp. 1085–1095. Elsevier. 2010.

[19] Ali, M.: Contribution to decisional human-robot interaction: towards collaborative robot companions, PhD Thesis, Institut National de Sciences Appliquees de Toulouse, France (2012).

[20] Olson, E. AprilTag: A robust and flexible visual fiducial system. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pages 3400-3407.

[21] http://opencv.org/

[22] http://pointclouds.org/

[23] McDermott, Drew and Ghallab, Malik and Howe, Adele and Knoblock, Craig and Ram, Ashwin and Veloso, Manuela and Weld, Daniel and Wilkins, David. "PDDL-the planning domain definition language". 1998.

[24] Haut, M., Manso, L.J., Gallego, D., Paoletti, M., Bustos, P., Bandera, A. and Romero-Garces, A. A Navigation Agent for Mobile Manipulators, in Robot 2015: Second Iberian Robotics Conference Volume 418 of the series Advances in Intelligent Systems and Computing pp 745-756.

Building 3D maps with tag information

Angel Rodríguez, Francisco Gómez-Donoso, Jesus Martínez-Gómez and Miguel Cazorla

***Abstract*—Finding an appropriate environment representation is a crucial problem in robotics. 3D data has been recently used thanks to the advent of low cost RGB-D cameras. We propose a new way to represent a 3D map based on the information provided by an expert. Namely, the expert is the output of a Convolutional Neural Network trained with deep learning techniques. Relying on such information, we propose the generation of 3D maps using individual semantic labels, which are associated with environment objects or semantic labels. So, for each label we are provided with a partial 3D map whose data belong to the 3D perceptions, namely point clouds, which have an associated probability above a given threshold. The final map is obtained my registering and merging all these partial maps. The use of semantic labels provide us a with way to build the map while recognizing objects.**

***Index Terms*—semantic mapping, 3D point cloud, deep learning**

I. Introduction

THE use of appropriate environment representations is needed for most of the current robotic systems. Traditionally, environment representations have been limited to metrical maps that evolved from 2D to 3D with the release of affordable range sensors. In addition to this metric representation, semantic labels can be also used to represent rooms or scene categories. However, the location of relevant elements of the environment should be explicitly provided, which involves human supervision and reduces the adaptability of the system to environment modifications.

In this work, we propose to exploit the representation capabilities provided by available pre-trained deep learning models to automatically label indoor environments. We did not train our own CNN. Instead of that, we adopted the architecture defined by GoogleNet [22] as well as the pre-trained model they provide. This model was trained using the dataset ImageNet 2014 and it obtains a top-5 accuracy of 88.9%. It has to be emphasized that we chose this model over the rest because of its high accuracy rate and the fact that it provides object recognizing features for over 1,000 different classes, which will make our system adaptable and very context independent.

Our approach relies on the use of RGB-D sensors, namely a Microsoft Kinect or Asus Xtion device, suitable for performing a 3D registration of the environment. Over this metric representation, we automatically integrate semantic labels that allow us to determine the most probable location of objects. This process successfully combines 2D semantic labeling with 3D registration in an automatic fashion. An overall scheme of the proposal can be seen in Fig. 1.

The rest of the paper is organized as follows. In Section II we review some related works and state-of-the-art solutions to the semantic mapping problem and the deep learning techniques. The process for annotating 3D maps based on semantic labels is presented in Section III. Experimental results and the applications of the proposals are discussed in Section IV. Finally, the main conclusions of this work as well as some future research directions are outlined in Section V.

Angel Rodríguez, Francisco Gómez-Donoso and Miguel Cazorla is with University of Alicante.
E-mail: angelillo1992@gmail.com, fgomez@dccia.ua.es, miguel.cazorla@ua.es
Jesus Martínez-Gómez is with University of Castilla La Mancha. E-mail: jesus.martinez@uclm.es

II. Related Work

Building an appropriate representation of the environment in which an autonomous robot operates is still a widely addressed problem in the robotics research community. This problem is usually known as map building or mapping since maps are considered the most common and appropriate environment representation [23]. A map is useful for robot localization, navigation [5] and path-planning tasks [3], but also for a better understanding of the robot's surroundings [17]. That is, a map may not be limited to metric (e.g. specific poses of objects/obstacles) and topological information (e.g. paths from one place to others), but it can also integrate semantic information (e.g. symbolic representations of objects, expected behaviors for specific locations, or even situated dialogues, to name a few) corresponding to the objects, agents, and places represented on it. In this paper, we propose the use of deep learning techniques to provide semantic information. That information is fused with 3D data in order to obtain a novel map representation, object-oriented.

While the fusion of 3D data and visual information for generating environment representations is not new [8], our proposal presents an imporal novelty regarding regarding the use of ground truth information. Namely, we rely on expert systems trained from global image annotations instead of pixel level labelling. This increases the range of feasible applications as datasets annotatied at pixel level are not easy to generate. Furthermore, the number of classes or categories in the available datasets, such as NYU Depth [20] is notoriously smaller than those with global annotations like ImageNet.

Deep learning architectures have recently revolutionized 2D object class recognition. The most significant example of such success is the CNN architecture, being *AlexNet* [11] the milestone which started that revolution. Krizhevsky *et al.* developed a deep learning model based on the CNN architecture that outperformed by a large margin (15.315 % error rate against the 26.172 % scored by the runner-up not based on deep learning) state-of-the-art methods on the *ImageNet* [19] ILSVRC 2012 challenge.

In addition to model generation for solving open problems [4], [16], the release of pre-trained models alongside with

the architecture of the networks allows for a direct application of the deep learning systems already generated and tested, as it has been done for the place categorization problem [18]. This is possible thanks to the existence of modular deep learning frameworks such as Caffe [9] that provides easy and fast neural architecture setup and the option of load these pre-trained models. The direct application of pre-trained models avoids the computational requirements for learning them: long learning/training time even using GPU processing, and massive data storage for training data. From the existing deep learning models, we should point out those generated from images categorized with generalist and heterogeneous semantic labels [11], [24]. The use of these models lets any computer vision system annotate input images with a set of semantic labels describing their content, as has been recently shown in [6], [15].

III. Using semantic labelling for 3D mapping

Semantic labeling let images to be described by means of a set of semantic concepts attributed to the scene perceived by the camera. This representation is suitable for human-robot interaction, as semantic terms can be easily included in human-robot interaction processes. The use of semantic labels also facilitates the understanding of robot surrounding, which may help to automatically determine the most appropriate robot behavior in any scenario.

To implement this annotation process we make use of existing deep learning annotation tools. Deep learning techniques, and more specifically Convolutional Neural Networks (CNN [13]), allow the generation of discriminant models while discovering the proper image features in a totally unsupervised way, once the network architecture has been defined. This is possible nowadays thanks to the availability of huge image datasets annotated with large and miscellaneous set of semantic labels, which efficiently permits the training of these discriminative classification models. In this work, we focus on the application of existing CNN models. The definition and building of these CNN models is beyond the scope of this paper, so we refer the reader to [1] for a more detailed view of deep learning in general and, to [10] for a better understanding of these CNN models.

A. Global image annotation using CNN

Let $Label = \{label_1, \ldots, label_{|Label|}\}$ be the set of $|Label|$ predefined semantic labels, and I an input image. The direct application of the existing CNN models on I generates a descriptor $d_{CNN}(I) = ([prob(l_1), \ldots, prob(l_{|L|})])$, where $prob(l_i)$ denotes the probability of describing the image I using the i-th label in $Label$. This obtains a representation similar to the Bag of Visual Words (BoVW [21], [14]) approach. To train a CNN model, we need to provide both the architecture of the network and the database to be used as training set. The architecture refers to internal details such as the number of convolutional or fully connected layers, or the spatial operations used in the pooling stages. On the other hand, the training set determines the number of semantic labels used to describe the input image. In this proposal, we take advantage of Caffe [9], a fast, modular and well documented deep learning framework that is widely used by researchers. We opted for this framework because of the large community of contributors providing pre-trained models that are ready to be used in any deployment of the framework.

B. 3D mapping with semantic labels

Given an image, the CNN model provides us with the semantic labels present in the image, so we can expect that the semantic label corresponds to an object present in the image as well. Unfortunately, we are not provided with the position of the image, and therefore the environment, where the object may be placed. So, we propose to move the camera around (left, right, up and down) to look for the limit when the object disappears. Imagine that we move the camera to the left. In that situation, it is expected to have high probability value of a given label (if the object is present in the image). However, when the object disappears, the associated probability decays. If we find the limit when the object appears and disappears, we have a way to determine where the object is (and where it is not). We just need to accumulate all the point clouds where the associated probability to a given semantic label is above a threshold.

We have to implement a mechanism to guide the camera movement. At this moment, this movement is provided by a human but an autonomous system must be developed. The point is that we have to guide the camera using the label probability gradient, searching first for the maximum value of a label and then the limits where the probability label is below the threshold.

This way, when an object is imaged in the current scene the CNN returns a peak probability. As the object disappears of the scene as the camera moves, this probability descends gradually. To deal with this event we propose the application of an hysteresis cycle. We apply the hysteresis cycle as follows: first we set a probability peak threshold Th_1 and a second lower threshold Th_2. When a probability of a given label exceeds Th_1, it means that this object is in the scene so this way it enters the hysteresis cycle. From this point we assume the next captures will have this object as well, and we accumulate them to the point cloud of this object. When the probability of this label is below a second much lower threshold Th_2, it means that this object is no longer in the scene. This event sets the end of the hysteresis cycle so we stop accumulating new point clouds, and we use the last of them to eliminate the exceeding points that do not belong to the object itself but to the background or the rest of the scene. To do so, we calculate the bounding box of this last point cloud that contains information that is no longer related to the given tag. We use this bounding box to remove all the points from the accumulated point cloud that are inside the space described by the bounding box as shown in Fig. 1 leaving, this way, only the points that correspond to the given tag.

As stated earlier, the CNN provides probabilities of finding an object in the scene. That means if we have different objects, which is the common thing, the CNN would provide high probabilities for all of them. As the probability is a value

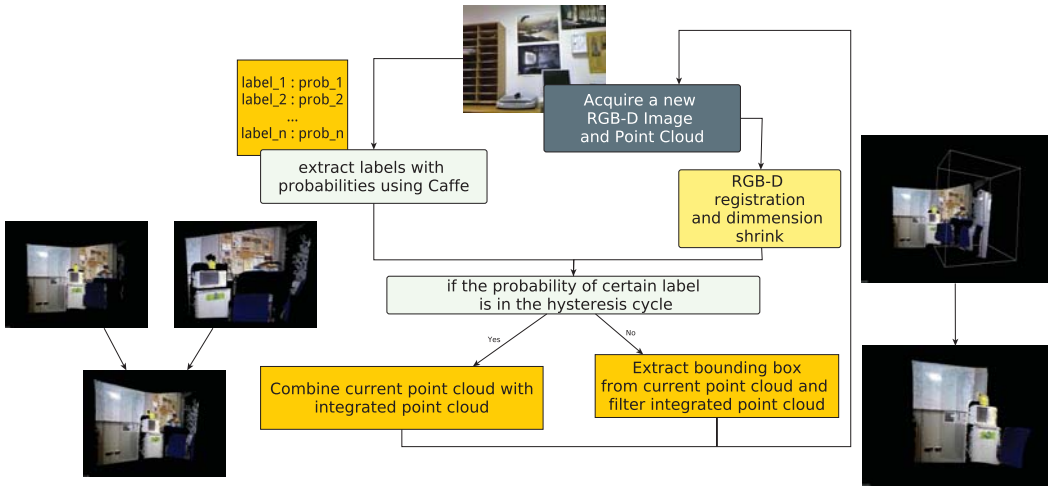


Figure 1. Overall pipeline of the proposal. When a new frame is captured, the RGB-D data is extracted and mapped to a point cloud, then the system registers it to a global coordinate frame. Alongside this process, it extracts the probability of finding objects using a CNN. If the probability exceeds certain threshold and fall into the hysteresis cycle (object is found in the scene), the current point cloud is merged with the point cloud that eventually will hold the 3D object. Otherwise, the bounding box of the current point cloud is calculated and used to crop the integrated point cloud.

normalized between 0 and 1, if we face this situation, the full probability of finding an object would be distributed over these different objects, causing that none of them reaches the detection threshold. To deal with this problem, we propose the use of a preliminary normalization process.

This mentioned process is as follows. As the CNN provides probabilities for over 1,000 different classes, several of them are not interesting to the environment we are about to label, so we initially select the first N labels which are most frequent. Then, each frame is forwarded to the CNN, and we retrieve the probabilities of these N labels and dismiss the rest. As the probabilities were distributed over the 1,000 classes, we need to redistribute them having in mind only the selected classes. This process is performed in order to remove CNN classifying mistakes involving classes that are not relevant to the current environment and to increase the difference between the probability values to make them more easily distinguishable . Fig. 7 shows the probabilities of 8 labels given a frame before and after the normalization process.

It has to be highlighted that the captured point clouds are registered. This is that we transform every new point cloud from the local camera coordinate frame to a global coordinate frame in order to generate a point cloud that represents an object out of several captured points clouds as the camera moves.

Fig. 1 shows a graphical scheme of the proposed method. The algorithm explaining all the phases of the method is detailed in Algorithm 1. The CNN we are using provides probabilities for over 1,000 different labels, so in order to improve the performance of the system we have also removed all the labels do not providing significant semantic meaning to our problem. For instance, as we are working in a laboratory, some labels like animals are not appropriated. The use of semantic

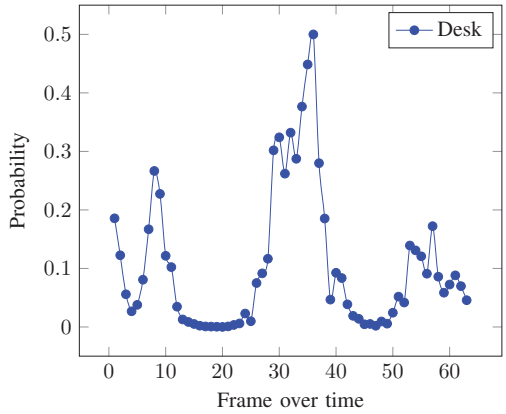


Figure 2. Probability profile of "Desk"

labels has a clear advantage against classical methods, like those using visual features to identify objects. CNN has the ability of recognizing the same category for two objects even when the visual appearance is complete different.

IV. Experiments and Results

The proposal has been evaluated using a real robot in an academic indoor environment. In order to generate the data sequence for the experimentation, we used a mobile robot fitted with a RGB-D camera, namely an Asus Xtion, on top. The robot performed a 360 degrees rotation over the Z axis taking frames every 5-6 degrees, which provided us with 63 different frames. We assume that the pointcloud registration

Algorithm 1 Algorithm to build the 3D map from semantic labels.

Require: Th_1, Th_2: thresholds for hysteresis.

1: $\{PC_{label_i} = \emptyset\}$ a set of 3D pointclouds, one for each label
2: $\{hyst_{label_i} = false\}$ a set of boolean variables, one for each label
3: **loop**
4: Get PC_j, a 3D point cloud and the associated RGB image. The pointcloud must be registered.
5: Get $LabelSet = \{(label_k, prob(label_k))\}$, the set of labels and associated probabilities from the RGB image.
6: **for** each label $label_m$ in $LabelSet$ **do**
7: **if** $prob(label_m) > Th_1$ **then**
8: $hyst_{label_m} = true$
9: **end if**
10: **if** $prob(label_m) > Th_2$ and $hyst_{label_m}$ **then**
11: $PC_{label_m} += PC_j$ // Add the current pointcloud to the map of the given label
12: **else**
13: Get bounding box (BB) of PC_j
14: Remove the points from PC_{label_m} inside BB
15: $hyst_{label_m} = false$
16: **end if**
17: **end for**
18: **end loop**
19: Optional: join all the pointclouds into a single map

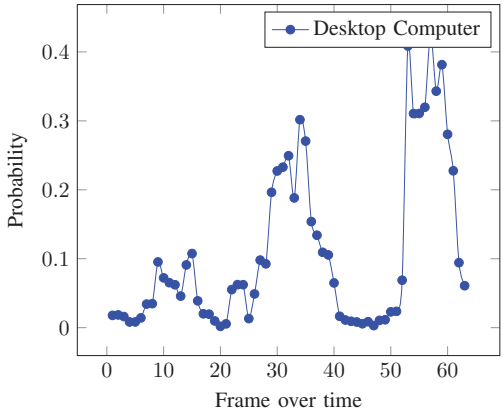


Figure 3. Probability profile of "Desktop Computer"

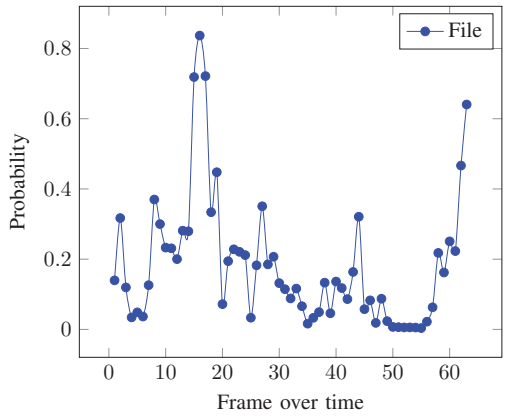


Figure 4. Probability profile of "File"

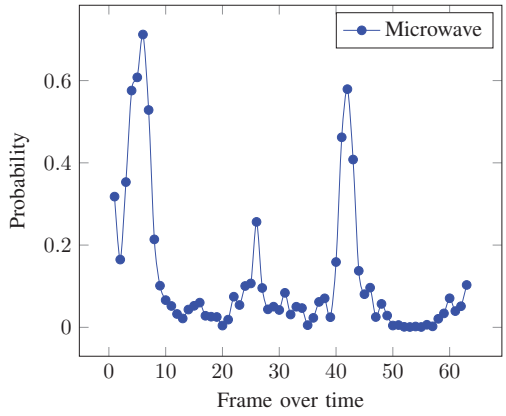


Figure 5. Probability profile of "Microwave"

has been performed (using method like ICP [2], any SLAM technique [7] or other registration techniques such as Rtabmap [12]) so we have all the pointclouds in the same coordinate axis without (or despicable) error. In the experiment, thus we know that the camera rotated 5 degrees over the Y axis between frames (the Y axis corresponds to the vertical one). Therefore, we could easily compute the transformation matrix between consecutive captures. The initial capture was used as the global coordinate frame and we then transformed the following captures applying the convenient rotation matrix to align all the point clouds.

The profiles of the probability values for labels "Desk", "Desktop Computer", "File", and "Microwave" are shown in Figures 2, 3, 4, and 5. For every label, there is one peak which indicates the position of the object. Fig 6 indicates the resulting pointcloud after removing the rest of the points. Regarding the hysteresis parameters of the algorithm, we settled two thresholds: the peak threshold $Th_1 = 0.4$ and the lower threshold $Th_2 = 0.25$. This thresholds have been empirically selected. Also, we extracted the first $N = 15$ most frequent

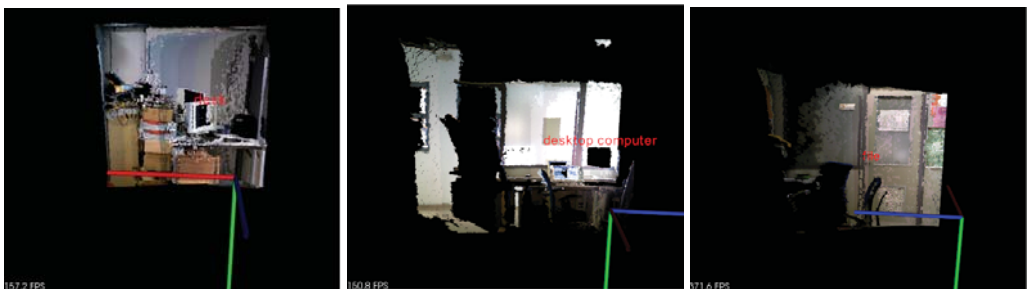


Figure 6. Labelled objects extracted from different point clouds.

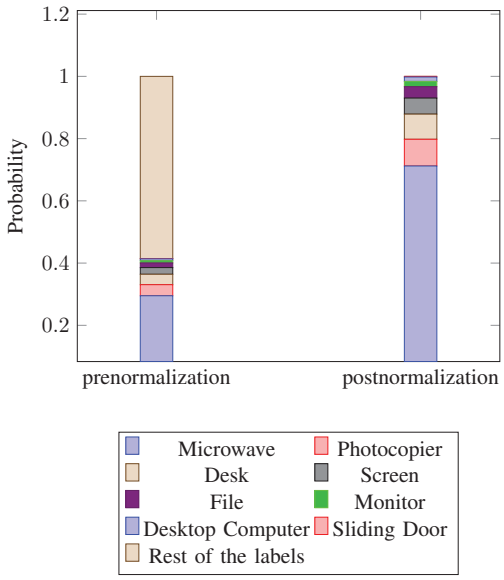


Figure 7. Prenormalization values shows the probabilities of the classes given by the CNN (only the 8 most relevant classes out of the 1,000 full label set) for the frame #6. The postnormalization values shows the probabilities after the normalization process.

objects in our test environment and we found that in this subset there are still irrelevant classes, so we manually removed them. Finally, we selected a subset of 8 different objects from the initial set of 15 most frequent ones, as shown in Fig. 7. This selection was made based on the frequency of appearance in the environment of the experiment. For another environment, we must select the most frequents after a first random capture of the robot. Although ImageNet provides more than 1000 categories and we used only 15, those categories will help to work with different environments. For instance, in a kitchen the most frequent labels are different that for a living room.

As seen in Fig. 3, the profile corresponding to label desktop computer has two peaks. The one reaching 0.3 is below the threshold and then it is not selected. However, a desktop computer is also shown in the pointcloud of the desk (the one in the same position of the previous peak). The reduction of the

Label	Probability	Finally Selected
Refrigerator	219.988	No
Sliding Door	194.273	Yes
Cash Machine	174.483	No
File	162.149	Yes
Dishwasher	116.037	No
Desk	112.222	Yes
Microwave	110.894	Yes
Desktop Computer	78.971	Yes
Monitor	68.187	Yes
Screen	66.352	Yes
Washer	61.099	No
Photocopier	60.067	Yes
Home Theater	49.314	No
Printer	41.118	No
Entertainment Center	30.460	No

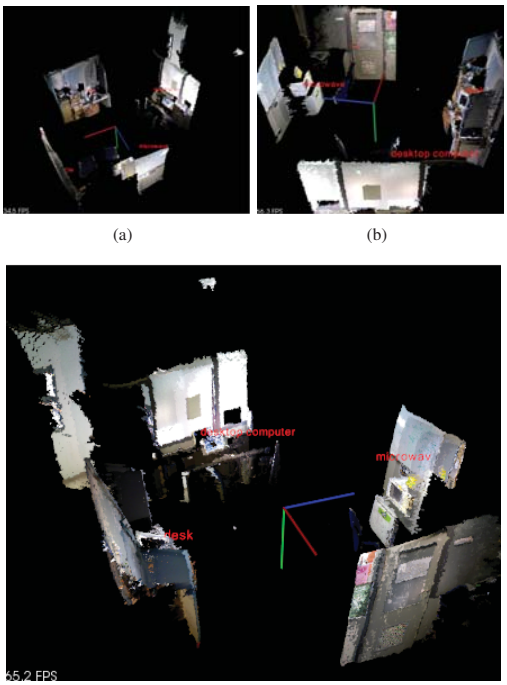
Table I
FIRST N LABELS MORE LIKELY TO APPEAR ACROSS THE SEQUENCE, WITH $N = 15$

thresholds would result into a more peaks not corresponding to the actual class of the object. This threshold selection is critical and a posterior study should be carried out.

Figure 8. Examples of door and file extracted from ImageNet, the dataset used to train the CNN.

It is also worth noted that over the frame 17 of the sequence, the file probability value throws a peak, but as shown in Fig. 6, that frame corresponds to a door. The appearance of it is very similar to a file, as shown in Fig. 8, so the CNN fails to classify this object successfully. So this is another limitation of the method: if there is an object that it has visual appearance

to another one, the CNN could provide a high value to the associated semantic label. Fig. 9 shows the complete map generated after processing four labels (those with the highest probabilities): microwave, desktop computer, desk and file.



(a) (b)

(c)

Figure 9. Final labelled map obtained from 4 labels.

V. Conclusion

In this paper we have presented the first approach to a new perspective for semantic 3D map building. Instead of recognizing objects, we leave this task to an oracle which provides us with the objects present in the image and the probability of those object to be present. We have used a convolutional neural network as an oracle, using Caffe and the model from Imagenet.

With that information, we process the 3D data, incorporating new pointclouds if the object probability is above a given threshold (applying hysteresis). If the probability is below that threshold, the bounding box from the current pointcloud is used to remove the points inside the bounding box from the accumulated point cloud. Results are convincing but more development must be done to get a full valid system. We have showed results using only a rotation (even a translation should provide similar results) around Z axis. In order to get a full 6 degrees of freedom system, several adjustments and experiments must be done.

Acknowledgment

This work was supported by grants DPI2013-40534-R, and TIN2015-66972-C5-2-R of the Ministerio de Economia y Competitividad of the Spanish Government, supported with Feder funds, and by Consejería de Educación, Cultura y Deportes of the JCCM regional government through project PPII-2014-015-P.

References

[1] Y. Bengio. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[2] P.J. Besl and N.D. McKay. A method for registration of 3-d shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[3] P. Bhattacharya and M.L. Gavrilova. Roadmap-based path planning - using the voronoi diagram for a clearance-based shortest path. *IEEE Robot. Automat. Mag.*, 15(2):58–66, 2008.

[4] L. Bo, X. Ren, and D. Fox. Unsupervised feature learning for rgb-d based object recognition. In *Experimental Robotics*, pages 387–402. Springer, 2013.

[5] O. Booij, B. Terwijn, Z. Zivkovic, and B. Kröse. Navigation using an appearance based topological map. In *International Conference on Robotics and Automation*, pages 3927–3932. IEEE, 2007.

[6] G. Carneiro, J. Nascimento, and A.P. Bradley. Unregistered multiview mammogram analysis with pre-trained deep learning models. In *Medical Image Computing and Computer-Assisted Intervention*, pages 652–660. Springer, 2015.

[7] M. Cazorla, P. Gil, S. Puente, J. L. Muñoz, and D. Pastor. An improvement of a slam rgb-d method with movement prediction derived from a study of visual features. *Advanced robotics*, 28(18):1231–1242, 2014.

[8] A. Hermans, G. Floros, and B. Leibe. Dense 3d semantic mapping of indoor scenes from rgb-d images. In *International Conference on Robotics and Automation*, pages 2631–2638. IEEE, 2014.

[9] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia*, pages 675–678, New York, NY, USA, 2014. ACM.

[10] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[12] M. Labbé and F. Michaud. Online global loop closure detection for large-scale multi-session graph-based slam. In *International Conference on Intelligent Robots and Systems*, pages 2661–2666. IEEE, 2014.

[13] H. Lee, R. Grosse, R. Ranganath, and A.Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 609–616. ACM, 2009.

[14] J. Martínez-Gómez, V. Morell, M. Cazorla, and I. García-Varea. Semantic localization in the PCL library. *Robotics and Autonomous Systems*, 75, Part B:641 – 648, 2016.

[15] V.N. Murthy, S. Maji, and R. Manmatha. Automatic image annotation using deep learning representations. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pages 603–606. ACM, 2015.

[16] N. Neverova, C. Wolf, G.W. Taylor, and F. Nebout. Multi-scale deep learning for gesture detection and localization. In *Computer Vision-ECCV 2014 Workshops*, pages 474–490. Springer, 2014.

[17] A. Pronobis, O. Martinez Mozos, B. Caputo, and P. Jensfelt. Multi-modal semantic place classification. *The International Journal of Robotics Research*, 2009.

[18] J.C. Rangel, M. Cazorla, I. García-Varea, J. Martínez-Gómez, É. Fromont, and M. Sebban. Scene classification based on semantic labeling. *Advanced Robotics*, pages 1–12, 2016.

[19] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, An. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[20] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *Computer Vision–ECCV 2012*, pages 746–760. Springer, 2012.

[21] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1470–1477. IEEE, 2003.

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. 2015.

[23] S. Thrun et al. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, pages 1–35, 2002.

[24] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems*, pages 487–495, 2014.

A study on applied cooperative path finding

J. Trullàs-Ledesma, N. ZakeriNejad, D. Quirós-Pérez and Ll. Ribas-Xirgo

***Abstract*—Cooperative path finding tries to determine the best paths for a set of robots to get to their goals with overall minimal cost. Solutions to this problem obtain a set of paths for mobile robots to go to their corresponding, fixed destinations. Their applicability is limited because they are run off-line. In this paper, we analyze cooperative path finding methods as for its possible on-line execution, with dynamically changing conditions and goals. To do so, a simulation of a realistic workshop has been built and several methods tested. Results show that most static cooperative path finding methods can be adapted to dynamic environments.**

***Index Terms*— Automated guided vehicles, autonomous mobile robots, collaborative path-planning, distributed systems, multi-agent systems, physical agents.**

I. Introduction

During the last years, mobile robot technology has been attractive for applications in the industry. Robotic vehicles can be used to automate industrial tasks such as transportation of goods, cleaning, and providing services in different parts of a building. As consequence, automated-guided vehicles (AGVs) are successfully used for internal transportation in the industrial domain.

Despite their favorable contribution to material handling and transportation efficiency in warehousing, manufacturing and automated laboratories, there is still much room for improvement.

The setup of transportation systems and their operation and maintenance are complex and expensive processes that could ultimately be solved by a sort of “plug and play” AGVs (e.g. [1]), if it existed. Following this reasoning, highly autonomous vehicles would be adaptable to many different situations so they could handle the uncertainties and unexpected events taking place in dynamically changing environments.

In this context, navigation is one of the most important problems that mobile robots face. Transportation vehicles must move through the environment to reach different locations and perform the required tasks, and they must do it efficiently.

In multi-robot systems, individual minimum-cost paths can lead to a number of path collisions that make the whole system operate at higher costs than expected.

These conflicts are difficult to solve with methods that are intended for local problem solving [2]. For instance, Fig. 1 shows a simple case where the collision of two paths is better solved by finding an alternative to one of them.

Even in the absence of difficult-to-solve conflicts, maximum efficiency of operation of these multi-robot systems (MRSs) can only be achieved when paths for vehicles that take into account other vehicles’ paths and varying situations that might happen during travelling.

There are two different types of path planning, namely for single robot and for MRSs. In a single robot path planning problem, only obstacles, stationary or not, have to be considered. Indeed, path planning in a network of connected nodes refers to choosing a sequence of arcs from the initial node to the destination node. When dealing with multiple vehicles in the same networks, the possibility of collision among two or more vehicles has to be taken into account. In other words, paths have to be generated so that only one robot occupies one place at one instant of time. Cooperative path planning refers to determining mutually exclusive plans (in time and space) for a group of vehicles.

Because of the large number of states and the high branching factor, the cooperative path planning problem can be very difficult to solve. In fact, the multi robot path planning problem is NP-complete and no polynomial time algorithm to solve the problem is known at this time [3].

The proposed approaches to solve this problem fall into two categories, the one of centralized methods and the one of distributed methods [4]. Centralized methods share a single memory space and program state, but distributed algorithms can be run on each individual agent.

Distributed approaches are more suitable for multi agent systems. In these algorithms there is no single point of failure. The computational time can be divided among all agents at the cost of increasing inter-communication messages.

In this work we shall review the cooperative path finding algorithms (section II) as for their applicability to real cases.

In manufacturing plants and warehouses, new transport orders are being incorporated along the time, so it is no longer possible to keep the initial planning until all transport orders have been fulfilled, as most approaches do.

Moreover, these are dynamic scenarios where obstacles

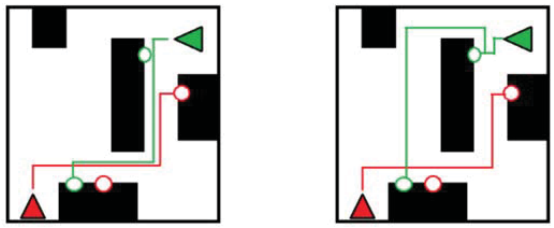


Fig. 1. Example of conflictive shortest paths (left) and cooperative, minimum cost paths (right).

Jonatan Trullàs-Ledesma is with Universitat Autònoma de Barcelona.

Negar ZakeriNejad is currently with Universitat Oberta de Catalunya.

David Quirós-Pérez is currently with Sopra Steria.

Lluís Ribas-Xirgo is with Universitat Autònoma de Barcelona.
E-mail: Lluis.Ribas@uab.cat.

can appear changing the map, and to make it even more complex, new agents could be incorporated during the execution time.

Fortunately, the increased complexity of dealing with the dynamics of real environments is softened by the relative simplicity of the working area layouts and the small number of units being used.

In order to be able to use this study to obtain an appropriate distributed algorithm for path planning in multi agent systems, further simplifications are required.

A major simplification is considered in this work concerning the path formation: path planning is done on a topological map derived from a geometrical one.

The complexity of the path planning is thus drastically reduced so that A* type of algorithms can be used to satisfactorily solve it (section III) for, at least, cases with a few robots on relatively small networks.

Distribution of the algorithms among agents is left as an implementation issue rather than algorithm level problems. In fact, the architecture of the multi-agent systems for transportation that we use [5, 6] organizes systems in two main layers. The topmost one corresponds to agents and the bottom one, to the physical entities of the systems, i.e. to the AGVs. The path planning is done in the top layer, which can be run on a single computer or computer system, or in a distributed fashion, if the top layer is also embedded into the AGVs.

For small area systems with a few vehicles, centralized structures where the top layer (agents) is run on a single computer that communicates with the bottom layer (mobile robots) are simple and relatively robust, provided that they have some redundancy.

Validation of cooperative path planners in real cases is not economically viable. Instead simulation can be used. Section IV is devoted to the different simulation platforms we have built to assess the applicability of path planning in realistic cases.

To focus the study, it is worth noting that cooperative path planning can only be viable for real cases if it can adapt the paths to changes that happen during execution of a solution. Any time a new transportation order is assigned to a mobile robot overall plan execution has to be re-computed. Unexpected obstacles also modify paths costs and inter-dependencies, which makes re-computation necessary.

As a consequence, the key to avoiding interference among planned paths is that algorithm execution times are short enough to be run at higher frequencies than those of changes.

II. State-of-the-art

One of the first approaches to solve the multi agent path finding problem is uses the so-called Local Repair A* (LRA*) algorithm. In this method, each agent uses a standard A* algorithm to look for a route to the destination, and ignores all other agents but the current neighbors. In the second stage, agents begin to follow their routes until an eventual collision is detected. Before each move an agent checks the next part of its path to find out if it is unoccupied by other robots. In case of a collision, the involved agents recalculate paths to their destinations with the A* algorithm. Many various versions of this method have been implemented but they all contain several drawbacks in complex environments which lead to unintelligent behaviors and to create unwanted cycles [7].

Cooperative A* (or CA*, for short) [8] is a hierarchical version of the A* algorithm that uses a 3D reservation table which stores the occupied locations at different times by each agent. After each agent chooses a path to traverse, it reserves the related space time intervals in the reservation table. The other agents have access to the table and based on that, plan their routes. Although the idea of reservation table is worthy, the algorithm still has some problems and may not be practical in critical situation such as narrow corridors.

CA* can be improved by using actual distances instead of estimated Manhattan distances. For this, the paths from the goal to each state must be computed by using A* in reverse.

To limit the depth of the search, a window with a fixed size can be used. Between the fixed intervals, agents resume their search from the last state in a random order.

A method to find an optimal path while avoiding a full A* search for all agents together is presented in [10], and further improved in [11]. The improvement consisted of reducing the number of nodes generated by A* through the use of intermediate states between the regular full states. Another technique partitions the agents into the several groups and then finds optimal paths for each group by using the first technique.

There are also other approaches to solve the problem. For example, it can be re-formulated as a stochastic optimization problem [9]. As such, Differential Evolution (DE) optimization can be applied to solve the problem in both centralized and distributed ways. In [9], they state that the proposed distributed algorithm is comparable to the Particle Swarm Optimization (PSO)-based path planning.

In [15] they demonstrate that the problem is always solvable in a bi-connected graph with at least two unoccupied vertices. Their proposed algorithm (BIBOX) runs in polynomial time. The algorithm is based on the idea that the bi-connected graph can be created adding cycles to an original cycle. Robots that have their target in a cycle are placed in the target location rotating all the robots along the cycle. If the target of a robot is not in the added cycle it is pushed into deeper cycle. They compare BIBOX with other planners like SGPLAN and LPG and claim that it is one order of magnitude faster.

In [16] they achieve an implicit coordination mechanism by means of a taxation mechanism. Their Iterative Taxing Framework (ITF) imposes penalizations (taxes) on agents that pass through specified locations. Bottlenecks are penalized so, instead of waiting n iterations until a node is free, a robot will follow a less penalized path.

In one of the implementations of the ITF [17], all agents share a common goal, and the problem is formalized as a cellular automata based on a $m \times m$ grid of cells. The run times are shorter than the ones for A*, but the algorithm is focused on this concrete domain of problems.

Another alternative is built upon using the direction that agents traveled a portion of the map to compose direction maps. In [18] agents use this information during planning.

Every time an agent crosses a portion of the map, the direction vector of this place is calculated as an average with the previous vectors. Navigating against the direction vector is penalized. Using navigation maps have some drawbacks (deadlocks and insolvable situations), but in certain navigation layouts it can reduce conflicts and collisions because navigation maps tend to form stable "flows" of agents in the same direction. This idea can be used combined with other techniques.

III. Analysis of A*-based Methods

The previous review of multi agent path finding algorithms includes LRA*, CA*, HCA* (hierarchical), and WHCA* (using windows), which are pretty simple and fast [9]. Therefore, they might be readily used for actual cases with relatively clear and small working areas and a few robots.

To verify our implementation of the above-mentioned algorithms and validate their performance as to be applied in dynamic environments, a series of 100 challenging, maze-like environments were generated. Each environment consisted of a 32x32 4-connected grid, and each grid was designated as impassable obstacles given a 20% chance. Any disconnected sub regions were additionally filled with obstacles to guarantee a fully connected map.

Agents were placed into this environment with randomly selected start positions and destinations, subject to the constraint that no two agents shared the same start position or destination. An agent was considered successful if it was able to reach its destination within 100 steps. If the agent exceeded this limit, was unable to find a route, or collided with any other agent, then it was considered to be a failure.

The first experimental results are related to the different window's size for WHCA* algorithm. The average success rates of each algorithm were measured to compare their performance. The number of agents which reach their destination in 100 runs is the same for all the algorithms. With few agents, all of the algorithms are able to reach their destination.

Average path lengths (number of steps) for each agent in 100 runs are shown in Fig. 2. By increasing the window's size, the average length of the paths become shorter. On the contrary, wider windows imply higher runtimes. However, these path calculation times might be right for real-time control at navigation level, depending on the speed of the vehicles, among other factors.

These results show that it is possible to have this algorithm running on-line to re-compute paths as conditions change. However, optimality cannot be guaranteed.

IV. Realistic Scenario

We have seen that off-line, centralized forms of exploration algorithms to find cooperative paths are not as complex as to hinder their application in an iterative, on-line fashion.

However, to validate this hypothesis it is necessary to adapt them to work in a realistic scenario.

In our case, we have chosen a likely floorplan of a workshop, with square workcells and corridors in between.

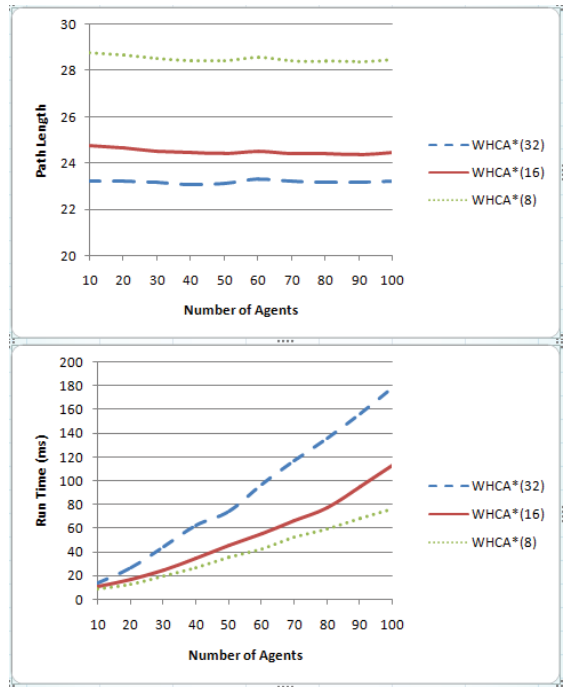


Fig. 2. Average path length for each agent (top) and total runtime (bottom).

Placement of workcells is configurable to enable testing a variety of scenarios. For simplicity, all robots have the same characteristics in terms of capabilities and speed. For this work, they can go to a given destination in terms of landmarks and distinguished positions like crossings and bifurcations, and speed is constant, i.e. independent of battery charge levels or frictional forces. Landmarks consist of visual labels placed at workcell docking positions.

With these capabilities, robot paths can be given in terms of map nodes instead of coordinates, e.g. "go to next workcell at right".

In Fig. 3, a working version of this scenario is shown, with KUKA youBot robots operating in a workshop to move small boxes from one cell to another. Workcells have labels in red that identify the places for boxes or *ports*.

Although this scenario works fine for software

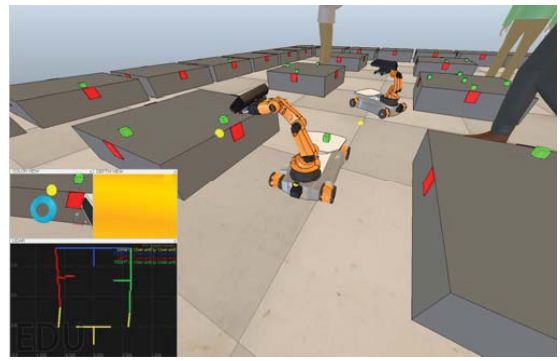


Fig. 3. V-Rep simulation of a plant operated by KUKA youBots [12].

simulation-only cases, it cannot be ported to reality at laboratory scale both for cost and size motivations. Having a prototype of the system enables checking viability of solutions for dynamic cooperative path planning, as for the aspects related to implementation and infrastructure, particularly for all issues about communication. Therefore, cheaper and smaller robots must be used.

In our case, we have built a first version of the real platform (Fig. 4) with differential-drive wheeled robots the size of a Parallax Boebot (it is built on its frame) with two processor boards: an Arduino board to control motors and sensors and a Raspberry Pi to run the higher levels of the local controller stack for the robot. They have an ultra-sound sensor mounted on a servo to be used as a 180-degree front sonar. For simplicity, the US sensor can only be used to create a sonar image when the robot is not moving. While in movement, it is used as a distance sensor for obstacle detection. Essentially, they can do the same operations than KUKA youBots but for the manipulation operations.

Workcells are simulated by cardboard boxes that can be placed as to mimic any workshop.

To be able to switch from software simulation to prototype, the simulated scenarios have been adapted to these robots. They cannot transport any box anywhere but, as for the transportation planning and management issues, it is not a big deal.

We have built two different software simulation environments: one in Netlogo and another in Blender. Both of them will be described next.

A. Netlogo Version

Netlogo is a multi-agent simulation framework built on Java which can be adapted to be used for multi-robot systems [13] like the ones for automated transportation on workshops and warehouses.

The idea is that all high-level aspects of robot systems are run on Netlogo, particularly those which have to do with inter-agent communication.

Hence cooperative path-planning can be executed in the Netlogo framework both in a centralized mode or distributed among agents (i.e. robot representatives at the higher levels of the controlling system). In both cases, transportation agents must communicate with robot counterparts.

The Netlogo version of the workshop is organized in different modules or Netlogo source files, ".nls". The *floorplan.nls* gathers all the functions related to the setup of the scenario, for which it uses methods from *cells.nls* and *topology.nls*. The cells module includes functions to create workcells and place them onto the workshop floor, and the topology module, those related to generate the maps of the working area.

Topological maps are graphs whose nodes represent significant spots on the workshop and whose arcs hold path characteristics between any two connected nodes. Although not used in this work, different travel costs can be assigned to each arc and vehicle. For this experiment, all robots share the same travel costs, and the algorithms are executed in a centralized manner, i.e. a coordinator agent takes robots' destinations and current positions to obtain a set of paths.

The setup of the scenario includes placing the mobile robots on the workshop, but creation of corresponding agents depends on a separated module for transportation agents, so-called *taxis*.

The *taxis.nls* module contains the controller stack for every taxi in the system. There are two distinct levels in accordance to its scope: the system level and the local one.

First level includes all layers of controllers that require system-wide information. It is the level where transport orders are managed and, in the case of centralized path planning, where agents get the routes from the planner.

Controllers at the local level are responsible for anything related to vehicle operation that can be locally managed, including parts of distributed algorithms.

In the Netlogo platform, local controllers also update the view of the system, as shown in Fig. 5.

A module has been added to act as a path planning coordinating agent. In this case, we have kept the same centralized architecture because all taxis share system-wide information that can be used by themselves of other agents at any time. In this case, taxi position at the topological map and destinations are used by the planner agent to provide every taxi with the appropriate path directions.

The path planner takes the goals assigned to every taxi and computes the paths for them to obtain an overall minimum cost solution. The global cost, again, consists of

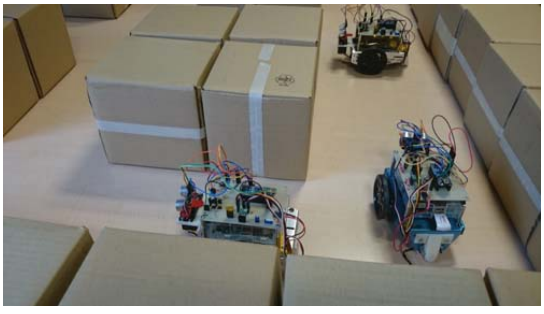


Fig. 4. Prototyping platform for automated workshop transportation systems.

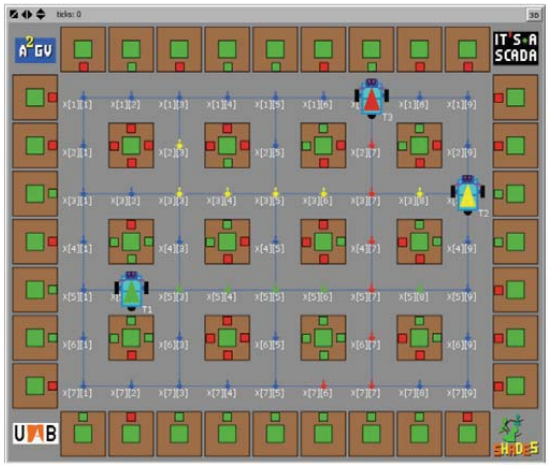


Fig. 5. Simulation of system functioning in Netlogo.

the path length and time, including waits to avoid collisions.

The relative simplicity of the map and the low number of vehicles makes it possible to use A*-type algorithms to obtain the solution paths. Execution times are small enough to be executed on every change in the transport orders or in the environment.

We have observed, though, that there is a cross dependence between transport order assignment and cooperative path finding efficiency. Some assignments make different paths to meet in opposite directions, which, for narrow corridors like the ones shown in Fig. 3, can only be solved by backtracking. Apart from this effect on the path finding execution time, it also may negatively affect the overall cost of the final solution. Thus, it seems reasonable to explore the transport order assignment space with respect to the cost of the resulting paths to achieve the completion of the transport orders for the system.

B. Blender Version

Netlogo allows an easy and fast modeling and simulation but it is not well suited for physical simulation. In fact, the previous platform contains a simple physical simulation engine to update the system state (i.e. all the data agents need to work) and the representation on the screen. However, it is neither prepared to simulate physical phenomena, nor sensors and actuators.

V-Rep, on the contrary, can be used to simulate MRSs and includes several physics engines. Similarly, MORSE [14] can also simulate MRSs.

Netlogo simulation can be concurrently run with V-Rep, MORSE or any other simulator execution or, even, with real robots so that it can take care of the higher level part of the transport controller system.

However, while this solution is interesting to test the whole system on more realistic scenarios it is also costlier for a rapid verification of a broad range of cases.

A balanced option is to directly use Blender, which MORSE is built upon. The idea is to use only the characteristics of Blender that enable simulating cases that are difficult to emulate within Netlogo, such as object detection.

1) Introduction to Blender

Blender is free and open-source software for developing 3D images and films, but some of its features make it interesting for our purposes:

- *Easy 3D modeling*. There are lots of modeling tools included (procedural, box modeling, sculpting...). And lots of free materials and models libraries.
- *A game engine*. The BGE (Blender Game Engine) allows a very easy configuration of an environment with realistic physics properties. Behaviors can be implemented using an intuitive mechanism of building blocks. Those blocks are sensors, controllers and actuators that can be connected using a friendly GUI. In Fig. 6 we show the building blocks of an agent. We have a Python controller invoking *agent.run* function, which can activate *goto* actuators.
- *Python modules*. This is the key feature for using the BGE as a simulation tool. Everything inside Blender can be controlled using Python scripts and allows to open and extend its features. As a sample of opening the system, we have implemented a communications module which can send and receive UDP messages from external systems. And, as a sample of extending the system we can use Python functions as controllers that can process sensor inputs and activate actuators.

All these elements help not only to rapidly generate a simple 3D environment for simulation of workshop transportation systems but also to integrate the top layer of their controllers.

2) System Organization

Our BGE simulation environment is composed by a zenithal camera, a plane and an empty object that we call *SimController* (SC). SC is invisible and has not physical properties, but it can react to the signal of a periodical delay sensor executing a Python function. It controls, on each simulation tick, which task must be executed. SC initializes the environment, processes communications, instantiates agents, maze walls, creates navigation topology, *et cetera*.

Fig. 7 shows a simple sample of a maze with two blue

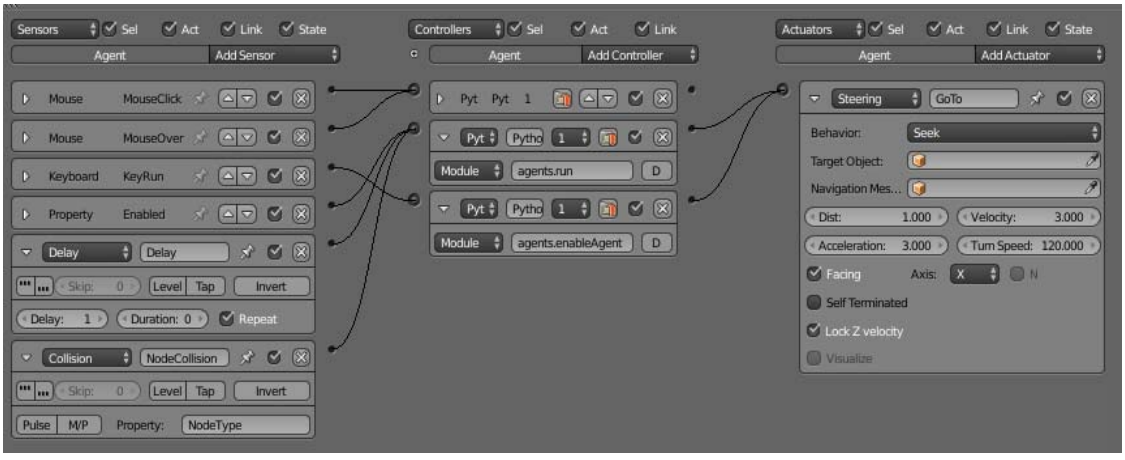


Fig. 6. BGE, building blocks GUI, sample for a transportation agent.

agents and a single goal (a sphere with red cones). In this case we visualize both agent paths.

Python functions are distributed in different external modules which are invoked by BGE controllers. The main modules are:

- *SimMaster.py*: Main functions invoked by SC and callbacks used in simulation events.
- *Common.py*: Common useful functions. Here we can find exporting to log file, for instance.
- *SimComm.py*: Communications related functions. Here we have an UDP server/client, and message processing classes. It can be used for communicating with Netlogo in case the top layer of the MRS is run on it and Blender is used to both simulate the bottom layer of robots and provide the system with a realistic visualization of its working. Also, it can communicate with real robots to make a concurrent execution of a Blender simulation and the reality.
- *Scenes.py*: This module includes functions for building scenarios. These functions instantiate and configure BGE elements according to user definitions.
- *Topology.py*: As we are testing graph-oriented navigation algorithms this module includes functions for creating and debugging a graph over our 3D environment. This "topology" can be visualized as a group of connected spheres (nodes).
- *PathFinding.py*: Those pathfinding functions and classes can be used for agents, and extended if needed. Here we can find functions like *AStar*, *HeuristicDistance*, *PushAgent*, *et cetera*.
- *Agents.py*: Transportation and other agents of the system. This module contains a collection with an instance of *Agent* class for each Blender Agent instantiated in the SC. *Agent* class contains a reference to 3D object representing the agent and all methods that can be invoked by BGE controllers. Furthermore, *Agent* class contains properties needed for pathfinding algorithms, as current path, current goal, possible local copy of topology, *et cetera*.

As with Netlogo, cooperative path finding does not take much CPU time to execute on Blender. However, we have not tested cases with unexpected cases or sensor/actuator failures, which would cause an increase on the computational complexity of the path finding problem.

Fig. 7. Zenithal view of a workshop in the Blender platform.

V. Conclusion

In general, cooperative path finding is complex to solve, much more under continuous transport order updating and within dynamically changing environments.

Fortunately, complexity of this problem is relaxed by using topological maps instead of navigation meshes, cell-based or geometrical maps because the quantity of nodes diminishes dramatically. Also, in many real situations, the number of robots is not very high.

We have developed several simulation frameworks for typical workshops to test the applicability of cooperative path finding algorithms.

When running random cases with the WHCA* algorithm, the execution times on an inexpensive computer have been short enough to not need any parallel execution to be usable on-line.

We have also tested similar algorithms with more realistic cases on different platforms and results have proved consistent with the previous statement.

However, even the centralized approach can be applicable, it requires an additional agent having access to the taxis common blackboard memory to be run, which makes the solution less plug-and-play than a distributed one.

In the near future we shall prove the viability of path-planning under changing situations and with sequences of transport orders along time, as both of them are factors that increase the complexity of the problem.

References

[1] —, *PAN-Robots: Plug&Navigate robots for smart factories*. EU FP7-ICT, no. 314193, 01.11.2012-31.10.2015. [http://www.pan-robots.eu/]

[2] A. Kimmel and K. Bekris, "Decentralized Adaptive Path Selection for Multi-Agent Conflict Minimization," in *Proc. of the 24th Int'l. Conf. on Automated Planning and Scheduling* (ICAPS), 21-26 June 2014, Portsmouth, USA.

[3] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *Proc. of IEEE Int'l. Conf. on Robotics and Automation* (ICRA), vol. 1, pp. 476–481, 2000.

[4] J. Chakraborty, A. Konar, L. C. Jain, and U. K. Chakraborty, "Cooperative multi-robot path planning using differential evolution," *J. Intell. Fuzzy Syst.*, vol. 20, no. 1, pp. 13–27, 2009.

[5] Ll. Ribas-Xirgo, A. Miró-Vicente, I. F. Chaile, and A. J. Velasco-González, "Multi-Agent Model of a Sample Transport System for Modular In-Vitro Diagnostics Laboratories," in *Proc. of the 17th IEEE Int'l. Conf. on Emerging Technologies and Factory Automation* (EFTA), Kraków, Poland, 2012.

[6] Ll. Ribas-Xirgo and I. F. Chaile, "An Agent-based Model of Autonomous Automated-Guided Vehicles for Internal Transportation in Automated Laboratories," in *Proc. of the 5th Int'l. Conf. on Agents and Artificial Intelligence* (ICAART), 15-18 Feb. 2013, Barcelona, Spain.

[7] M. Čáp, P. Novák, J. Vokřínek, and M. Pěchouček, "Asynchronous Decentralized Algorithm for Space-Time Cooperative Pathfinding," in *Int'l. Wkshp. on Spatio-Temporal Dynamics at ECAI*, 2012.

[8] D. Silver, "Cooperative pathfinding," in *Proc. of the 1st Conf. on Artificial Intelligence and Interactive Digital Entertainment* (AIIDE'05), 2005, pp. 23–28.

[9] J. Chakraborty, A. Konar, U. K. Chakraborty, and L. C. Jain, "Distributed cooperative multi-robot path planning using differential evolution," in *Evolutionary Computation, IEEE World Congress on Computational Intelligence* (CEC), pp. 718–725, 2008.

[10] T. Standley, "Finding optimal solutions to cooperative pathfinding problems," in *Proc. of the 24th AAAI Conf. on Artificial Intelligence* (AAAI), pp. 173–178, 2010.

[11] T. Standley and R. Korf, "Complete algorithms for cooperative pathfinding problems," in *Proc. of the 22nd Int'l. Joint Conf. on Artificial Intelligence*, vol 1, pp. 668–673, 2011.

[12] Ll. Ribas-Xirgo, "A Virtual Laboratory of a Manufacturig Plant operated with Mobile Robots," in *Proc. of the 19th IEEE Int'l. Conf.*

on Emerging Technologies and Factory Automation (EFTA), Barcelona, Spain, 2014.

[13] N. R. Ramli, S. Razali and M. Osman, "An overview of simulation software for non-experts to perform multi-robot experiments," in *Int'l. Symp. on Agents, Multi-Agent Systems and Robotics* (ISAMSR), pp. 77–82, Aug. 2015.

[14] G. Echeverria, S. Lemaignan, A. Degroote, S. Lacroix, M. Karg, P-Koch, C. Lesire and Serge Stinckwich, "Simulating Complex Robotic Scenarios with MORSE," in *Proc. of the 3rd Conf. on Simulation, Modeling, and Programming for Autonomous Robots* (SIMPAR), pp. 197–208, 2012.

[15] P. Surynek, "A Novel Approach to Path Planning for Multiple Robots in Bi-connected Graphs," in *Proc. of the IEEE Int'l. Conf. on Robotics and Automation* (ICRA), Kobe, Japan, 3613–3619, 2009.

[16] Z. Bnaya, R. Stern, A. Felner, R. Zivan, and S. Okamoto, "Multi-agent path finding for self interested agents," in *Proc. of the 6th Annual Symposium on Combinatorial Search*, pp. 38–46, July 2013.

[17] Y. Tavakoli, H.H.S. Javadi, and S. Adabi, "A cellular automata based algorithm for path planning in multi-agent systems with a common goal," IJCSNS, 8(7):119, 2008.

[18] R. Jansen and N. Sturtevant, "A new approach to cooperative pathfinding," in *Proc. of the 7th Int'l. Joint Conf. on Autonomous Agents and Multiagent Systems*, Vol. 3, pp. 1401–1404, 2008.

Multi-thread impact on the performance of Monte Carlo based algorithms for self-localization of robots using RGBD sensors

Francisco Martín, Vicente Matellán and Francisco J. Lera

***Abstract*—Using information from RGBD sensors requires huge amount of processing. To use these sensors improves the robustness of algorithms for object perception, self-localization and, in general, all the capabilities to be performed by a robot to improve its autonomy. In most cases, these algorithms are not computationally feasible using single-thread implementations. This paper describes two multi thread strategies proposed for self localize a mobile robot in a known environment using information from a RGBD sensor. The experiments will show the benefits obtained when different numbers of threads are compared, using different approaches: a pool of threads and creation/destruction scheme. The work has been carried out on a Kobuki mobile robot in the environment of the RoCKiN competition, similar to RoboCup@home.**

***Index Terms*—localization, 3D maps, RGBD sensors, octree, Multi-threading**

I. Introduction

We are interested in developing software for robots that help people in domestic environments. This software consists of many processes that must be running at once on board an autonomous robot. To carry out household tasks, a robot must perceive objects and people, and be able to interact properly with both. In addition, robots navigate along their environment to perform these tasks, so it is important to have a reliable navigation.

In recent years, 3D sensors are becoming very popular as standard equipment in mobile robots. Nevertheless, in most cases they don't take advance of this information. Maps usually describe 2D information, and created either directly from lasers 2D or transforming their RGBD information to 2D distances. In many works, the software used is based on this idea [1]. While the laser is a very precise sensor, we think they are missing the 3D capabilities of RGBD sensors, which could be very beneficial in environments with symmetries in 2D, but with a rich 3D structure and color. Although the use of RGBD data can improve the capabilities of the robot, it presents a performance problem. The amount of information from this sensor is twice that of a normal image, since each pixel also includes spatial information. The processing of this cloud of points is not easy because usually requires spatial transformations and distance calculations.

Francisco Martín is with University of Rey Juan Carlos.

E-mail: francisco.rico@urjc.es, vicente.matellan@unileon.es, fjrodl@unileon.es

Vicente Matellán and Francisco J. Lera is with University of León, León.

To make a robotic system that requires so much computing resources feasible, it is necessary to pay attention to how the processes that make up the system are executed. Usually, each of these processes usually correspond to a kernel process, delegating the scheduling to the operating system or libraries. Sometimes a single process performs very heavy computation that can not be addressed as a pipeline. Processing point clouds RGBD sensor is an example of this. These sensors are becoming increasingly popular because they provide useful information for reconstruction of objects, mapping and, in general, any process that benefits from having a 3D structure with color. The processing of these clouds of points is computationally expensive. PCL library [2] has become the standard way to handle these point clouds. In addition to its integration with ROS, it offers a lot of tools for segmentation, recognition, feature extraction and correlation points, among others. Many of the functions have a GPU version, which speeds up the processing of these clouds. On other occasions, functions as point clouds searches have not a GPU version available, being a real bottleneck in many applications. In addition, GPU programming requires a low-level programming in which most of the available libraries for scientific computing are not available.

Using multiple threads is a strategy to improve the efficiency of a process with large computing requirements. The optimal number of threads depends on the number of cores of the processor that is running the software. The ideal is to distribute the work in a number of threads similar to the number of cores. Current conventional computers have multiple cores available, and they are not always completely exploited. Most of the implementations use a single thread approach for all the computation.

To validate our research in this field, we participate in robotic competitions that simulate a home environment: RoCKiN [4] and RoboCup@home [3]. The competitions have proved to be a good tool to advance robotics. They present a common scenario where different groups can evaluate and compare their research using standardized tests which can measure the performance of each proposal. In this competition the robot must develop a series of activities to help a person with a degree of dependence in their home environment. These missions include to receive and recognize visitors, to find objects around the house or taking a glass of water from the kitchen to the bedroom. It is also important that the robot share its environment with humans safely.

In this paper we present two multi thread strategies to

perform the self localization based on Monte Carlo algorithm [5] using RGBD information intensively. We will demonstrate how a multi-thread approach can make this approach feasible in real time. In order to distribute computing in multiple threads, we have developed two strategies: a pool of threads and threads that are created and destroyed. We will test both strategies with different numbers of threads, from 1 to twice the cores available. The direct consequence is an improvement in the response time of the self localization algorithm, and thus the execution frequency. The higher this frequency, the greater the speed at which a robot can move safely, thereby improving the time in which a robot can reach from one position to another in the environment.

The remainder of this paper is structured as follows. After discussing related work in the following section, we will briefly present the self localization algorithm to be improved in section III. The main contribution of this paper, the multi thread approach will be presented in section IV. In Section V we then will present experimental results of the multi thread comparison. Finally, in section VI we will present the conclusions and future works.

II. Related work

The organization of the execution of software is critical in robot programming. In a system that requires real-time characteristics, where there are deadlines for performing operations. If these deadlines are not met, the result can not be valid, or even disastrous for operation of the robot. Developing robotic systems with such characteristics have led to several works [6][7] focused on systems and methodologies whose emphasis is real time. In many cases, the real-time conditions are applied to the access to sensors and actuators [8][9], while these conditions are relaxed for rest of the software.

In other cases, soft real-time conditions are enough. These approaches uses a graceful degradation, as in the case of Bica [10], where one thread performs all the computation of the robot. Each module executes at its own frequency, forming an activation tree representing its execution dependencies. The execution is performed in cascade, ensuring the execution frequency of each one. If a process exceeds its time slice, the system degrades delaying the execution of others, doing their best to recover their frequency. Our proposal is capable of performing a similar degradation, but taking advantage of multi-core processors of current architectures. Our proposal manages to avoid concurrency problems, but at the cost of a less generalizable solution.

ROS [11][1] is currently the standard in robot control systems. A robotic application in ROS is composed by several interconnected nodes. Drivers for accessing the robot sensors and actuators are implemented using nodes, that provide a common interface by standard messages. Each node has a thread for the user and others responsible of managing communications. There are no race conditions between these threads because it is explicitly synchronized when data handlers of these message queues are executed and when the user code is executed. From the point of view of the user, the nodes run in a single thread. The way to balance the computation load is to design the complete process as a pipeline of ROS nodes. The operating system scheduler distributes the execution of the nodes in the available resources.

The processing design presented in this paper would have been implemented in ROS creating several ROS nodes that divide the work. Besides the problems of synchronization between processes, the communication delay makes not feasible this way. Our proposal is running on a node ROS, but creating threads inside it in a very controlled way.

A very effective way to speed up processes that require large amounts of computing is to use the power offered by the Graphic Processor Unit (GPU) [12]. These components contain many computing units that offer perform extensive computations in parallel [13]. This approach is useful when the same operation is applied to large amount of input data. GPU parallelization techniques have been used in expensive processes, like training neural networks [14][15] or Neural Gas Accelerated [16][17]. RGBD data processing requires great computing power, so the GPU parallelization techniques have been used extensively in this field [18]. In [20], a SLAM algorithm that uses information from a sensor RGBD is presented. In this paper, it uses a GPU parallelization to make an algorithm real-time properties. Our approach is different because we speed up the process using multiple threads on the CPU, instead of a GPU. On one hand, we get less performance, but on the other hand, it allows the use of standard libraries.

The problem of self localization of a mobile robot has received great attention from the start of the Mobile Robotics. Self localization methods estimate the most probable position of a robot in its environment using the information from its sensors and a from a map. Kalman Filter [21][22][23][24] is one of the first widely used estimator for nonlinear systems. This method maintains a gaussian state estimation of the robot position, and its associated uncertainty. This method has difficulties to start if the initial position is unknown, or when kidnapping situations occur. There are works that try to overcome these limitations, maintaining a population of extended Kalman filters [25].

Other popular methods are called Markovian. These methods represent the environment in states that may correspond to a regular grid [26] or irregular regions [27][28]. The probability associated to each state is updated using Bayes' Theorem, establishing a priori probability of possible observations in each state. The main advantage of this method is that it is *global*. This means that it can maintain multiple hypotheses about the position of the robot and it can recover from situations of kidnappings or unknown initial positions. The main disadvantage of this method is its high computational cost when high precision is required, or when the environment is very extensive. In [29] this problem is addressed by dynamically varying the number of cells and the size of the grid, but this complicates how the probability of some states are updated when the robot moves.

Currently, the most widely used methods are based on particle filters [5][30][31], also called Monte Carlo method [32]. This method is based on sampling a probability distribution by a set of hypotheses, called particles. Those particles most likely will remain in the population, while the less likely ones

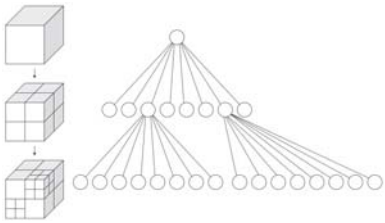


Fig. 1. Map organization in an octree.

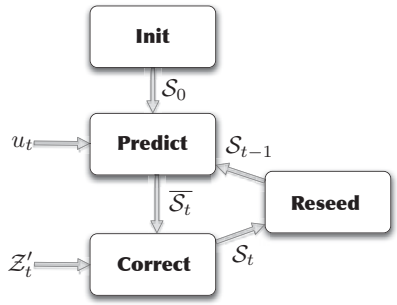


Fig. 2. Monte Carlo Algorithm scheme.

will be replaced by others. The robot position is determined by the accumulation of particles with high probability. This method is independent of the size of the environment, and has been used in many applications [33][34]. Furthermore, this method is very flexible and allows many optimizations. In [35], new particles created in each cycle are not generated at random, but taking into account the latest perceptions. It can use a variety of perceptions, achieving a very robust method in highly dynamic environments such as [36] where applied to legged robot in robot soccer matches.

Neither there are enough jobs on using 3D maps for self location. In [37], they use RGBD information, but to find 2D planes, using a 2D map. In [38], the navigation of a robot with legs is made using a 3D map of the environments is made, although the self location information is performed using only a laser distance. Our approach takes full advantage of the position information and color offered by the RGBD sensor and a 3D map of colored dots.

III. Self Localization Algorithm

A. Map

A map contains the knowledge of the environment that the robot uses to self localize. In our work we use RGBD sensors to create the map, so that our map is made up of colored 3D points. A map $\mathcal{M}$ is a set of points with a position and a color in the HSV color space, (x, y, z, h, s, v). This set of points is structured as an octree [39], as shown in Figure 1. It is a tree structure in which each node subdivide the space into eight octants. Using an octree, it is efficient [40] to calculate the points in an area, or the neighbors of a particular point.

In general, building an octree has a complexity $O(Nlog(N))$. Searching on a map has a complexity of $O(log(N))$ in the best case. The search operation $find(\mathcal{M}, p, R)$ returns a set of points $\mathcal{MS} \subseteq \mathcal{M}$ starting from a point p and radius R, in which,

$$find(\mathcal{M}, p, R) \rightarrow \mathcal{MS}, dist(p, p_j) < R, \forall p_j \in \mathcal{MS} \quad (1)$$

Moreover, this set is ordered, so,

$$dist(p, p_n) < dist(p, p_m), \forall p_n, p_m \in \mathcal{MS}, n < m \quad (2)$$

B. RGBD Monte Carlo localization

The state of a robot s can be seen as a displacement (x, y, z) and a rotation (ϕ, θ, ψ), which form an invertible transformation from the origin of coordinates on the map to the robot. In the case of wheeled robot, notation position can be simplified to (x, y, ϕ).

To estimate the robot position, we will use a Monte Carlo algorithm, whose scheme is shown in Figure 2. This algorithm samples the probability distribution $bel(S_T)$ that represents the position of the robot as a set $\mathcal{S}_t$ of hypotheses about the position of the robot S_T, also called particles,

$$\mathcal{S}_t = \{s_t^1, s_t^2, \cdots, s_t^N\} \quad (3)$$

each element of $\mathcal{S}_t$ is a hypothesis S_T associated to an probability so

$$s_t^i \sim p(s_t^i | z_{1:t}, u_{1:t}) \quad (4)$$

The accumulation of particles in a region of the environment indicates the position of the robot.

Initially, S_0 is set depending on the a priori knowledge that we have about the position of the robot. If we start from a known state $s_{initial}$, $s_0^n = s_{initial}, \forall s_0^n \in \mathcal{S}_0$. This problem is usually called *tracking*, because the problem focuses on correcting the errors in the odometry using sensory information. On the other hand, If $s_{initial}$ is unknown, $bel(S)$ is uniformly distributed on the set of possible states of $\mathcal{S}$. This problem is more complex, and is called *global* self location problem, where we have to determine the position of the robot from a situation of complete ignorance. This method is effective to solve both problems.

In the prediction phase, u_t represents the transformation (rotation and translation) of s_{t-1} to s_t measured by the proprioceptive sensors of the robot. This is the expected displacement depending on the control commands generated at each instant t. The application to a state s_{t-1} produces the *prediction* of the state *a priori* s_t, or $\overline{s_t}$.

$$\overline{s_t} = s_{t-1} * u_t \quad (5)$$

In the *prediction* phase, particles in $\mathcal{S}_{t-1}$ are updated using the transformation u_t, applying a noise n_t so

$$n_t \sim N(0, \sigma_u^2) \tag{6}$$

that is a gaussian error that is expected to be u_t, which follows a normal distribution. In wheeled robots, where this approach has been applied, the standard deviation σ_u^2 is low, since the odometry information is very reliable.

In the *correction* phase, we update $\overline{\mathcal{P}_t}$ to $\mathcal{S}_t$ using the perception Z_t. Under normal conditions, Z_t can be composed for nearly 300000 elements. Such amount of information makes computationally not feasible to use this full set to calculate the weight of each particle in $\mathcal{S}_t$. The number of times we calculate $p(\overline{s_t^i}|z_t^j), \forall \overline{s_t^i} \in \overline{\mathcal{S}_t}, \forall z_t^j \in Z_t$ can be 640×480 points $* 200$ particles $= 61440000$. The calculation of $p(\overline{s_t^i}|z_t^j)$ involves comparing each point z_t^j with its neighbors in the map $\mathcal{M}$, which increases the computational requirements of the whole process. In addition, we want run our algorithm several times per second, ideally between [10-20]Hz, to be used by a navigation algorithm.

To make possible the execution of our algorithm to this frequency, we do not use the full perception Z_t, but randomly select a subset $\mathcal{Z}_t'$ of $\mathcal{Z}_t$, so $|\mathcal{Z}_t'| < N$, where $N \in [100-500]$. This range of values facilitates the execution of the algorithm at an acceptable rate, and it is significant enough to update the probability of $\overline{\mathcal{S}_t}$, as will be shown in experimentation section.

For each element of $\overline{\mathcal{P}_t}$ we calculate a weight w_i, which corresponds to a probability given the set of perceptions $\mathcal{Z}_t'$.

$$w_i = \frac{1}{|\overline{\mathcal{S}_t}|} \sum_{i=1}^{|\overline{\mathcal{S}_t}|} \sum_{j=1}^{|\mathcal{Z}_t'|} p(\overline{s_t^i}|z_t^j) \tag{7}$$

$$p(\overline{s_t^i}|z_t^j) = \frac{p(z_t^j|\overline{s_t^i}) * p(\overline{s_t^i})}{p(z_t^j)} \tag{8}$$

Considering that $\overline{s_t^i}$ represents a transformation from the origin of the map, we can calculate the position of z_t^j in the map.

$$l = z_t^j * \overline{s_t^i}^{-1} \tag{9}$$

As we saw in the section III-A, it is possible to obtain a set $\mathcal{MS}$ using the function $find(\mathcal{M}, p, R)$. The probability $p(z_t^j|\overline{s_t^i})$ is calculated from similarity of these two points based on the color difference and in the distance in position.

In the last part of the algorithm, we create a new set $\mathcal{S}_t$ from $\overline{\mathcal{S}_{t-1}}$ after incorporating u_t and $\mathcal{Z}_t$. This phase is known as *resampling*. Figure 3 shows this process. $\overline{\mathcal{S}_t}$ is represented at the top of this figure as an ordered vector, where the color indicates the weight w_i of each particle $\overline{s_t^i}$ in $\overline{\mathcal{S}_t}$. Particles whose w_i is higher are placed at the beginning (in green) and the least likely (in red) are placed at the end.

In our approach we perform resampling in two steps. 50% of the most likely particles remain of $\overline{\mathcal{S}_t}$ to $\mathcal{S}_t$, while the other 50 % are removed. Next, these particles are replaced by others generated from the existing ones in $\mathcal{S}_t$, initialized to $w_i = \frac{1}{|\overline{\mathcal{S}_t}|}$. In our approach, we use the first 25% of the most likely particles to perform the generation of the new ones.

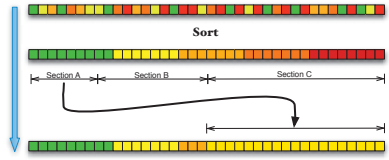


Fig. 3. Resampling of $\mathcal{S}_t$.

Through this process, in each cycle the less likely particles are eliminated, and are replaced by others in positions where it is more likely to be robot. This method is effective to solve the problem of *tracking*, where there is already a set of initial particles is supposed to be close to the actual position of the robot. The new particles will correct errors in u_t with the information from sensory $\mathcal{Z}_t$.

IV. Multi-thread approach

The workflow of the Monte Carlo algorithm is an iterative pipeline, as shown in Figure 2. In the prediction phase, u_t applies to all particles $s_t^i \in \mathcal{S}_{\sqcup -\infty}$, which is not a heavy workload. In contrast, the correction phase needs many more computational resources. For every particle $s_t^i \in \overline{\mathcal{S}_\sqcup}$ a test is performed using the perception points $\mathcal{Z}_t' \subset \mathcal{Z}_t$. This test has several steps:

1) To apply to $z'^i_t \in \mathcal{Z}_t'$ the transformation that s_t^i represents, obtaining l_i.
2) For each point l_i, we make the set $\mathcal{MS}$ using the $find(\mathcal{M}, l_i, R)$ function, where $|\mathcal{MS}| \leq 20$. One reason to use threads instead of GPU is this function $find$, which is the one that consumes more resources. Without a real GPU implementation it becomes in the real bottleneck in this processing.
3) To accumulate the probability calculated from comparing the distance metric and color betweenz'^i_t and every element in $\mathcal{MS}$.

This amount of work is divided into a set of threads $\mathcal{T}$, as shown in Figure 4. It is assigned a subset of $\mathcal{S}_{\sqcup -\infty}$ to each thread, to apply the phases of the Monte Carlo algorithm.

The response time of a process is the time since it starts until the result is obtained. The response time depends on the number of threads $|\mathcal{T}|$ that can run in parallel. On a UNIX system, each thread is mapped to a kernel process, and the operating system scheduler assigns each thread to each processor core. For this reason, when $|\mathcal{T}|$ is greater than the number of processor cores, the response time of the system does not improve.

Creating a POSIX thread on Unix/Linux is relatively inexpensive. As it is a kernel process that shares almost all its memory with the rest of the threads in the same user process, creating a thread consists in reserving space for its stack and its registers. Creating a POSIX thread in C ++ is to create a `std :: thread` object, specifying a function to run and its arguments. The parent thread can synchronize with the new threads through several mechanisms. The simplest is by `join`, which is a function that blocks the calling thread until one of his sons ends.

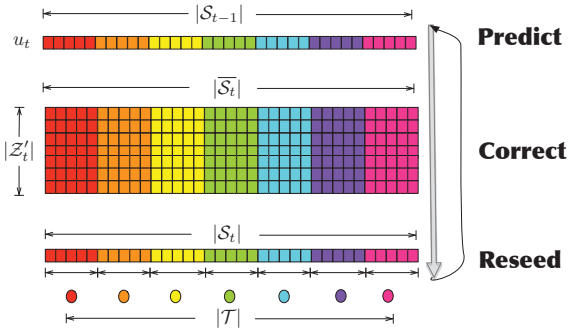


Fig. 4. Workload assigned to each thread, represented as circles, in $\mathcal{T}$.

The following code shows the multi thread approach in which $\mathcal{T}$ threads are created and destroyed in each iteration to process a subset of $\mathcal{S}_{\sqcup-\infty}$. Synchronization follows a `create/join` scheme. For simplicity, we show only the code of the correction phase. Other phases are similar, in terms of number of threads and synchronization scheme.

```
int numParts = 200;
int numThreads = 8;
int numPercepts = 20;
Particles parts[numParts];
Points percept[numPercepts];

void doCorrect(int init, int end)
{
        for(int i=init; i<end; i++)
                update(parts[i], percept);
}

void correct()
{
        thread T[NumThreads];

        for(int i=0; i<NumThreads; i++)
        {
                int start = i*(numParts/numThreads);
                int end = (i+1)*(numParts/numThreads);

                T[i] = thread(doCorrect, start, end);
        }

        for(int i=0; i<NumThreads; i++)
                T[i].join();
}
```

This approach is valid and it works properly. The drawback is that we create and destroy 8 thread per phase in each iteration of the algorithm. If the frequency is 10 Hz, we create 240 threads per second, 14400 threads per minute. Usually, in a Unix system there is a maximum number of threads that can run simultaneously, although this limit does not affect us because it is always 8 in our case. The relevant limit in this case is the maximum number of PIDs that it is configured the system. In long operations, this limit is reached easily.

The most convenient approach in this case is to maintain a pool of threads that are created at the beginning. These threads wait to be request to start processing, and when they finish are suspended until they return to be required. Still, we must be careful because:

- It is no possible to abort a thread from a pool of threads.
- It is nos possible to determine when a thread in a pool has finished.

For this reason, we must be very careful when designing a solution based on this approach. The threads of the pool must be perfectly synchronized to run only at certain times, when the data to be processed are available. In addition, it is necessary that the main thread knows when all the threads have finished. To coordinate this type of scheme we can use many mechanisms: mutexes, locks, barriers, conditions, etc. In our implementation we used semaphores. This synchronization mechanism is very simple:

- The semaphore S is created with an initial value of N.
- If a thread calls the `wait()` method of S, N is decremented in 1. If $N < 0$, the calling thread suspends in S.
- If a thread calls the `post()` method of S, N is incremented in 1. If $N > 0$, N blocked threads in S are activated.

```
int numParts = 200;
int numThreads = 8;
int numPercepts = 20;
Particles parts[numParts];
Points percept[numPercepts];
thread T[NumThreads];

void initThreads()
{
        for(int i=0; i<num_threads_;i++)
                start_sems = semaphore(0);

        end_sem=semaphore(0);

        for(int i=0; i<NumThreads; i++)
        {
                int start = i*(numParts/numThreads);
                int end = (i+1)*(numParts/numThreads);

                T[i] = thread(doCorrect, start, end);
        }
}
void doCorrect(int init, int end, int th_id)
{
        while(true)
        {
                start_sems[th_id]->wait();
                for(int i=init; i<end; i++)
                        updateProbs(parts[i], percept);
                end_sem->post();
        }
}

void correct()
{
        for(int i=0; i<num_threads_; i++)
                start_sems[i]->post();
        for(int i=0; i<num_threads_;i++)
                end_sem->wait();
}
```

In the previous source code shows that each thread `T[i]` that is created to perform the correct phase has its own semaphore, `start_sems[i]`, initialized to 0. Threads are created after initializing these semaphores. All of them run the `doCorrect()` function. The id argument is used by each thread in this function to identify its own semaphore `start_sem[id]`. All the threads then are blocked in each one's semaphore. The main thread also have its own semaphore, `end_sem`, initialized to 0, that is used to block it while the other threads are processing their data.

When the main thread executes the `correct` function, it wakes the other threads `T[i]` calling the `post( )` of each `start_sems[i]`. The main thread can not leave the function `correct()` until each threads has completed its

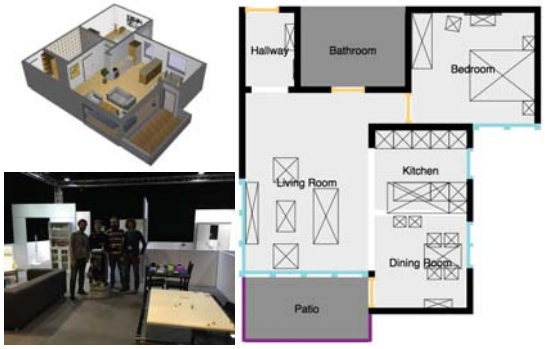


Fig. 5. RoCKiN environment.

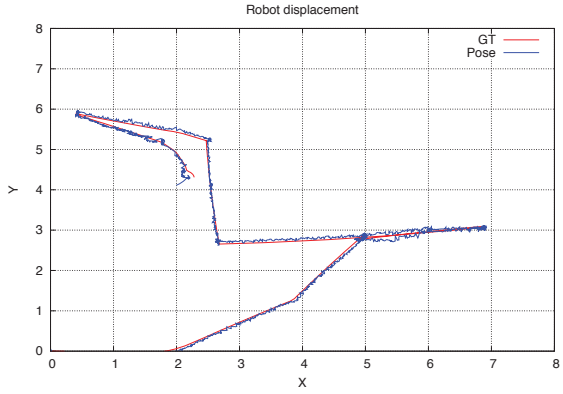


Fig. 6. Route carried out by the robot. Red line is the actual position and the blue one is the position estimated by our algorithm.

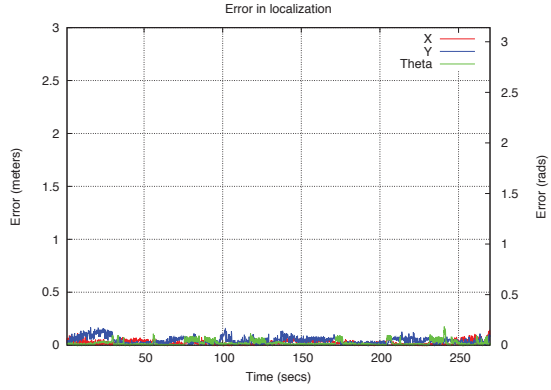


Fig. 7. Error en la localización

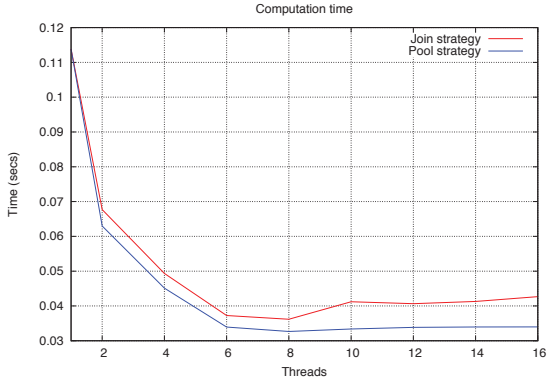


Fig. 8. Average response time both strategies and different number of threads.

work, so call N times the operation `wait()` of `end_sem`. When each thread `T[i]` finishes its work, it calls the `post()` method of `end_sem` to notify to the main thread this finalization. The last thread `T[i]` in finishing makes the main thread wakes up. Then, each thread `T[i]` blocks until next iteration.

V. Experiments

The self location method proposed in this paper has been implemented to execute on board of two different robots: RB-1 and Kobuki. Both robots move by wheels, they have a laser and RGBD (Asus Xtion) sensor. The robot RB-1 has a computer on board Intel i7 with 8 cores and 8Gb of RAM. The Kobuki robot has no onboard computer. In this case, we have equipped with a laptop with similar features to the computer aboard the RB-1.

First we will show the results of an experiment measuring the validity of the localization algorithm. The aim of this paper is not the reliability and robustness of algorithm but the multithread strategies to implement it. Still, we will demonstrate that the implemented algorithm functioning properly. Therefore, we will show an experiment conducted in the environment of the RoCKiN competition, shown in Figure 5. In this experiment, the robot will follow a route from the Hallway of the house to the Dinning Room, and then to the bedroom. In total, the robot will travel 20 meters in not autonomous mode (Figure 6). These accuracy results are independent of the level of multithreading of the self localization algorithm. The results show that this algorithm is robust and accurate.

Once validated the algorithm, we will conduct an experiment designed to determine the improvement obtained using a multithreaded approach. For this goal, we have implemented the algorithm of self location with both the create/join scheme as a pool of threads. Each scheme has been running for 1000 iterations, showing the average response time, shown in Figure 8. In this experiment we have established an amount of 1 to 16 threads in each phase of each iteration of the algorithm of self localization.

In the case of a thread, the average response time per iteration are 113 ms, similar in both experiments. As increasing the number of threads, the difference between the two strategies is increased due to the cost of creation and destruction of threads. The response time increases as we decrease the number of threads. The minimum is around 8 threads, since we are

running this experiment on a CPU with 8 cores. Further thread increase does not improve the response time, and it begins to deteriorate in the case of the create/join strategy, due to the cost of the threads creation/destruction. It is important to note that we managed to reduce the response time from 113ms to 33ms. The let us to run the algorithm at 30 Mhz if necessary, instead of the initial 9 Mhz.

VI. Conclusions y and future work

In this paper we have presented a study of the impact of using multi thread mechanisms to address a problem that requires intensive computation, such as a location with RGBD sensors. Naturally, the use of multi threading in multi core CPUs reduces the response time of the algorithms. We have shown how to improve the response time of the algorithm when using as many threads as cores, becoming counterproductive if this amount is higher.

In addition, we measured the overhead produces by a create/join thread scheme rather than the one based on a thread pool scheme. In addition, we have shown how to synchronize the threads of this pool of threads using semaphores. Using this scheme, threads are waiting a signal to start processing. At finalization, threads signal for continuing the execution of the main thread.

We have described an algorithm based on self location Monte Carlo algorithm, using an RGBD sensor. The use of colored dots in the mapped space, and their use in novel self localization. The disadvantage is that this algorithm requires many computational resources. This paper presents the benefits of addressing this problem throughout a multi threading approach. This work has been tested in the last competition ROCKIN 2015 in Lisbon, shown that it works properly and it is able to keep the robot located with an acceptable response time.

One of the future works is to apply GPU parallelization techniques and compare the results with the results obtained in this article. Still, we continue to believe that GPU programming requires a low-level programming and limits the use of libraries have not a GPU version available.

Acknowledgment

The authors would like to thank the Spanish Ministry of Economy and Competitiveness for the partial support to this work under grant DPI2013-40534-R. The authors would like to thank Francisco J. Ballesteros and Enrique Soriano, the Ubiquitous Systems Laboratory, for their valuable comments on the implementation of synchronization thread pool.

References

[1] Marder-Eppstein E., Berger E., Foote T., and Gerkey, B. and Konolige K., *The Office Marathon: Robust Navigation in an Indoor Office Environment*. International Conference on Robotics and Automation. 2010.

[2] Rusu R.B. and Cousins S., *3D is here: Point Cloud Library (PCL)*, IEEE International Conference on Robotics and Automation (ICRA). 2011.

[3] Holz D., Ruiz-del-Solar J., Sugiura K., and Wachsmuth, S.,*On RoboCup@Home - past, present and future of a scientific competition for service robots*, Lecture Notes in Computer Science, Vol. 8992, pp. 686–697, 2014.

[4] P. U. Lima, D. Nardi, L. Iocchi, G. Kraetzschmar, and M. Matteucci, *RoCKIn@Home: Benchmarking Domestic Robots Through Competitions*, In Robotics Automation Magazine, IEEE, vol. 21, no. 2, pp. 8-12, 2014.

[5] Fox, D., Burgard, W., Dellaert, W., Thrun, S., *Robust Monte Carlo localization for mobile robots*, Artificial Intelligence. Vol. 128, pp. 99–141. 2001.

[6] Brega R., Tomatis, R. and Arrast, K., *The Need for Autonomy and Real-Time in Mobile Robotics: A Case Study of XO/2 and Pygmalion*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000), Takamatsu, Japan, Oct. 30 - Nov. 5, 2000. Journal in Engineering Applications of Artificial Intelligence. Vol. 17, Issue 5, pp. 469–483. August 2004.

[7] Ce Li, Takahiro Watanabe, Zhenyu Wu, Hang Li and Yijie Huangfu. *The Real-Time and Embedded Soccer Robot Control System*, Robot Soccer, Vladan Papi (Ed.), InTech, DOI: 10.5772/7352. 2010.

[8] Gu, J.S. and de Silva, C.W., *Development and implementation of a real-time open-architecture control system for industrial robot systems*,

[9] Simpkins, A., *Real-Time Control in Robotic Systems*, Applications, Control and Programming, Dr. Ashish Dutta (Ed.), InTech, DOI: 10.5772/27883. 2012.

[10] Martín, F., Agüero, C, Plaza, J.M., *A Simple, Efficient, and Scalable Behavior-based Architecture for Robotic Applications*, ROBOT'2015 Second Iberian Robotics Conference.

[11] Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T.., Leibs, J., Wheeler, R., and Ng, Andrew Y., *ROS: an open-source Robot Operating System*. ICRA Workshop on Open Source Software. 2009.

[12] Satish, N., Harris, M., Garland, M., *Designing Efficient Sorting Algorithms for Manycore Gpus. NVIDIA Corporation*, 23rd IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2009, Rome, Italy, May 2009.

[13] Kirk, D., Hwu, W., *Programming Massively Parallel Processors: A Hands-On Approach*, Morgan Kaufmann Ed., 2010.

[14] Parigi, G., Pau, D., Piastra, M., *GPU-based Parallel Implementation of a Growing Self-Organizing Network*, Proceedings of ICINCO 1, pp. 633-643. Jan 2012.

[15] Jang, J., Park, A., Jung, K., *Neural network implementation using CUDA and Open MP*, Proceedings of the International Conference on Digital Image Computing: Techniques and Applications, DICTA 2008, Canberra, ACT, Australia, Dec 2008.

[16] Orts, S., Garcia, J., Serra, J.A., Cazorla, M., *3D Model Reconstruction using Neural Gas Accelerated on GPUs*, Applied Soft Computing, Vol 32, pp. 87-100. July 2015.

[17] Orts, S., Garcia, J., Viejo. D., Cazorla, M., *GPGPU implementation of growing neural gas: Application to 3D scene reconstruction*, Journal of Parallel and Distributed Computing. Oct 2012

[18] Amamra A., Aouf, N., *GPU-based real-time RGBD data filtering*, Journal of Real-Time Image Processing, pp. 1–8, Sept 2014.

[19] Wasza, J., Bauer, S., Hornegger, J., *Real-time Preprocessing for Dense 3-D Range Imaging on the GPU: Defect Interpolation, Bilateral Temporal Averaging and Guided Filtering*, IEEE International Conference on Computer Vision (ICCV), pp 1221–1227. Dec 2011.

[20] Lee, D., *GPU-based real-time RGBD 3D SLAM*, Proceedings of the 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), pp 46–48. 2012.

[21] Rudolph E. Kalman, *A New Approach to Linear Filtering and Prediction Problems*. Transactions of the ASME, Journal of Basic Engineering, Vol. 82, No. Series D, pp. 34–45, (1960).

[22] Lastra, R., Vallejos, P. and Ruiz-del-Solar, J,*Self-Localization and Ball Tracking for the RoboCup 4-Legged League*, Proceeding of the 2nd IEEE Latin American Robotics Symposium LARS 2005, Santiago de Chile, Chile (2005)

[23] Tesli, Luka and A. krjanc, Igor and Klanaear, Gregor, *EKF-Based Localization of a Wheeled Mobile Robot in Structured Environments*, Journal of Intelligent & Robotic Systems, Vol. 62-2, pp. 187–203. 2011.

[24] Hamzah Ahmad and Toru Namerikawa, *Extended Kalman filter-based mobile robot localization with intermittent measurements*, Systems Science & Control Engineering, Vol. 1-1,pp. 113–126, 2013.

[25] Martín F., and Matellán V., and Barrera P., and Cañas, J.M., *Localization of legged robots combining a fuzzy-Markov method and a population of extended Kalman filters*, Robotics and Autonomous Systems, Vol. 55, pp 870–880, 2007.

[26] Dieter Fox, Wolfram Burgard and Sebastian Thrun, *Markov Localization for Mobile Robots in Dynamic Environments*, Journal of Artificial Intelligence Research, Vol. 11, pp. 391–427, 1999.

[27] Sebastian Thrun and Arno Bücken, *Integrating Grid-Based and Topological Maps for Mobile Robot Navigation*, Proceedings of the AAAI

Thirteenth National Conference on Artificial Intelligence, Vol. 2, pp. 944–950. Portland, OG(1996)

[28] Martín F., and Matellán V., and Barrera P., and Cañas, J.M., *Visual Based Localization for a Legged Robot*, Lecture Notes on Computer Science. Vol. LNAI-4020. pp 708–715. (2006).

[29] Martín F., *Visual Localization based on Quadtrees*, ROBOT'2015 Second Iberian Robotics Conference, Volume 418 of the series Advances in Intelligent Systems and Computing pp 599-610. 2015.

[30] Sebastian Thrun, *Particle Filters in Robotics*, Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-02), pp. 511-518. San Francisco, CA (2002).

[31] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schultea and D. Schulz, *Probabilistic Algorithms and the Interactive Museum Tour-Guide Robot Minerva*, International Journal of Robotics Research, Vol. 19, No. 11. (2000), pp. 972-999.

[32] Dieter Fox, Wolfram Burgard, Frank Dellaert, Sebastian Thrun, *Monte Carlo Localization: Efficient Position Estimation for Mobile Robots*, Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI'99), pp. 343-349. Orlando, FL. (1999).

[33] R. Conde, A. Ollero and J.A. Cobano, *Method based on a particle filter for UAV trajectory prediction under uncertainties*. 40th International Symposium of Robotics. Barcelona, Spain (2009).

[34] Nak Yong Ko, Tae Gyun Kim and Sung Woo Noh, *Monte Carlo Localization of Underwater Robot Using Internal and External Information*. 2011 IEEE Asia-Pacific Services Computing Conference, APSCC. 2011, pp. 410–415. Jeju, South Korea (2011).

[35] Scott Lenser and Manuela Veloso, *Sensor Resetting Localization for Poorly Modelled Mobile Robots*, Proceedings of ICRA-2000, the International Conference on Robotics and Automation, pp. 1225-1232. San Francisco, CA (2000).

[36] T. Röfer, T. Laue and D. Thomas, *Particle-filter-based self-localization using landmarks and directed lines*, RoboCup 2005: Robot Soccer World Cup IX, Vol. 4020, pp. 608–615, Lecture Notes in Artificial Intelligence. Springer (2006).

[37] Biswas J., and Veloso, M,. *Localization and Navigation of the CoBots Over Long-term Deployments*. The International Journal of Robotics Research, Vol. 32-14. pp 1679–1694, 2013.

[38] Maier, D.*Real-time navigation in 3D environments based on depth camera data*, 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids), pp. 692–697. 2012.

[39] Eberhardt H., Klumpp V.,, Uwe D. Hanebeck, *Density Trees for Efficient Nonlinear State Estimation*, Proceedings of the 13th International Conference on Information Fusion, Edinburgh, United Kingdom, July, 2010.

[40] Hornung A., Wurm K. M., Bennewitz M., Stachniss C., Burgard W., *OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees*, Autonomous Robots, Vol. 34-2, pp. 189–206, 2013.

An Aerial Autonomous Robot for Complete Coverage Outdoors

L.V. Campo, J.C. Corrales and A. Ledezma

***Abstract*—the aerial robots have become popular as unmanned aerial vehicles for both research and commercial applications. The most research projects develop new functionalities on aerial robots to get autonomous movements and to plan tasks in controlled and indoor environments. However, the applications such as precision agriculture or environmental monitoring require coverage outdoors of areas.**

This paper proposes an air autonomous system for accurate outdoor navigation, with the objective of supporting tasks as risk detection and vegetation monitoring. The system uses AR Drone quadcopter and the Robot Operative System (ROS) to design a control architecture based on data fusion and path planning algorithms to accomplish outdoor coverage tasks. The results of the research describe the accurate of the positioning methodin open environments and the validation of algorithms based on graphs for aerial coverage.

***Index Terms*—Autonomous aerial robot, coverage planning, aerial navigation, outdoors, quadcopter.**

I. Introduction

The development of technology for precision agriculture and environment control have importance for all researches since they support the progress of food production methods and the ecosystems preservation.The technique used for continuous monitoring tasks is known as remote sensing. Remote sensing usually obtains data from remote sensors (cameras, sonar, and laser) to build a map of the state of the objects or the phenomenon. Unmanned aerial vehicles or satellites carry the sensors and perform a sweep of the area to collect the information.

Although the technique is effective, the aerial robots in its role of unmanned aerial vehicles can take the concept beyond, as they are able to fly at low altitudes and reach previously unimagined areas. Based on this, our research aims to build an air system for autonomous navigation of aerial robots in open environments, to collect information on terrain and environment.

The aerial robots are considered as micro aerial vehicles in the classification of the unmanned vehicles [1]. Their weight does not exceed 2 kg, the flying height is between 60 and 300 meters, and the maximum range is five thousand meters. They have the ability to take off vertically, simple mathematical-physical model (Euler model) and have easy maneuverability compared to fixed-wing drones; which depend on the dynamics of air and fuselage design to be in the air. However, the aerial robots have some limitations in outdoors as the limited flight time (maximal 20 minutes), little payload capacity, and poor navigability in rough environments.

In spite of the above, aerial robots have achieved applications for agriculture and environmental control at a time and cost lower than traditional technologies. As an example, a research project deploys anaerial robot for remote sensing of a vineyard in central Italy, with six rotors Mikropkopter called VIPtero[2]. The aerial robot operatesfor 10 minutes and carries a multi-spectral camera to calculate manyvegetation indices. Another research with digital images is the FlightCopter system [3]. The aerial vehicle operates on crops of oats and peas on a farm north of Hesse in Germany. The robot flies on 30 meters, in about 10 minutes. With the system,they calculate the normalized vegetation differential index (NVDI) to relate soil biomass throughout the farm.In the case of environmental control, there are studies such as tracking animals in open fields using radio tags, identified with a robot carries an omnidirectional antenna to get the location of the animals[4]. Others papers for monitoring water and air pollution, using an aerial robot with biosensors to measure the state of water and air on cities[5]. Similarly, a series of research papers usethe aerial robots with chemicals sensors for monitoring mountainous areas, natural parks, and forests.

The most of the projects use software tools that provide companies like Mission Planner [6]or development groups as QgroundControl [7], which are based on a software programmer of zigzag path to coverage tasks. The programmer preloads waypoints without specifying the positioning of robots in open environments or plan for optimal coverage.

According to the previous review, ourresearch seeks to improve the autonomous navigation ofan aerial robot type multirotor, AR Drone[8], for coverage outdoors. The development builds a hybrid control architecture over ROS, in which the *reactive layer* manages the positioning of the robot based on data fusion from the IMU and GPS, and *planning layer* is responsible for coverage path planning for an autonomous flying robot at outdoors.

The content of the document is as follows: section II explores the related papers for contextualizing the proposal, the section III describes the method for developing the project, section IV details the results and respective discussion and section V describes conclusions on the project.

Liseth V. Campo is with the Engineer Telematics Group (GIT) of University of Cauca. E-mail: liscampo@unicauca.edu.co

Juan C. Corrales is with the Engineer Telematics Group (GIT) University of Cauca. E-mail: jcorral@unicauca.edu.co

AgapitoLedezma is with the Control Learning and Systems Optimization Group (CAOS) of University Carlos III of Madrid. E-mail: ledezma@inf.uc3m.es

II. Related Work

The development of the proposal is based on twochallenges: the first is the development of a method of navigation in open environments and the second is a method for the complete coverage planning. Below, the related work with the topics:

A. Navigation Aerial Robots in Open Environments

The navigation methods may or not be geo-referenced using global positioning receivers (GPS). Most researchers without GPS have cameras and use techniques to navigate in unknown environments such as SLAM (Simultaneous Localization and Mapping). An example is a development of navigation systems based on digital cameras facing forward, using algorithms as Klein [9]or FastSLAM[10]. The system processes the images captured with anaerial robot in a ground station, and without external assistance covers an area unreferenced[11]. Others authors use a stereo camera front and a digital camera that looks down. They developthe algorithms to estimate the pose, with the map as a graph and estimating the speed of air from the vertical camera[12].The failures with systems using cameras are the high cost of resources and a radio communication powerful in the implementation of positioningtechniques and accurate mapping to cover large areas.

Theinvestigations with GPS receiver for navigation in outdoor areas, avoidcameras because they involucrate less flight time. These approaches integrate GPS data to the odometry (estimation of the position and orientation) of the robot during the navigation [13]. The most methods studies on a linear transformation of coordinates and data fusion algorithms such as extended Kalman filter (EKF)[14].Theydo not require complex processing and allow a higher probability estimate in positioning the robot. For this reason, researchersprefer the GPS-based positioning for coordination tasks with several aerial robots outdoors [15].

The AR Drone quadcopter has a navigation system based on inertial movement unity (IMU), which is used in outdoors or indoors indistinctly.Additional, it has cameras and GPS receiver but does not implement a method for outdoors navigation with them.The present projecthas selected the GPS-based techniquesusing a data fusion algorithm as the extended Kalman filter. The filteroptimizes the navigation and the flight time to cover large areas without assistance.

B. Planning Autonomous Flight in Open Environments

According to the air navigation system ofa robot, flight planning can be local or global. Local planning takes the perception of the world from sensors carried by the robot. According to those, the robot decides how to proceed further. In the local planning, previously defined points are only the start and end position.

Algorithms for local mapping include obstacle detection and limitstechniques, and based on them, the area is divided. On divisions, the robot performs movements in zigzag on subdivisions [16]. Most techniques to discover critical points in the obstacles by sensors, assume a graph of adjacencies to know the distribution of subdivisions.Other studies do not map graphsbut use artificial intelligence algorithms. One approach is the coverage with a swarm of robots, which actsimilarly as fluids in an area [17]. In the same way, there are works inspired by the behavior of a neural network human or behavior of animals like ants [18], [19].

On the side of the global planning, not only it is necessary the starting and destination, but also a knowledge of the environment to the flight plan (limits, size, obstacles, georeferences). The algorithms usually divide the coverage area in regular forms (squares or triangles) or irregular sections delimited by obstacles in the area.

There are algorithms by trapezoids [20] or no uniform polygons throughout the area [21]. The coverage of each subdivision is a type zigzag route to complete the full coverage area. It is different with algorithms based on regular forms, called cells. Each subdivision indicates a waypoint over the coverage area.Thus, the algorithms do not increase complexity to divide the area, but seeks the best path through each free and not visitedpoint. The algorithms usea graph to map each cell as a node. For example, the algorithm wave front or distance transform assigns a value of zero to the end node and then expands a wave front in its four neighboring nodes [22]. Another algorithm builds a minimum spanning tree, covering most free nodes in the area. Then a path follows a spiral with the starting node and the final node in the same position [23].To close the review, one of the most known for the task of coverage is the D* algorithm. It optimizes the A* algorithm to display a route based on the shortest distance [24].

Our research selected the global planning for applications in open and natural environments. The reason is the high cost of processing involved with the camera-basedtechniques. In addition, the techniquesbased on cameras or lasers do not ensure complete coverage and the study areas are usually small. The global planning is opportune because the coverage in agriculture or natural environmentis for large areas and if the information is incomplete, itcannot build a reliable map.

III. Method

A. AR Drone 2.0 and ROS

AR Drone is the aerial robot selected to research work. The reason is its stability in outdoors, sensor sensitivity, and the software development kit. AR Drone uses inertial motion unit, magnetometers, sonar, barometer and cameras, to calculate the position, orientation and speed, with a high margin of accuracy even in aggressive maneuvers.

The flight control robot can be manual, using a mobile application, or programmed on software,using a GPS called FlightRecorder. In our proposal ROS is the software platform for programming the control of AR Drone, since it is a direct interface between simulated environment and real environment. For AR Drone,ROS developers have created a package called *ardrone_autonomy*, which contains the robot controller, model with sensors, and codes fornavigation. The present project is based on odometry, altitude and GPS receiver messages from *ardrone_autonomy* to build the telemetry of the quad copter and the coverage path by georeferenced waypoints.

B. Architecture

The Fig.1 shows the architecture to develop the proposal. It hasin three layers, considering a hybrid architecture: reaction layer, planning layer and action layer.

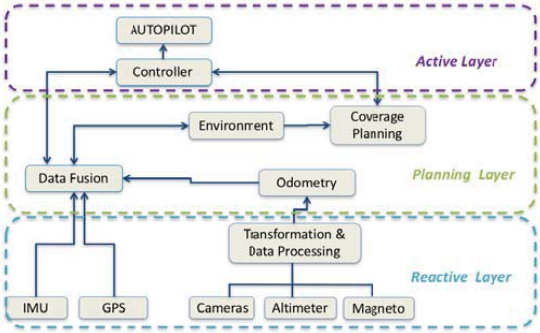


Fig.1ROS architecture project

Reactive layer takes data transmission from sensors AR Drone, specifically IMU and GPS to estimate the position, orientation, and speedby a data fusion module. The results of module changewith the environment. The next layer is the planning; it is responsible for predefining aglobal coverage pathwith the map. When the above layers are processed, then the active layer manages the local and global movements of the robot. In the active layer is the autopilot who manages the actuators (motors) for performing the controller orders.

C. Estimated Positioning

To achieve precision in the navigation of air robot; data fusion techniques are applied as in architecture. Our approaches selected the Extended Kalman Filter to correct and predict the position and velocity of the robot in 3D (6DOF: x, y, z, roll, pitch, yaw) space because it is robust for dynamic systems as open environment [25].

To estimate the positioning with EKF, the project modifies the ROS package called *ethzasl_sensor_fusion.* The filterhas two phases: the first phase combines IMU data and odometryto generate an estimated position. In addition, the second phase takes the estimated position and the GPS receiver data (longitude and latitude), for determinate position and velocity in outdoors.

Phase 1: EKF with IMU, altitude and odometry.

The nonlinear dynamical system is like (1):

$$x_k=f_k(x_{k-1})+w_{k-1} \quad (1)$$

Where $x_k$is the state of the robot (3D pose, 3D orientation, and their velocities) in a time k. f_k:$R^n$is a function of transition nonlinear states and w_{k-1} is process noise, which is distributed. Estimated state is (2):

$$y_k=h_k(x_k)+ v_k \quad (2)$$

Where y_k is the measure of the state of the robot at a time k,h_k:R^m is a model nonlinear, which maps the state within the space of measurements. In addition, v_k is the measurement noise distributed.

Prediction Step: It projects the estimated state and the error covariance in the elapsed time. Equations (3) and (4) represent this step.

$$\hat{x}_k = f_k(x_{k-1}) \quad (3)$$

$$\hat{P}_k = F_k P_{k-1} F_k^T + Q_k \quad (4)$$

The error covariance estimated is P_k, which is projected by F_k, the Jacobean f_k. Then it is disturbed by the noise covariance process, Q_k.

Correction Step: Kalman gain, K_K, is calculated using a matrix of observations, and his covariance. The gain actualizes the covariance matrix and the state vector.

$$K_K = \hat{P}_k H_k^T [H_k \hat{P}_k H_k^T + R_k]^{-1} \quad (5)$$

$$x_k = \hat{x}_k + K_K[y_k - H_k \hat{x}_k] \quad (6)$$

$$P_K = [I - K_k H_k]\hat{P}_k[I - K_k H_k]^T + K_k R_k K_k^T \quad (7)$$

Correction step assumes that the measurements of the sensors produce state variables previously estimated. For this reason, H_k is an identical matrix.

Phase 2: EKF with GPS and estimated odometry from phase 1.

The first step in the phase is the transformation from the reference frame to GPS UTM coordinates. The transformation matrix depends on UTM position and orientation from the first position recorded by the GPS. For EKF of phase 2, x_k is the state of the robot (estimated odometry, longitude, and latitude) in a time k, the rest of steps are similar to phase 1

D. Coverage Planning

To plan a coverage in open and natural environments is required the parameters as the area, as limits, obstacles, and dimensions. According to the architecture, the planning process begins with the characterization coverage area and their respective georeferencing, this process is executed by QG is tool. With the detection ofdimensions and obstacles, a map in grayscale is generated. The tool is the OpenCV library, which is on Qt plugin ROS.

The next step is to decompose the area with uniform grids, which are represented as an adjacent graph including obstacles and free nodes. Then, a coverage strategy takes the map in graphs and find an optimal path. Selected algorithms to determine the optimal path are the heuristic algorithm based on Dijkstra[26], the distance transform coverage[27] and the coverage in spirals[28]. Each implements a backtracking technique to ensure the visit all free nodes. The backtracking strategy takes the list of visited nodes stored in the route and revisitfor searching free neighbors.

The description of each algorithm below:

Heuristics Coverage:

Set Start Node to Current Node
Set all Nodes to Not Visited
Loop
Find unvisited Neighboring node with smallest distance

If No Neighbor Node found then
Mark as Visited and start Backtracking
If with Backtracking No Neighbor Node found then
Stop at Goal
Else
Mark as Current Node
IfNeighboring Node distance>= Current Node distance then
Mark as Visited and Stop at Goal
Set Current Node to Neighboring Node
Loop End

Distance Transform Coverage:

Set Goal Node to Current Node
Set all Nodes to Not Visited
Loop
Find unvisited Neighboring node with highest DT value
If No Neighbor Node found then
Mark as Visited and Start Backtracking
If with Backtracking No Neighbor Node found then
Stop at Start
Else
Mark as Current Node
If Neighboring Node DT value<= Current Node DT value then
Mark as Visited and Stop at Start
Set Current Node to Neighboring Node
Loop End

Spiral Coverage:

Set Start Node to Current Node
Set all Nodes to Not Visited
Loop
Find unvisited Neighboring Front Node.
If No Neighbour Node found, then
Find unvisited Neighboring Lateral Node and change orientation robot.
If No Neighbour Node found then
Mark as Visited and start Backtracking
If with Backtracking No Neighbor Node found then
Stop at Goal
Else
Mark as Current Node
Set Current Node to Neighboring Node
Loop End

Implementation of global planning module includes topics, services and messages available by the ROS *ardrone_autonomy* package, as *ardrone/setgpstarget* and *ardrone/setautoflight*. The visualization and calculation of paths by algorithms is performed on a Qt plugin ROS.

IV. Results and Discussion

The place for testing is in the city of Popayan, Colombia(2.459167N, -76.600278W), which is a tropical area, mountainous, with an average temperature of 19.0 °C, 78% of average humidity, 1760 meters above the sea and wind speed about 5.4 kilometers per hour. Flights take place during the day and in uncongested areas by people or animals.

A loop path is an experiment to evaluate the positioning accuracy estimated by the extended Kalman filter implemented in the reactive layer. The graphics of the Fig.2, Fig.3 and Fig.4, show the tests. The closed path is initially programmed with straight roads that follow a rectangle of 3.5 meters by 6.8 meters. The Fig.2 shows the real pathof the aerial robot, only with odometry from manufacturers. The path shows an unstable behavior with significant drifts during the flight. Some positions of the path have duplicate values and scattered by the sampling frequency. The Fig.3 shows the 2D positioning of the rectangular trajectory with the first EKF implemented. The trajectory shows a significant correction of the basic odometry, with more stable sections and smoother turns. The final phase with EKF is presented in the Fig.4. In the graphic,the 3D closed path is the result of a combination of filtered odometry but in UTM space con altitude from the barometer.

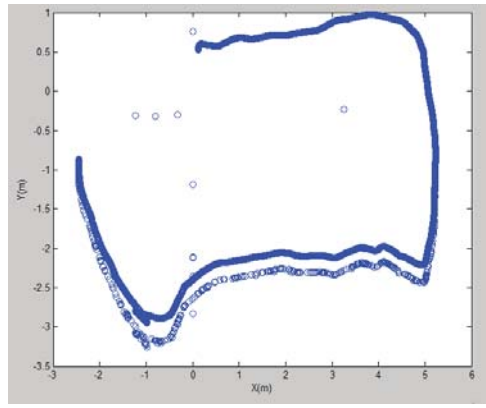


Fig.2 Basic Odometry

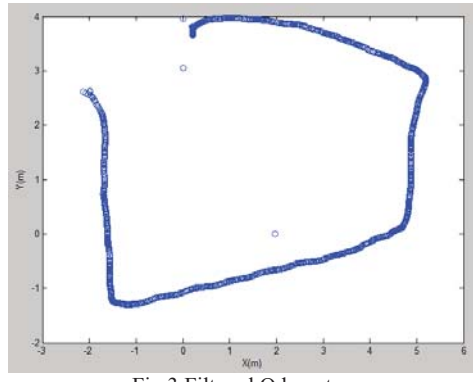


Fig.3 Filtered Odometry

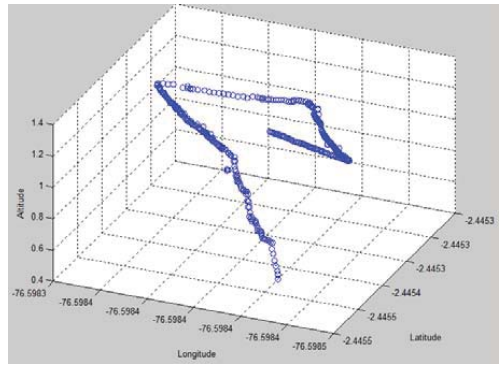


Fig.4 GPS Odometry

Table I present the absolute error for each process with basic odometry, filtered odometry,and combining GPS and odometry filtered.

TABLE I
ACCURACY OF ESTIMATED POSITION IN OUTDOORS ENVIRONMENT

Odometry	Average position error (x, y) m
Basic	±10, ±11
Filtered	±7, ±9
GPS filtered	±3, ±4

According to the results of the figures and Table I is visible an improvement in basic odometry close to 25% with the Kalman filter with IMU, altitude and odometry; and about 40% with the GPS odometry implemented. The main drawbacks to implementing of the filter is the ability of the receiver and dependence on the sensors with relief conditions for calculating altitude. The above results have amargin optimal margin compared to the dimensions of the area and the inaccuracy of the basic odometry for air navigation in open environments.

The following results show the layer planning of architecture. In this section, the three evaluated algorithms conclude with a mission that includes optimal coverage path to manage energy resources, visit all cells and maintain the functionality of the air robot. TheFig.5, Fig. 6, Fig. 7, Fig. 8 and Fig.9 represent the programmed path algorithms. In the figures, the red points are start positions and the green points are the goal positions. Each graphic presents a black route, which is the result of applying the process described in Coverage Planning section, and a gray route to the real path.The squarecell size (the subdivision of the area) selected for the three algorithms is 5.2 meterswith a flying height set at 80 meters.

The next figures exhibit thefirst experiments of the coverage algorithmsin ROS platform. They launch a series of waypoints evenly distributed, which are transformed to UTM coordinates. In the coverage close to buildings, the robot undergoes major drifts; it may be interference communication channels and local winds. Inlots, the robot attempt is lower.

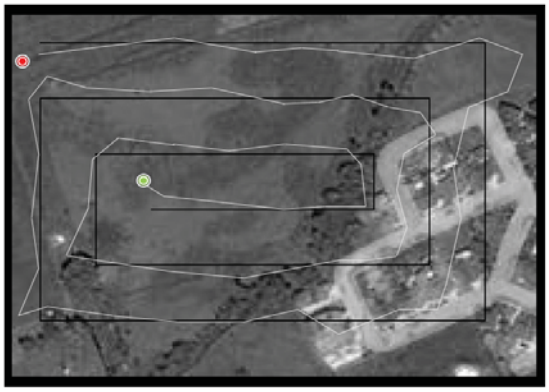


Fig.5 Spiral algorithm on area without obstacles

The Fig.5 and Fig.9 describes a defined path from Spiral Coverage, the real path tends to follow the spiral, however when obstacles are added, the turns are very pronounced, with increased power consumption. The algorithm has the disadvantage that goal position will always be far from the end position, which is not appropriate in extensive areas.

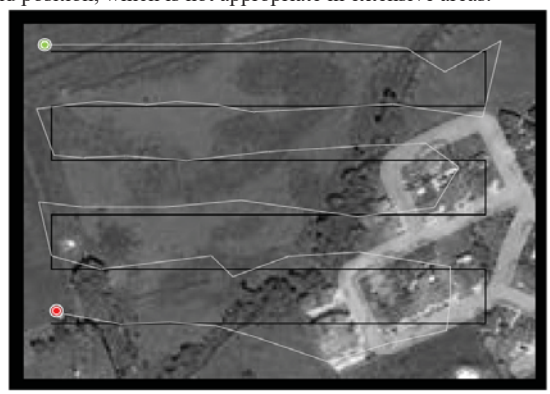


Fig.6 Distance transformed algorithm on area without obstacles

The Fig. 6 is an example of path generated from heuristic and distance transform algorithm. The figure shows a zigzag path type, and except for the built-up area, the turns and straight paths of robot follow the benchmarks.

The Fig.7 and Fig. 8 shows coverage paths from heuristic and distance transform algorithm for areas with obstacles. In they, there is an average of 10 nodes revisited, which are the result of the process of backtracking. Although the route is accurate for both algorithms, the main differences are dependent on the intensity and direction of the wind.

Fig.7Heuristics algorithm on area with simulated obstacles

Fig.8 Distance transformed algorithm on area with simulated obstacles

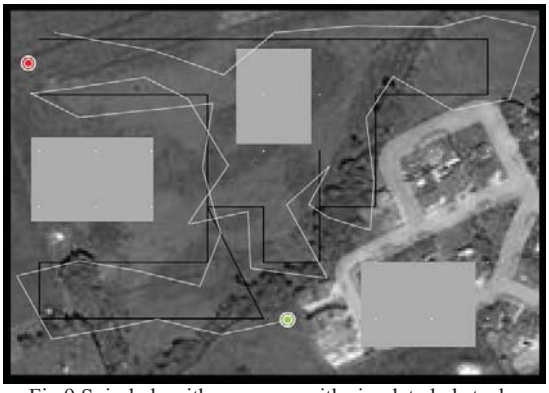


Fig.9 Spiral algorithm on area with simulated obstacles

The evaluation metrics for algorithms coverage planning are set out in Table II. The results show that the heuristic algorithm has greater coverage, but the algorithm is set up with a neighborly eight, possibly with access to the most of the free nodes. Conversely, the algorithm spiral has the lowest percentage of coverage; however, it has fewer turns and better manage energy resources with less flight time. The results of Table II are collecting several tests at different locations; however, in each test the algorithms have the same position of free nodes.

The above figures only show one of the areas studied by this project, and in this particular case the spectrum of the buildings is matched with the simulated obstacles for calculating waypoints do not consider that area.

TABLE II
COMPARISON OF THREE ALGORITHMS IMPLEMENTED

Algorithm	Free Nodes Visited (%)	Flight Time (min)
Heuristic	90	10,2
Distance Transform	85	9,8
Spiral	70	9,4

As mentioned above, the performance of the robot in performing the programmed route is variable. This is due to factors such as the wind. Depending on the intensity and direction of the wind, the aerial robot can take less or more time in a path in the same area. As complementaryresearch to power management with coverage algorithms, we performed tests of the effects of the wind direction on the coverage paths. Table III shows the results for a path generated from spiral coverage algorithm.

TABLE III
WIND DIRECTION AND FLIGHT TIME

Robot Orientation to the wind	Flight Time (min)
Against	11,2
In sense	>12 (with drift)
Cross	9,5

Table II is an experimental precedent, which states that paths with the robot orientation initial cross wind, it ispossible to maintain the power of the robot and complete tasks without fear of bad energy expenditure for flightsin the sense of the wind, or loss the robot.

V. Conclusion

The development of the research project takes origin in the need to create a solution with MAV low-cost technology for applications in natural and open environments, such as precision agriculture and environmental monitoring. To achieve the aim, a system based on AR Drone, a low cost micro aerial vehicleis built. The method selected integrates and optimizethe sensor information, for completing accurate coverage paths. The proposal evaluates three algorithms based on square grids decomposition. However, the selection of a unique algorithm for coverageis not declared, since it requires testing of different scenarios.

The results present the advantages and limitations of ourautonomous systemfor aerial coverage outdoors. In addition to precision movements programmed, the system creates strategies for energy management with coveragealgorithms from roboticsand considerations of the effect of the wind on outdoor navigation. These features make the difference with software tools found for land planning since they allow the construction of specific aircraft systems for environments such as agriculture and environmental control, where there arephysical phenomena changing and the accuracy of the information collected is important.

The further project seeks validation of architecture designed with other tests, to ensure portability and efficiency of the robot in environments such as crops or wooded areas. In addition, the authors work in the implementation of the proposed architecture on an open sourcerobot, with which it is expected to have more flight time and validate algorithms navigation, since the design of AR Drone limit the project to a basic prototype.

Acknowledgment

The authors thank to the University Carlos III of Madrid and projects TRA2011-29454-C03-03 and TRA2015-63708-R funded by the Ministry of Economy and Competitiveness of Spain, for the support to develop. To Vicerrectoria de Investigaciones of the University of Cauca and to WPA2-AgroCloud project of RICCLISA funded by Colciencias Colombia, for the resources and equipment,and Colciencias Colombia for support to Liseth V. Campo with a research grant.

References

[1] OACI, «Sistemas de aeronaves no tripuladas (UAS),» Quebec, 2011.

[2] J. Primicerio, S. F. Di Gennaro, E. Fiorillo, L. Genesio, E. Lugato, A. Matese and F. P. Vaccari, «A flexible unmanned aerial vehicle for precision agriculture,» *Precision Agriculture,* vol. 13, nº 4, pp. 517-523, 2012.

[3] R. Jannoura, K. Brinkmann, D. Uteau, C. Bruns and R. G. Joergensen, «Monitoring of crop biomass using true colour aerial photographs taken from a remote controlled hexacopter,» *Biosystems Engineering,* vol. 129, pp. 341-351, 2015.

[4] A. Posch and S. Sukkarieh, «UAV based search for a radio tagged animal using particle filters,» of*Australasian Conference on Robotics and Automation (ACRA)*, Sidney, Citeseer, 2009, pp. 2-4.

[5] Y. Lu, D. Macias, Z. S. Dean, N. R. Kreger and P. K. Wong, «A UAV-Mounted Whole Cell Biosensor System for Environmental Monitoring Applications,» *IEEE Transactions on NanoBioscience ,* pp. 811-817, 2015.

[6] 3D Robotics, 2016. [Online]. Available: https://3dr.com/kb/mission-planning-mission-planner/. [Last accessed: March 27, 2016].

[7] Pixhawk Project at ETH Zurich , 2010. [Online]. Available: http://qgroundcontrol.org/. Last accessed: March 27, 2016].

[8] Parrot, «Parrot Ar Drone,» 2015. [Online]. Available: http://ardrone2.parrot.com/. [Last accessed: March 30, 2016].

[9] M. Blosch, S. Weiss, D. Scaramuzza and R. Siegwart, «Vision based MAV navigation in unknown and unstructured environments,» of*Robotics and automation (ICRA)*, Anchorage, IEEE, 2010, pp. 21-28.

[10] J. Yang, D. Rao, S. Chung and S. Hutchinson, «Monocular vision based navigation in GPS-denied riverine environments,» of*Proceedings of the AIAA Infotech@ Aerospace Conference*, St. Louis, AIAA , 2011, pp. 1-12.

[11] K. Bipin, V. Duggal and K. Madhava Krishna, «Autonomous navigation of generic monocular quadcopter in natural environment,» of*Robotics and Automation (ICRA)*, IEEE, Ed., Seattle,, IEEE, 2015, pp. 1063-1070.

[12] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen and M. Pollefeys, «Vision-based autonomous mapping and exploration using a quadrotor MAV,» of*Intelligent Robots and Systems (IROS)*, IEEE, Ed., Vilamoura, IEEE, 2012, pp. 4557-4564.

[13] V. H. Andaluz, F. A. Chicaiza, A. Meythaler, D. R. Rivas and C. P. Chuchico, «Construction of a quadcopter for autonomous and teleoperated navigation,» of*Design of Circuits and Integrated Systems (DCIS)*, IEEE, Ed., Estoril, IEEE, 2015, pp. 1-6.

[14] V. Duggal, M. Sukhwani, K. Bipin, G. S. Reddy and K. M. Krishna, «Plantation Monitoring and Yield Estimation using Autonomous Quadcopter for Precision Agriculture,» of*ICRA*, IEEE, 2015, pp. 1-7.

[15] G. Vásárhelyi, C. Virágh, G. Somorjai, N. Tarcai, T. Szörényi, T. Nepusz and T. Vicsek, «Outdoor flocking and formation flight with autonomous aerial robots,» of*Intelligent Robots and Systems (IROS 2014)*, IEEE, Ed., Chicago, IEEE, 2014, pp. 3866-3873.

[16] E. Garcia and P. G. De Santos, «Mobile-robot navigation with complete coverage of unstructured environments,» *Robotics and Autonomous Systems,* vol. 46, nº 4, pp. 195-204, 2004.

[17] M. A. Batalin and G. S. Sukhatme, «Spreading out: A local approach to multi-robot coverage,» of*International Symposium on Distributed Autonomous Robotic Systems (DARS)*, Tokyo, Springer, 2002, pp. 373-382.

[18] C. Luo and S. X. Yang, «A bioinspired neural network for real-time concurrent map building and complete coverage robot navigation in unknown environments,» *Neural Networks, IEEE Transactions on,* vol. 19, nº 7, pp. 1279-1298, July 2008.

[19] Y. Gabriely and E. Rimon, «Spanning-tree based coverage of continuous areas by a mobile robot,» *Annals of Mathematics and Artificial Intelligence,* vol. 31, nº 4, pp. 77-98, 2001.

[20] T. Oksanen and A. Visala, «overage path planning algorithms for agricultural field,» *Journal of Field Robotics 26,* p. 651–668, 2009.

[21] H. Choset and P. Pignon, «Coverage path planning: the boustrophedon cellular decomposition,» *Proceedings of International Conference on Field and Service Robotics,* 1997.

[22] A. Zelinsky, R. Jarvis, J. Byrne and S. Yuta, «Planning paths of complete coverage of an unstructured environment by a mobile robot,» *Proceedings of International Conference on Advanced Robotics,* p. 533–538, 1993.

[23] Y. Gabriely and E. Rimon, «Spiral-stc: an on-line coverage algorithm of grid,» *Proc. IEEE Int. Conf. Robotics and Automation,* p. 954–960, 2002.

[24] M. Dakulovic, S. Horvatic and I. Petrovic, «Complete Coverage D* Algorithm for Path».

[25] R. Negenborn, «Kalman Filter Extensions,» of*Robot localization and Kalman filters*, Utrecht , Utrecht University, 2003.

[26] E. W. Dijkstra, «A note on two problems in connexion with graphs,» *Numerische mathematik,* vol. 1, nº 1, pp. 269-271, 1959.

[27] J. Valente, A. Barrientos and J. Del Cerro, «Coverage path planning to survey large outdoor areas with aerial robots: A comprehensive analysis,» *Introduction to Modern Robotics I,* 2011.

[28] E. Gonzalez, O. Alvarez, Y. Diaz, C. Parra and C. Bustacara, «BSA: a complete coverage algorithm,» of*Robotics and Automation, 2005. ICRA 2005.*, IEEE, 2005, pp. 2040-2044.

[29] K. Schmid, T. Tomic, F. Ruess, H. Hirschmuller and M. Suppa, «Stereo vision based indoor/outdoor navigation for flying robots,» of*Intelligent Robots and Systems (IROS)*, IEEE, Ed., Tokyo, IEEE, 2013, pp. 3955-3962.

Generation and control of locomotion patterns for biped robots by using central pattern generators

Julián Cristiano, Domènec Puig and Miguel Angel García

***Abstract*—This paper presents an efficient closed-loop locomotion control system for biped robots that operates in the joint space. The robot's joints are directly driven through control signals generated by a central pattern generator (CPG) network. A genetic algorithm is applied in order to find out an optimal combination of internal parameters of the CPG given a desired walking speed in straight line. Feedback signals generated by the robot's inertial and force sensors are directly fed into the CPG in order to automatically adjust the locomotion pattern over uneven terrain and to deal with external perturbations in real time. Omnidirectional motion is achieved by controlling the pelvis motion. The performance of the proposed control system has been assessed through simulation experiments on a NAO humanoid robot.**

***Index Terms*—Adaptive control, biologically inspired control, central pattern generators, CPGs, Matsuoka's oscillator.**

I. Introduction

During the last decades, biped locomotion has basically been tackled as an inverse kinematic problem, aiming to generate a dynamic locomotion pattern by calculating trajectories for the robot arms and legs in the robot's Cartesian space under the constraint that the robot walks while keeping its dynamical balance. This is a valid solution widely used in humanoid robots. However, animals and humans do not need to compute any Cartesian space trajectory nor require precise models of their body or the environment, since their complex nervous system is able to automatically learn motion patterns by controlling extensor and flexor movements and then adapt them according to internal changes or external environmental conditions.

Many studies show the presence of specialized networks of neurons able to generate the rhythmic patterns in animals and humans, such as walking, running and swimming. These networks are called *central pattern generators* (CPGs). The term central indicates that sensory feedback is not necessary for the generation of rhythmic signals. CPGs are modelled as networks of neurons capable of generating stable and periodic signals controlled through a set of constant parameters. In the case of vertebrates, these networks are located in the central nervous system within the spinal cord. The output signals from these CPGs are sent to the muscles through the peripheral nervous system. High-level commands are sent to the different CPGs by the brain through the spinal cord. These commands do not generate the periodic signal by themselves, since the oscillation is autonomously generated within the CPG in the spinal cord.

Julián Cristiano and Domènec Puig are with Rovira i Virgili University. E-mail: julian11495@yahoo.com
Miguel Angel García is with Autonomous University of Madrid.

Currently, many works about CPG-based locomotion control of legged robots and other types of robots have been proposed ([1], [2]). The CPG networks have mainly been used for controlling the robot gait in the robot's task-space or in the robot's joint-space. Biped locomotion is a complex problem since it involves the inherent instability of humanoid robots. Therefore, it is important to develop an appropriate control scheme capable of generating stable motions, and CPGs have shown to be an appropriate model for solving this problem adequately. Thus, the robotics community has shown an increasing interest in locomotor central pattern generators since these networks are able to generate complex high-dimensional signals for controlling coordinated periodic movements with simple input signals.

Within the task-space approach, a CPG network that generates the stepping and propulsive motion for locomotion control of a biped robot was proposed in [3]. The feedback pathways for propulsive motion were obtained through a gradient method, by using the pelvis angular velocity in the sagittal and coronal planes as inputs in order to generate a feedback signal that controls the trajectory of the legs in the walking direction. However, only results on flat terrain were reported. Alternatively, a control system that generates the motion of a biped robot in the task-space by using nonlinear oscillators was presented in [4]. These movements are modulated through the signals provided by touch sensors. Later in [5], the same authors extended their previous work in order to control the turning behaviour of the biped robot. In [6], a method was proposed to generate a walking pattern and stabilize it based on coupled oscillators without real time computation of the zero moment point (ZMP). In [7], a CPG is utilized to describe and modulate the trajectory of the robot's center of gravity and, as a result, the trajectories of its limbs in the workspace. Experiments show that the robot is able to walk on both flat and inclined terrain with slopes of +/-10 degrees. In [8], a pattern generator system for biped locomotion based on CPG networks is proposed. The system operates in the task-space. The authors claim that the robot can walk on flat and inclined terrain with slopes of +/- 7 degrees.

Regarding the joint-space approach, a CPG implemented with coupled nonlinear oscillators was proposed in [9] in order to control the biped locomotion of a humanoid robot. The system is able to learn an arbitrary signal in a supervised framework. It can modulate some parameters and allows the introduction of feedback signals provided by the robot's sensors. However, well defined locomotion patterns must be

defined in advance.

In [10], the signals for the robot's joints are generated by using coupled oscillator models based on sensory information about the location of the center of pressure and its velocity. However, results on flat terrain were only reported.

In turn, a feedback mechanism for phase regulation by using load sensory information was proposed in [11]. The signals for the motors are specified in the joint-space through mathematical formulations that define the angular displacement, with the parameters that characterize the system's behaviour being hand-tuned. Later [12], the same authors proposed a multi-objective staged evolutionary algorithm in order to find out the parameters that characterize the open-loop behaviour of the system. However, due to the reduced number of individuals used by the genetic algorithm and that a hand-tuned gait was included as an individual in a random initial population, thus biasing the final convergence, there is no guarantee that the algorithm ends up exploring the whole search space and, as a result, that it finds out all feasible solutions. In addition, the control system was only tested on flat and sloped terrain with a maximum ascending slope of 4 degrees and a maximum descending slope of 2.5 degrees.

In [13], a control scheme for qualitative adaptive reward learning with success failure maps applied to humanoid robot walking was proposed. However, that technique does not ensure a stable interaction with the floor, since the robot tends to drag its feet when walking, which is likely to lead to falls on uneven terrain. The authors present results with the NAO walking on slopes of +/-10 degrees.

Table I summarizes the most representative control schemes for locomotion control of biped robots that have successfully been tested on small-size humanoid robots. The proposed technique belongs to the joint-space category, as the CPG output signals directly drive the angular position of the robot's joints, and yields results comparable to those reported in [7] in terms of walking speeds and types of terrain, although the latter is a task-space approach that requires solving the inverse kinematics, thus limiting the response time to unexpected events, which may end up compromising the robot's safety. The proposed CPG guarantees that the open-loop control system generates a locomotion pattern that correctly interacts with the floor. In addition, it allows a straightforward modulation of the locomotion patterns through sensory feedback in order to cope with uneven terrain and transitions between different types of ground, and eases the introduction of additional feedback controllers to deal with external perturbations.

This paper is organized as follows. Section II describes the control system. Experimental results are presented and discussed in Section III. Finally, conclusions and future work are given in Section IV.

II. CPG-BASED CONTROL SYSTEM

This section describes a CPG network and the associated methodology to automatically estimate the configuration parameters of the system in order to generate well-characterized locomotion patterns in straight line. The locomotion pattern is automatically obtained with a genetic algorithm by evaluating the locomotion performance with different combinations of parameters through dynamics simulations [14]. Some feedback strategies are presented in order to continuously walk on various types of terrains and to deal with external perturbations.

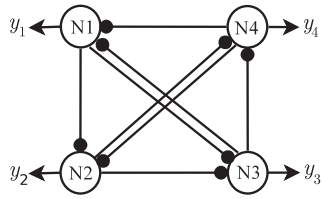


Fig. 1. CPG network of 4 neurons as proposed in [15].

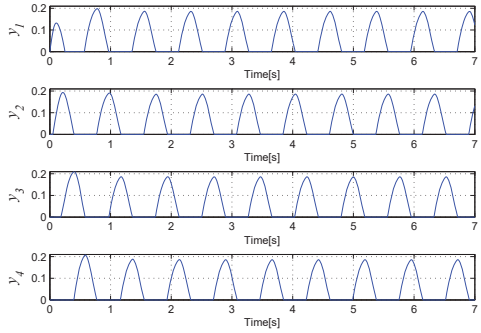


Fig. 2. Output signals of the 4-neuron CPG network.

A. CPG network and neuron's model

The CPG utilized in this work is based on a network of 4 interconnected neurons with mutual inhibition previously proposed by Matsuoka [15]. The topology of that CPG is shown in Fig. 1. That network has been chosen as it generates oscillatory output signals in phase, anti-phase and with phase differences of $\frac{\pi}{2}$ and $\frac{3\pi}{2}$ radians. These phase differences are sufficient to control the robot's movement directly in the joint space, as shown in [10]. In the present work, however, that network directly drives the robot's joints instead of the phase oscillators used in [10]. The interconnection weights between the neurons of that CPG, which have been set according to [3], are shown in Table II. Figure 2 shows the output signal of each neuron of the CPG network.

The CPG's neurons are defined according to the well-known Matsuoka's neuron model:

$$\tau \dot{u}_i = -u_i - \sum_{j=1}^{N} w_{ij} y_j - \beta v_i + u_e + f_i \quad (1)$$

$$\tau' \dot{v}_i = -v_i + y_i \quad (2)$$

$$y_i = max(0, u_i), \quad i = 1, ..., N$$

The external input u_e affects the amplitude of the neuron's output signal. The frequency of the output signal is determined

TABLE I
CPG-BASED LOCOMOTION CONTROL SYSTEMS TESTED ON SMALL SIZE HUMANOID ROBOTS

Authors	*Pattern generator*	*Feedback strategies*	*Tested terrain*	*Employed robot*
S. Aoi et al. [4]	Coupled oscillators (task space)	Phase resetting through the impact instant	Flat terrain	HOAP-1
J. Morimoto et al. [10]	Coupled oscillators (joint space)	COM used for modulation of phase resetting	Flat terrain Maximum obstacle height of 3.5mm	Qrio
V. Matos et al. [11]	Coupled oscillators (joint space)	Phase regulation	Flat terrain Maximum ascending slope of 4 degrees Maximum descending slope of 2.5 degrees	Darwin
C. Liu et al. [7]	CPG-task space control (task space)	Modulation of the COM trajectory	Flat terrain Maximum ascending slope of 10 degrees Maximum descending slope of 10 degrees	Nao
J. Nassour et al. [13]	Neurobiological-inspired learning algorithm (joint space)	Inertial sensor used to adjust the center of oscillation of ankle joints	Flat terrain Maximum ascending slope of 10 degrees Maximum descending slope of 10 degrees	Nao
K. Song [8]	CPG-task space control (task space)	Posture controller	Flat terrain Maximum ascending slope of 7 degrees Maximum descending slope of 7 degrees	Nao
Proposed approach J. Cristiano et al.	CPG-joint space control (joint space)	Posture controller Stepping controller Stride length controller Phase resetting controller	Flat terrain Maximum ascending slope of 10 degrees Maximum descending slope of 10 degrees	Nao

TABLE II
CPG'S INTERCONNECTION WEIGHTS

$w_{1,1}$	0.0	$w_{1,2}$	0.0	$w_{1,3}$	2	$w_{1,4}$	0.5
$w_{2,1}$	0.5	$w_{2,2}$	0.0	$w_{2,3}$	0.0	$w_{2,4}$	2
$w_{3,1}$	2	$w_{3,2}$	0.5	$w_{3,3}$	0.0	$w_{3,4}$	0.0
$w_{4,1}$	0.0	$w_{4,2}$	2	$w_{4,3}$	0.5	$w_{4,4}$	0.0

TABLE III
INTERNAL PARAMETERS FOR EACH NEURON

Parameter	*Value*	*Parameter*	*Value*
τ_o	0.2800	u_e	0.4111
τ'_o	0.4977	f_i	0
β	2.5000		

by the time constants τ and τ'. The set of parameters must satisfy some requirements in order to yield stable oscillations ([15], [16]). Term f_i is a feedback variable that can be used to control the output amplitude and to synchronize the output signals with a periodic input signal. Parameter w_{ij} represents the bidirectional interconnection weight between two neurons. Those inteconnection weights determine the phase difference among the output signals generated by the CPG. When a network of neurons is set, they all oscillate together according to their internal parameters and the network interconnections, converging to specific patterns and limit cycles. Variable N represents the number of neurons that constitute the CPG ($N = 4$ in this work).

Parameter K_f has been introduced as proposed in [17] in order to modulate the frequency of the output signal. The time constants in (1) and (2) are thus reformulated as:

$$\begin{aligned} \tau &= \tau_o K_f \\ \tau' &= \tau'_o K_f, \end{aligned}$$

where τ_o and τ'_o are the original time constants.

The internal parameters that determine the behaviour of each neuron are summarized in table III. The CPG generates stable oscillations provided those parameters satisfy some requirements ([15], [16]).

In this work, the proposed control system has been tested on the NAO platform [18], which is a small size humanoid robot with 21 degrees of freedom, 56 cm tall and weighting 4.8 Kg. Notwithstanding, the same control system can easily be adapted to other humanoid robots with a similar kinematic structure.

The locomotion control of humanoid robots in the joint space must control the pitch and roll motion of the different robot's joints from the output signals generated by the CPG. In this work, the controllers proposed in [10] have been used to determine the angle in radians of the following joints of the NAO robot:

$$\begin{aligned} RHipPitch &= bias1 + a(-\xi(y_1 - y_3) + (y_2 - y_4)) \\ LHipPitch &= bias1 + a(\xi(y_1 - y_3) - (y_2 - y_4)) \\ LKneePitch &= bias2 + b(y_2 - y_4) \\ RKneePitch &= bias2 + b(y_4 - y_2) \\ RAnklePitch &= bias3 + c(\xi(y_1 - y_3) + (y_2 - y_4)) \\ LAnklePitch &= bias3 + c(-\xi(y_1 - y_3) - (y_2 - y_4)) \\ RHipRoll &= d(y_2 - y_4) \\ LHipRoll &= d(y_2 - y_4) \\ LAnkleRoll &= e(y_4 - y_2) \\ RAnkleRoll &= e(y_4 - y_2) \\ RShouldPitch &= bias4 + f(y_1 - y_3) \\ LShouldPitch &= bias4 - f(y_1 - y_3) \end{aligned} \quad (3)$$

Those controllers depend on 10 internal parameters: 4 biases ($bias1, ..., bias4$) and 6 gains (a, b, c, d, e, f). Parameter ξ controls the stride length. Both the latter and the locomotion frequency, which is controlled through the value of K_f, determine the robot's walking velocity. By taking into account the relationship between locomotion frequency and stride length in the human gait, which has been studied in [19], table IV shows the pairs (K_f, ξ) that have experimentally been chosen in this work for 5 reference velocities of the NAO robot. The

remaining joints have experimentally been set to the constant values shown in table V in order to yield a stable upright position.

TABLE IV
PARAMETERS RELATED TO LOCOMOTION FREQUENCY AND STRIDE LENGTH FOR SOME VELOCITIES IN ACCORDANCE WITH HUMAN GAIT

Velocity [cm/s]	1	3	5	7	9
K_f	1.0010	0.8546	0.7410	0.6583	0.6046
ξ	1.1318	1.3445	1.5760	1.8262	2.0950

B. Estimation of CPG parameters through evolutionary computation

The genetic algorithm (GA) proposed in [20] has been applied in order to estimate the best combination of all internal parameters of the locomotion controllers specified in the previous section. In the present work, the input parameter of the GA is the required velocity in straight line. The chromosome structure is composed of 10 traits associated with the respective gains and biases that constitute the internal parameters of the locomotion controllers: (a, b, c, d, e, f) and ($bias1$, $bias2$, $bias3$, $bias4$). Table VI shows the allowed intervals for those parameters, which constitute the GA's search space. Those limits were experimentally delimited by taking into account the range of variation of the optimum solutions found by the GA after an extensive set of executions.

The GA's fitness function evaluates each individual of the current population at the end of a constant simulation period (30 seconds in this work) in which the robot is allowed to walk using the Webots real-time simulator. In particular, the fitness function that is maximized in order to sort out the individuals evaluated by the GA in each generation is the average of four terms. The first term applies a Gaussian function to the difference between the required velocity in straight line and the velocity reached at the end of the simulation period for the evaluated individual. The second term applies a Gaussian function to the difference between the distance that the robot should travel in straight line at the required velocity at the end of the simulation period and the final distance traveled by the evaluated individual. That term is maximized if the robot follows a straight path during the whole simulation period. The third term corresponds to the deviation distance with respect to the straight-line path at the end of the simulation period. That deviation is negated in order to be maximized. This term is maximized when the robot reaches the desired destination along the straight path at the end of the simulation period. The fourth term is the percentage of time within the simulation period that the robot's ZMP stability margin is above a given threshold. That term is maximized when the robot's stability is optimal during the various motion stages (both single-support and double-support modes).

In order to obtain acceptable locomotion patterns, two restrictions were imposed to the solutions yielded by the GA. The first restriction prevents solutions with large torso inclinations. In particular, solutions with a torso inclination above 16 degrees were rejected in this work. With lower thresholds, the GA hardly found valid solutions, whereas higher thresholds led to unnatural bent postures while walking. The second restriction is associated with the ground clearance. Specifically, it is required that the swing foot be parallel to the floor and with the sole's height higher than 1 cm for the swing leg most of the time. That guarantees a correct interaction between the robot and the floor, as well as the avoidance of small obstacles.

TABLE V
NAO'S JOINTS WITH CONSTANT ANGLES

Joint name	*Angle (rad)*	*Joint name*	*Angle (rad)*
$HeadPitch$	0	$LShouldRoll$	0.23
$HeadYaw$	0	$LElbowYaw$	-1.61
$RShouldRoll$	-0.23	$LElbowRoll$	-0.5
$RElbowYaw$	1.61	$HipYawPitch$	0
$RElbowRoll$	0.5		

TABLE VI
GENETIC ALGORITHM SEARCH SPACE

CPG parameters	*Parameter range*	*CPG parameters*	*Parameter range*
a	0.1 to 0.5	f	0 to 2
b	0.4 to 0.9	$bias1$	-1.7 to 0.5
c	0.1 to 0.5	$bias2$	0.5 to 2
d	0.95 to 2.15	$bias3$	-1 to -0.2
e	0.95 to 2.15	$bias4$	1.35 to 1.43

C. Feedback strategies

Some feedback pathways have been introduced in the CPG described above in order to adjust the locomotion pattern in real time.

1) Posture controller: The posture controller keeps the robot's trunk in an upright position by using information provided by the robot's gyrometer and accelerometer. The trunk inclination in the sagittal plane can be controlled by changing the value of parameter $bias1$ in (3). This parameter is set proportionally to the difference between the reference inclination θ and the current trunk inclination estimated from the sensors, $\hat{\theta}$, as well as to the derivative of that difference, both in radians:

$$bias1 = bias1_0 + k_1(\theta - \hat{\theta}) + k_2\frac{d(\theta - \hat{\theta})}{dt},$$

where $bias1_0$ is the original $bias1$ parameter.

2) Stepping controller: It regulates the interaction between the robot's feet and the ground by synchronizing the output signals generated by the CPG with the real time interaction between the robot and the floor by using the measures provided by the force sensors located in the robot's feet soles. Such synchronization is performed by taking advantage of the entrainment property of neural oscillators. Thus, the frequency of the generated locomotion pattern is adjusted according to the current interaction between the feet soles and the floor. This allows the control system to compensate for both external perturbations and mismatches related to the robot's mechanical parts. Furthermore, if the stride length is set to zero, this controller guarantees the correct stepping.

Let L_f, L_b, L_l and L_r be the force measures corresponding to the four force sensors located at the front, back, left and right positions of the left foot, respectively. Likewise, let R_f, R_b, R_l and R_r be the corresponding force measures for the right foot. The stepping controller is defined as:

$$\begin{aligned} F_L &= L_f + L_b + L_l + L_r \\ F_R &= R_f + R_b + R_l + R_r \\ f_1 = f_2 &= k_3(-F_L + F_R) \\ f_3 = f_4 &= -f_1, \end{aligned}$$

where f_1, f_2, f_3 and f_4 are the feedback inputs corresponding to the respective 4 neurons of the CPG (1).

3) Stride length controller: It modulates the stride length ξ by taking into account the stability margin along the sagittal plane, μ_X, which is measured in centimetres. The goal is to lower the stride length whenever the stability margin is reduced in order to recover stability. The stride length is redefined as:

$$\xi = \begin{cases} k_4\mu_X, & \mu_X <= \kappa \\ \xi_0, & \mu_X > \kappa, \end{cases}$$

where κ is a threshold that has experimentally been set to 3 cm and ξ_0 is the original stride length.

D. Omnidirectional controller

In real applications, it is necessary that the robot explores its workspace by changing its walking direction at any moment. In particular, a joint located in the robot's pelvis is used to control the walking direction in order to describe a circular motion in either the clockwise or counterclockwise directions. That joint in the NAO is referred to as $HipYawPitch$. The following controller is utilized to determine its angle in radians:

$$HipYawPitch = k_5(y_1 - y_3),$$

where y_1 and y_3 are the corresponding CPG's output signals and k_5 is a variable whose magnitude is inversely proportional to the curvature radius and whose sign determines whether the direction of circular motion is clockwise (negative sign) or counterclockwise (positive sign).

E. Phase resetting controller

Phase resetting is a fast and simple feedback strategy that has also been used to change the phase of the locomotion pattern generated by the control system in order to recover the robot's balance whenever an external perturbation is applied to the robot's body. This effective feedback strategy is suitable for humanoid robots with reduced computational capability since it does not require a complex processing of data.

The closed-loop system for locomotion control of biped robots with phase resetting must detect the external force applied to the robot's body through the fast analysis and tracking of the measures provided by the robot's sensors. Once the external perturbation is detected by the system, it must react by activating the phase resetting mechanism in order to quickly recover balance.

This controller synchronizes the neurons' output signals in order to modify the current phase of the locomotion pattern generated by the system to a desired phase given an external event or stimulus, such as an external force applied to the robot's body. The aim of this mechanism is the generation of a force in the direction opposite to the one of the force generated by the external perturbation by changing the phase of the current locomotion pattern in order to guarantee the fast recovery of balance.

The information provided by the 3-axis accelerometer is used to detect the instant at which the external force is applied to the robot's body and also to estimate the magnitude and direction of the external force applied to the robot's body. According to the current phase of the generated locomotion pattern and the external force applied to the robot, the phase resetting controller must react by changing the current phase of the locomotion pattern to another phase that allows the robot to recover its balance. The phase change is effective after Δt seconds.

III. Experimental results

The proposed locomotion control system has been tested on a NAO biped robot in simulation on the Webots simulator. For determining the gains and biases of the locomotion controllers for any given velocity, the GA interacts with the Webots simulator in order to evaluate the different individuals belonging to every generation. A total of 12,000 individuals were evaluated for every generation in order to cover a wide range of possible solutions within the search space. Each individual requires the simulation of the robot while walking in straight line during the evaluation period, which was set to 30 seconds in this work.

The proposed system has been evaluated upon 5 reference velocities: 1, 3, 5, 7 and 9 cm/s, which span the same speed range as tested in [11]. For each reference velocity, the GA was executed 50 times in order to find out the best combination of internal parameters of the locomotion controllers. The GA stops whenever either the fitness function does not significantly vary for a predefined number of generations (3 generations with a fitness variation below 0.001 in this work) or a maximum number of generations is reached (8 generations in this work). Only the solutions whose fitness values were above a predefined threshold (2.4 in this work) were selected and the median of their corresponding parameters computed. Table VII shows the median values of those parameters for the 5 tested velocities. A total of 7 solutions had a fitness value above the predefined threshold for 1, 3 and 7 cm/s, whereas 4 solutions passed that threshold for 5 and 9 cm/s. Those solutions represent optimal locomotion patterns for the given reference velocities. Intermediate velocities can be obtained without changing the selected pattern by slightly modifying the values of the stride length, ξ, and/or the frequency gain, K_f.

The control scheme proposed was evaluated in simulation studies using a workspace that consists of an ascending 10-

TABLE VII
OPTIMAL PARAMETERS OF LOCOMOTION CONTROLLERS FOUND BY THE GA FOR SEVERAL VELOCITIES

Velocity [cm/s]	1	3	5	7	9
a	0.19016	0.28748	0.33363	0.36681	0.40146
b	0.40524	0.58164	0.67348	0.72823	0.90000
c	0.20877	0.20030	0.22286	0.28086	0.35967
d	1.68173	1.84968	1.93682	1.94680	2.01228
e	2.14299	2.07704	1.93374	1.83895	1.67558
f	0.73806	0.85100	0.93300	1.21052	1.28011
bias1	-0.99668	-0.88754	-1.02878	-1.08941	-1.09422
bias2	1.68364	1.47947	1.47344	1.41980	1.50161
bias3	-0.74073	-0.68465	-0.62478	-0.54938	-0.54021
bias4	1.37688	1.37644	1.36787	1.35089	1.42999

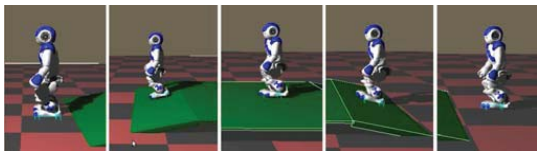


Fig. 3. System behaviour in closed-loop

Fig. 4. Turning behaviour with the omnidirectional controller

degree slope, followed by a flat surface and a final descending 10-degree slope. The robot started and stopped walking on the flat surface on both sides of the slope.

Fig. 3 contains a sequence of snapshots showing the performance of the system while successfully traversing the workspace at a velocity of 5 cm/s. Fig. 4 shows an example of a circular motion in the counterclockwise direction described by the robot using the omnidirectional controller and the optimal parameters found for the walking velocity of 5 cm/s. The stride length ξ in that case was set to zero in order to be able to turn in place. The feedback gains that successfully deal with that environment at that speed were heuristically found in simulation. Future work will aim at automatically finding those feedback gains according to the available sensory information in order to deal with increasingly challenging environments.

A. Phase resetting experiment

In this section, a simple experiment is presented to show the suitability of the phase resetting controller for fast recovery of balance in biped robots. In the experiment described below, the external force was considered to be applied to the robot's head along a known direction defined manually. This force guarantees that the robot will fall down when the feedback mechanism is not activated. Therefore, it has been used to test the system operating in both open and closed loop. The simulator allows the definition of the exact point in the robot's body in which the force is applied, as well as its desired

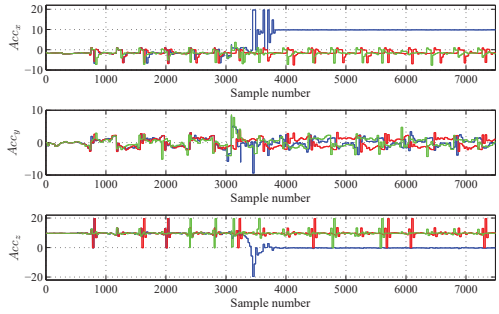


Fig. 5. Measures provided by the accelerometer located in the robot's trunk. The measures are in $\frac{m}{s^2}$. In the plots, the red line represents the system response when there is no external force applied to the robot's body. Thus, the robot is just walking. The blue line represents the behaviour when the external force is applied to the robot's head and the phase resetting controller is not activated. Finally, the green line represents the behaviour when the phase resetting controller is activated and the external force is applied to the robot's head.

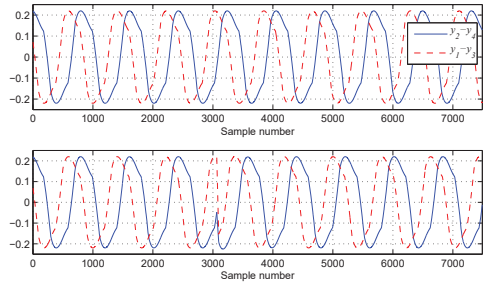


Fig. 6. Output signals of the 4-neuron CPG network shown in fig. 1. The plots represent the system's response without (top) and with (bottom) the proposed phase resetting mechanism.

magnitude and direction. The external force was also applied at a known phase of the locomotion pattern and at the same point on the robot's body in order to test the system under the same dynamic conditions.

The external force was applied at the instant in which the robot is standing on a single foot (right foot at the highest position and left foot supporting the full robot's weight). This pose was chosen as an example to validate that the control system is able to deal with unstable situations. Figure 7 represents the instant at which the external force is applied to the robot's head while the robot is standing on its left foot.

The locomotion pattern was generated by means of the proposed CPG-joint-space control scheme, with the parameters found for the straight-line locomotion pattern by considering a walking speed of 5 cm/s. In the experiment, the controller's response time (Δt) was set to 40 ms. However, this time could be smaller according to the desired system's response.

Figure 5 represents the measures provided by the robot's accelerometer for 3 possible situations, namely, the system response in open-loop without any external force applied to

Fig. 7. System behaviour with phase resetting off.

Fig. 8. System behaviour with phase resetting on.

the robot's head (red), the system response in open loop with the external force applied to the robot's head (blue) and, finally, the system response in closed-loop with the external force applied to the robot's head (green). The sampling time was set to 1.7 ms. The information provided by the robot's accelerometer was used in order to determine the instant in which the external force is applied to the robot's body and thus the phase resetting controller is activated. The effect of the phase resetting mechanism in the output signals generated by the CPG network used to control the generated locomotion pattern is shown in fig. 6. From these figures it can be observed the fast and stable response produced by the system.

The effect of the phase resetting mechanism can be appreciated in fig. 6 and in the plots shown in fig. 5. The external force is detected by the system in sample number 3064. The feedback mechanism is activated at that instant. After the controller's response time (40 ms) the system compensates for the external force applied to the humanoid robot's head through a fast motion that generates a force in the opposite direction. This minimizes the effect of the external perturbation and manages to recover balance quickly.

A sequence of snapshots showing the performance of the robot when phase resetting is off and on are shown in fig. 7 and fig. 8, respectively. These experiments have shown that the closed-loop response is fast and effective, which makes this system suitable for humanoid robots with reduced processing capabilities. This system can also deal with larger forces than those tackled by other control strategies.

Experimental results showing the behaviour of the overall system in the simulated workspace can be found on the companion website[1].

IV. Conclusions

The proposed system belongs to the joint-space category, as the CPG output signals drive the angular position of the robot's joints through a set of controllers whose optimal configuration of internal parameters is computed through an evolutionary GA given a desired walking speed in straight line. The proposed CPG guarantees that the open-loop control system generates a locomotion pattern that correctly interacts with the floor. It also straightforwardly modulates the locomotion patterns through sensory feedback so that the robot can cope with uneven terrain and transitions between different types of ground, and facilitates additional feedback controllers to deal with external perturbations. This is a very important feature because it enables the system to be improved incrementally by adding controllers so that more complicated situations can be copes with. The performance of the proposed control system has been assessed through simulation experiments on a NAO humanoid robot, showing the effectiveness of the proposed approach, although it can also be applied to other families of humanoid robots with a similar kinematic structure.

Future work will include the rigorous study of feedback controllers in order to cope with more complex types of terrain and external perturbations. Furthermore, a rigorous study about the variation of the internal parameters of the locomotion controllers (gains and biases) will be conducted with the final aim of establishing mathematical models that allow the system to automatically determine optimal parameters for any required velocity and direction, without executing the GA-based optimization process for every new speed. Finally, it is necessary to define feasible strategies to automatically compute the feedback gains based on sensory information about the environment in order to be able to cope with increasingly challenging real environments.

[1]Companion website: https://youtu.be/Pl71G04ujws

References

[1] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks*, 21(4):642–653, 2008.

[2] Junzhi Yu, Min Tan, Jian Chen, and Jianwei Zhang. A survey on cpg-inspired control models and system implementation. *Neural Networks and Learning Systems, IEEE Transactions on*, 25(3):441–456, March 2014.

[3] Gen Endo, Jun Morimoto, Takamitsu Matsubara, Jun Nakanishi, and Gordon Cheng. Learning CPG-based Biped Locomotion with a Policy Gradient Method: Application to a Humanoid Robot. *The International Journal of Robotics Research*, 27(2):213–228, 2008.

[4] Shinya Aoi and Kazuo Tsuchiya. Locomotion Control of a Biped Robot Using Nonlinear Oscillators. *Autonomous Robots*, 19(3):219–232, 2005.

[5] Shinya Aoi and Kazuo Tsuchiya. Adaptive behavior in turning of an oscillator-driven biped robot. *Autonomous Robots*, 23(1):37–57, 2007.

[6] Inyong Ha, Yusuke Tamura, and H. Asama. Gait pattern generation and stabilization for humanoid robot based on coupled oscillators. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3207–3212, 2011.

[7] Chengju Liu, Danwei Wang, and Qijun Chen. Central Pattern Generator Inspired Control for Adaptive Walking of Biped Robots. *Systems, Man, and Cybernetics: Systems, IEEE Transactions on*, 43(5):1206–1215, 2013.

[8] K. T. Song and C. H. Hsieh. Cpg-based control design for bipedal walking on unknown slope surfaces. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 5109–5114, May 2014.

[9] L. Righetti and A.J. Ijspeert. Programmable central pattern generators: an application to biped locomotion control. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1585–1590, 2006.

[10] J. Morimoto, G. Endo, J. Nakanishi, and G. Cheng. A Biologically Inspired Biped Locomotion Strategy for Humanoid Robots: Modulation of Sinusoidal Patterns by a Coupled Oscillator Model. *Robotics, IEEE Transactions on*, 24(1):185–191, 2008.

[11] V. Matos and Cristina P. Santos. Central Pattern Generators with Phase Regulation for the Control of Humanoid Locomotion. In *IEEE-RAS International Conference on Humanoid Robots*, Japan, 2012.

[12] Miguel Oliveira, Vitor Matos, Cristina P. Santos, and Lino Costa. Multi-objective parameter CPG optimization for gait generation of a biped robot. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3130–3135, 2013.

[13] J. Nassour, V. Hugel, F.B. Ouezdou, and G. Cheng. Qualitative adaptive reward learning with success failure maps: Applied to humanoid robot walking. *Neural Networks and Learning Systems, IEEE Transactions on*, 24(1):81–93, Jan 2013.

[14] J. Cristiano, D. Puig, and M. A. Garcia. Efficient locomotion control of biped robots on unknown sloped surfaces with central pattern generators. *Electronics Letters*, 51(3):220–222, 2015.

[15] Kiyotoshi Matsuoka. Sustained oscillations generated by mutually inhibiting neurons with adaptation. *Biological Cybernetics*, 52(6):367–376, 1985.

[16] Kiyotoshi Matsuoka. Analysis of a neural oscillator. *Biological Cybernetics*, 104(4-5):297–304, 2011.

[17] Dingguo Zhang, Philippe Poignet, Ferdinan Widjaja, and Wei Tech Ang. Neural oscillator based control for pathological tremor suppression via functional electrical stimulation. *Control Engineering Practice*, 19(1):74 – 88, 2011.

[18] David Gouaillier, Vincent Hugel, Pierre Blazevic, Chris Kilner, Jerome Monceaux, Pascal Lafourcade, Brice Marnier, Julien Serre, and Bruno Maisonnier. Mechatronic Design of NAO Humanoid. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 769–774. IEEE, May 2009.

[19] F. Danion, E. Varraine, M. Bonnard, and J. Pailhous. Stride variability in human gait: the effect of stride frequency and stride length. *Gait & Posture*, 18(1):69–77, 2003.

[20] Kevin M. Passino. *Biomimicry for Optimization, Control, and Automation*. Springer, 1st edition. edition, August 2004.

Making compatible two robotic middlewares: ROS and JdeRobot

Satyaki Chakraborty and José M. Cañas

***Abstract*—The software in robotics makes real the possibilities opened by the hardware. In contrast with other fields, robotic software has its own requirements like real time and robustness. In the last years several middlewares have appeared in the robotics community that make easier the creation of robotic applications and improve their reusability. Maybe ROS (Robot Operating System) is the most widespread one, with a wide user and developer community. This paper presents the work towards making compatible two of them, JdeRobot and ROS, both component oriented. A compatibility library has been developed that allows JdeRobot components to directly interoperate with ROS drivers, exchanging ROS messages with them. Two experiments are presented that experimentally validate the approach.**

***Index Terms*—robotics middleware, software, ROS**

I. Introduction

MOST of the robot intelligence lies on its software. Once the robot sensor and actuator devices are set, the robot behavior is fully caused by its software. There are many different ways to program in robotics and none is universally accepted. Some choose to use directly languages at a very low level (assembler) while others opt for high-level languages like C, C++ or Java.

Good programming practices are an emerging field in the software engineer area but also in robotics. Several special issues of robotics journals, books on the topic have been published and also specific workshops and tracks have been created inside ICRA and IROS. The [1]Journal of Software Engineering for Robotics promotes the synergy between Software Engineering and Robotics meanwhile the IEEE Robotics and Automation Society (TC-SOFT) has founded the Technical Committee for Software Engineering for Robotics and Automation.

Compared with other computer science fields, the development of robot applications exhibits some specific requirements. First, liveliness and real-time operation: software has to take decisions within a fast way, for instance in robot navigation or image processing. Second, robot software has to deal with multiple concurrent sources of activity, and so the architecture tends to be multitasking. Third, computing power is usually spread along several connected computers, and so the robotic software tends to be distributed. Fourth, the robotic software typically deals with heterogeneous hardware. New sensors and actuator devices continuously appear in the market and this makes complex the maintenance and portability to new robots or devices. Fifth, Graphical User Interface (GUI) and simulators are mainly used for debugging purposes. Sixth, the robotic software should be expansible for incremental addition of new functionality and code reuse.

Mobile robot programming has evolved significantly in recent years. In the classical approach, the application programs for simple robots obtain readings from sensors and send commands to actuators by directly calling functions from the drivers provided by the seller. In the last years, several *robotic frameworks* (SDKs, also named middlewares) have appeared that simplify and speed up the development of robot applications, both from robotic companies and from research centers, both with closed and open source. They favor the portability of applications between different robots and promote code reuse.

Middlewares offer a simple and more abstract access to sensors and actuators than the operating systems of simple robots. The SDK *Hardware Abstraction Layer* (HAL) deals with low level details accessing to sensors and actuators, releasing the application programmer from that complexity.

They also provide a particular software architecture for robot applications, a particular way to organize code, to handle code complexity when the robot functionality increases. There are many options here: calling to library functions, reading shared variables, invoking object methods, sending messages via the network to servers, etc. Depending on the programming model the robot application can be considered an object collection, a set of modules talking through the network, an iterative process calling to functions, etc.

In addition, robotic frameworks usually include simple libraries, tools and common functionality blocks, such as robust techniques for perception or control, localization, safe local navigation, global navigation, social abilities, map construction, etc. They also ease the code reuse and integration. This way SDKs shorten the development time and reduce the programming effort needed to code a robotic application as long as the programmer can build it by reusing the common functionality included in the SDK, keeping themselves focused in the specific aspects of their application.

As developers of JdeRobot framework since 2008 the authors faced a strategic decision: competing with ROS is pointless, instead of that, it is more practical to make compatible JdeRobot applications with ROS framework. From the point of view of the small JdeRobot team, one advantage is to reduce the need of development of new drivers, using instead the ROS ones and focusing the efforts in the applications themselves. Another advantage is the direct use of ROS

Satyaki is with Jadavpur University and José M. is with Universidad Rey Juan Carlos

E-mail: satyaki.cs15@gmail.com, josemaria.plaza@urjc.es

[1]www.joser.org

datasets and benchmarks, which are increasingly common in robotics scientific community. The aim of this paper is to present the current approach to compatibility between ROS and JdeRobot.

Section II gives an introduction to ROS and JdeRobot frameworks. Section III presents two previous approches, while section IV describes the current proposed approach. Two experiments of the compatibility library working are presented in section V. Some conclusions finish the paper.

II. Two robotic middlewares: ROS and JdeRobot

Cognitive robotic frameworks were popular in the 90s and they were strongly influenced by the Artificial Intelligence (AI), where planning was one of the main key issues. One of the strengths of such frameworks was their planning modules built around a sensed reality. A good example was Saphira [12], based on a behavior-based cognitive model. Even though the underlying cognitive model usually is a good practice guide for programming robots, this hardwired coupling often leads the user to problems difficult to solve when trying to do something that the framework is not designed to support.

Modern robotic frameworks are more based on software engineering criteria. Key achievements are (1) the hardware abstraction, hiding the complexity of accessing heterogeneous hardware (sensors and actuators) under standard interfaces, (2) the distributed capabilities that allow to run complex systems spread over a network of computers, (3) the multiplatform and multi-language capabilities that enables the user to run the software in multiple architectures, and (4) the existence of big communities of software that share code and ideas.

One relevant middleware was Player/Stage [3], the de facto standard ten years ago. Stage is a 2D robot simulation tool and Player is network server for robot control. Player provides a clean and simple interface to the robot's sensors and actuators. The client program talks to Player over a TCP socket, reading data from sensors, writing commands to actuators, and configuring devices on the fly. Client programs can be written in any of the following languages: C++, Tcl, JAVA, and Python. In addition, the client program can be run from any machine that has a network connection to the robot or to the machine on which the simulator is running.

Another important example is ORCA [8], [5], an open-source framework for developing component-based robotic systems. It provides the means for defining and developing the building-blocks which can be pieced together to form arbitrarily complex robotic systems, from single vehicles to distributed sensor networks. It uses the ICE communication middleware from ZeroC and its explicit interface definition to exchange messages among the components. It was discontinued in 2009, but was very influential.

Other relevant component-based framework is RoboComp [10], [11] by Universidad de Extremadura. It is open source and also uses ICE communication middleware as glue between its components. It includes some tools based on Domain Specific Languages to simplify the whole development cycle of the components. Most component code is automatically generated from simple and abstract descriptions over a component template. In addition, RoboComp includes a robot simulation tool that provides perfect integration with RoboComp and better control over experiments than current existing simulators.

Other open source frameworks that have had some impact on current the state of the are CARMEN by Carnegie Mellon and Miro by University of Ulm. They also use some component-based approach to organize robotic software using IPC and CORBA, respectively, to communicate their modules. There are also closed source frameworks as well, like Microsoft Robotics Studio or ERSP by Evolution Robotics.

A. ROS

The [2]Robot Operating System (ROS) [9] is one of the biggest frameworks nowadays. It was founded by Willow Garage as an open source initiative and it is now maintained by Open Source Robotics Foundation. It has a growing user and developer community and its site hosts a great collection of hardware drivers, algorithms and other tools. ROS is a set of software libraries and tools that help to build robot applications. From drivers to state-of-the-art algorithms, and with powerful developer tools simplifies the development of robotics projects. It is multiplatform and multilanguage.

The main idea behind ROS is an easy to use middleware that allows connecting several components, named *nodes*, implementing the robotic behavior, in a distributed fashion over a network of computers using hybrid architecture. ROS is developed under hybrid architecture by *message* passing, mainly in publish-subscribe fashion (*topics* in Figure 1). Message passing of typed messages allows components to share information in a decoupled way, where the developer does not require to know which component sends a message, and vice versa, the developer does not know which component or components will receive the published messages.

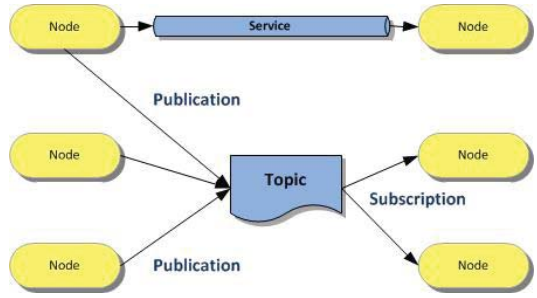


Fig. 1. ROS messages: topics, services

Nodes send and receive messages on topics. A topic is a data transport system based on a subscribe/publish system. One or more nodes are able to publish data to a topic, and one or more nodes can read data on that topic. A topic is typed, the type of data published (the message) is always structured in the same way. A message is a compound data structure. It comprises a combination of primitive types (character strings, Booleans, integers, floating point, etc.) and messages (a message is

[2]http://www.ros.org/

a recursive structure). RPC mechanisms (like *services*) are available as well.

Resources can be reached through a well defined naming policy and a ROS master. Current release is Jade Turtle, the 9th official ROS release. It is supported on Ubuntu Trusty, Utopic, and Vivid.

B. JdeRobot

The JdeRobot platform[3] is a component based framework that uses the powerful object oriented middleware ICE from ZeroC as glue between its components. ICE allows JdeRobot to run in multiple platforms and to have components written in any of the most common programming languages interacting among them. Components can also be distributed over a network of computational nodes and by extension use all the mechanisms provided by ICE as secure communications, redundancy mechanisms or naming services.

The main unit for applications is the component. A component is an independent process which has its own functionality, although it is most common to combine several of these in order to obtain a more complex behavior. There are several types of components, according to the functionality. *Drivers* offer a HAL (Hardware Abstraction Layer) to communicate with the different devices inside the robot (sensors and actuators). The entire configuration needed by the components is provided by its configuration file.

The communication between JdeRobot components occurs through the ICE (Internet Communications Engine) middleware. The ICE communication is based on interfaces and has its own language named *slice*, which allows the developer to define their custom interfaces. Those interfaces are compiled using ICE built-in commands, generating a translation of the slice interface to various languages (Java, C++, Python...). This allows communication between components implemented in any of the languages supported by ICE.

JdeRobot widely uses third party software and libraries (all of them open source) which greatly extends its functionality: OpenCV for image processing; PCL for point cloud processing; OpenNi for the RGB-D support, Gazebo as the main 3D robot simulator and GTK+ for the GUI implementations. Besides, JdeRobot provides its own libraries which give to robotics commonly used functionality for the developing of applications under this framework. It also provides several tools. For instance, it includes `CameraView` tool to show images from any source and includes `kobukiViewer` tool for teleoperating a Kobuki robot and show the data from all its sensors (cameras, laser, encoders).

It has evolved significantly since its inception and it is currently at its 5.3 version, which can be installed from packages both in Ubuntu and Debian Linux distributions.

III. Previous works

A. Translator component

In the first approach towards JdeRobot-ROS compatibility we developed a standalone process that translates ROS messages to JdeRobot ICE interfaces and viceversa [1]. This adaptor process is named *jderobot_ros* and allows JdeRobot components to talk to ROS nodes, and allows ROS nodes to communicate with JdeRobot components. It links with both the ICE middleware and the ROS libraries. The translation for every message has to be explicitely coded. The compatibility is intended just for the common sensors and actuators.

In Figure 2 the images from a JdeRobot camera in Gazebo simulator reach the *camera_dumper* ROS node.

Fig. 2. JdeRobot camera in Gazebo reaches the camera-dumper ROS node

In Figure 3 a ROS Pioneer robot in Gazebo is handled from the JdeRobot *teleoperatorPC* component, that shows images from the robot stereo pair, data from the laser sensor and sends motor commands.

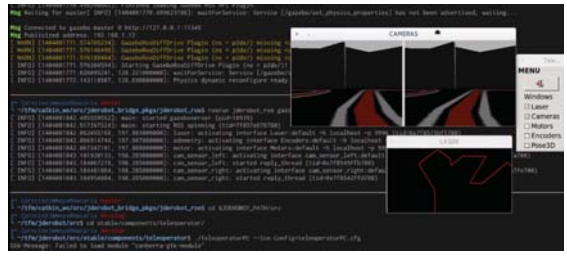


Fig. 3. ROS robot in Gazebo reaches the teleoperatorPC JdeRobot component

B. ROS-ICE Bridge with Template class

In the second approach we tried to avoid the additional translator process [2]. Then we developed a template class to be used in the JdeRobot component that wants to connect to ROS nodes, and to be used in the ROS node that wants to connect to any JdeRobot component. Again, the translation for every message has to be explicitely coded. The compatibility is intended just for the common sensors and actuators.

The *ROS-ICE Bridge* is implemented by using an abstract, template class, which contains an *Ice Proxy*, and also, pointers to *ROS* core components like: ROS-Node, ROS-Publishers and ROS-Subscribers.

1) Sending data from ROS Publisher to JdeRobot component: The workflow for sending data from a ROS interface to a JdeRobot application is described in figure 4. Basically, a derived object from the ROS-Ice class initializes a ROS-Subscriber for the corresponding topic and it also implements

[3]http://jderobot.org

a ROS-callback method. In this function, the derived object must translate the received input into a *slice* format and then send it over to the JdeRobot component, over the Ice-Proxy.

It is worth mentioning that this workflow could be useful if a developer would like to use a graphical user interface from the `Rviz` package. The Rviz package in ROS not only provides features to send data to a backend application, but it also offers a window to visualize real time 3D models of the robot in question.

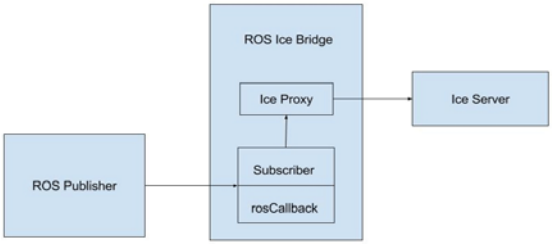


Fig. 4. Workflow of ROS Publisher to ICE component

Another interesting use is getting data on ROS log files from JdeRobot applications. In ROS it is possible to record data in so called *rosbag* files. The framework also provides a way to play-back the recorded data in a synchronized manner and to publish the information on the same topics that were registered. This is possible because at recording time, the timestamp and the channel of the transmitted information, are stored in the *bag* file as well. In this situation, the workflow of the program is as in Figure 4. The only difference is that the ROS Publisher is not developed by the user, instead it is the player in ROS framework which reads the data in the log file.

2) Sending data from JdeRobot component to ROS applications: The workflow of the program to receive in a ROS node data from JdeRobot components is as described in Figure 5. The idea is to create a derived object from both the *ROS-ICE* bridge and the "ICE Server" of a random sensor. Once an Ice proxy calls one of its methods, the bridge is supposed to take the input, in slice format, transform it into a ROS message and then publish it on a ROS topic.

This flow of the applications is useful, because many commercial and even industry robots are configured to be controlled and to send information over the ROS framework. Therefore the ROS application could use a JdeRobot driver, if needed, in order to interact with a device.

IV. Compatibility library

In the current approach we focused only on using ROS nodes (drivers) from JdeRobot applications, as this is the most common and useful scenario. We chose to create a library to be used in JdeRobot components. Using this library the components may talk with ROS drivers too. Regardless the data source, a JdeRobot sensor driver or a ROS sensor driver, the sensor data are mapped to the same local API. The processing side of the application code locally reads the sensor

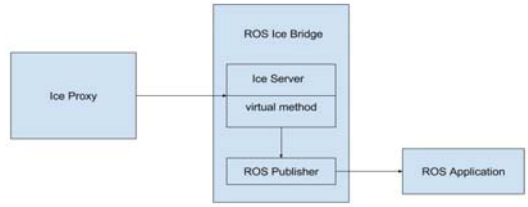


Fig. 5. Workflow of JdeRobot component to ROS application

there. Regardless the data sink, a JdeRobot actuator driver or a ROS actuator driver, the motor commands are mapped from the same local API. The processing side of the application code locally writes the actuator commands there.

The ROS nodes can be used just as the come, without touching at all their code. The library adds the capability of the JdeRobot component to directly talk to ROS nodes if configured to do so.

Current stable version of the JdeRobot middleware follows the ICE server-client architecture. While ICE interfaces help build a stable framework for server-client architectures, using ROS nodes as drivers are advantageous in terms of scalability and reusability. Keeping this idea in mind, we developed a compatibility library that translates ROS messages into local data structures in the client components. The library has been developed to allow the JdeRobot client components (currently supports the *CameraView* component and the *KobukiViewer* component) to communicate both with the ROS drivers as well as their ICE server counterparts. The communication between a typical JdeRobot client component and its ROS driver (via the compatibility library) or ICE server(directly through existing ICE interfaces) is shown in the Figure 6. All the right side of the Figure 6 lies inside the JdeRobot component.

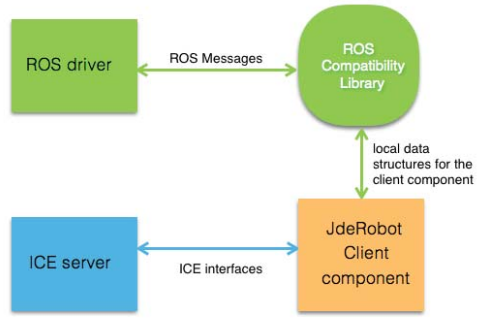


Fig. 6. Block diagram showing the communication between a JdeRobot client component with its ROS driver as well as ICE server.

The ROS compatibility library is divided into several segments. Each segment provides methods to translate ROS messages of a particular type of sensor or actuator. For

instance, the current implementation contains the following segments for: (1) Image translation (via cvBridge) (2) Laser data translation (3) Encoder data translation (4) Motor data translation. The first segment is used by the CameraView component while all the four segments are simultaneously used by the kobukiViewer component.

The JdeRobot client components are thus modified to support message retrieval from both the ICE servers as well as ROS driver. The user has the option to choose between the ROS driver and the ICE server by setting or resetting a boolean flag in the configuration file of the corresponding client component.

Using ROS drivers with the compatibility library for JdeRobot applications has one significant advantage. As discussed in the previous section, the *ROS-Ice Bridge* architecture uses an intermediate translation of ROS messages into a slice format and then follows the usual ICE server-client architecture to communicate with a JdeRobot component and vice versa. This method (from sending message from ROS publisher to ICE server) involves three steps: (1) sending the message from ROS publisher to ROS subscriber (2) Translating the message into a slice format (3) communicating the information between an ICE client and an ICE server via the ICE interfaces. The same remains true for sending the message from ICE client to ROS subscriber. In order to cut down the overhead, its beneficial to modify the client components by providing them with the ability to receive information directly from either its ICE server or its ROS driver.

V. Experiments

This section contains information about how the ROS compatibility library has been used for the *CameraView* component and the *KobukiViewer* component.

A. CameraView with a ROS camera node

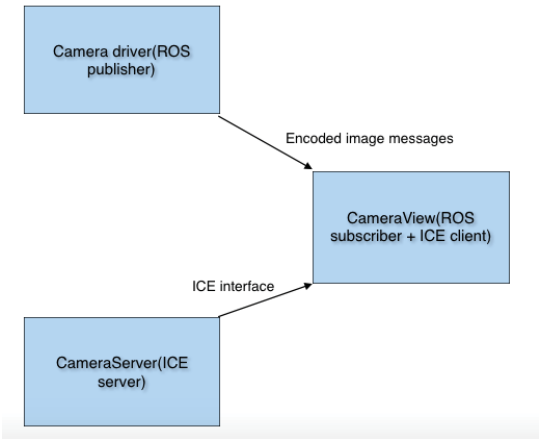


Fig. 7. CameraView component may communicate both with a ROS driver as well as an ICE server.

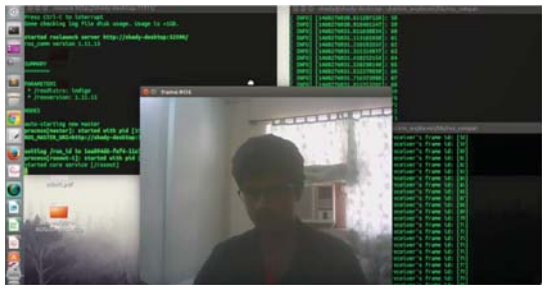


Fig. 8. The ROS camera driver is publishing images taken from a USB webcam while the CameraView component is receiving images from the ROS driver and displaying them.

In the current implementation, the *CameraServer* component (the ICE server component) was replaced by a ROS driver. The ROS driver is in fact a ROS node publishing images as ROS messages via *cvBridge*. On the other hand, the *CameraView* component has been modified to act as a ROS subscriber that receives the encoded image messages from the driver. The compatibility library takes care of translating this message into an OpenCV *Mat* object, which is then displayed in the *CameraView* component. Also, along with the images, the frame number or id is also published to keep track of whether the frames are arriving in sequence or not. Figure 8 shows the `CameraView` component being driven by a ROS publisher.

The `CameraView` component is a relatively simple component as in this case only a single node is publishing ROS messages in the driver and in the client component, only a single node is listening to those messages. Hence there is no need of concurrency control or multithreaded spinners. In the next subsection we cover a more challenging problem where the client component receives messages from multiple ROS publishers.

B. KobukiViewer with a ROS Kobuki robot

JdeRobot has support for running and testing the koukiRobot or the Turtlebot in a simulated gazebo world. The simulated Kobuki robot (Figure 9) has two cameras, one 2D laser scanner and encoders as sensors, and motors as the only actuators. The ICE servers are implemented as gazebo plugins and the ICE client component (the *KobukiViewer* component) communicates with the servers through different ICE interfaces. The ICE interfaces currently supported by the KobukiRobot simulation in JdeRobot are: 1) Camera Images 2) Laser data 3) Pose3d (Encoder data) 4) Motors.

We thus modified the Gazebo plugins to run ROS nodes in stead of ICE servers. These nodes can act as ROS publisher or ROS subscriber (or both) depending on whether messages are being sent to or received from the client component. The camera driver, laser driver and the pose3d driver only send sensor data as messages to the *kobukiViewer*. Hence in these cases we only need a ROS node to publish these messages in different ROS topics. On the other hand, the motor driver

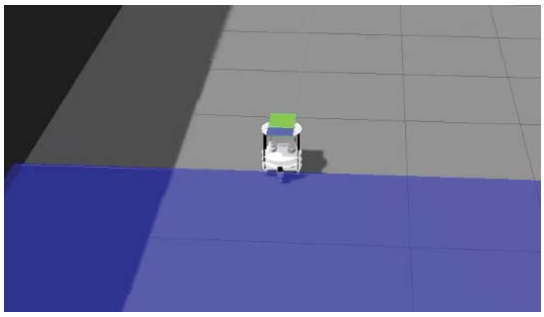


Fig. 9. Kobuki robot simulation in Gazebo.

needs to publish its values as well as receive values from the client component to update its parameters. Hence in this case, the ROS driver needs a bidirectional communication using a ROS publisher node that publishes the actuator data as well as a ROS subscriber node that listens to the client component for updating the values of the actuator.

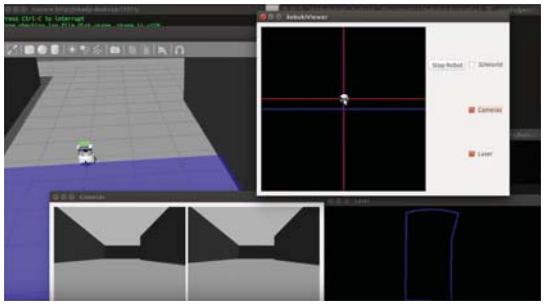


Fig. 10. Figure shows KobukiViewer component being run with ROS drivers.

In the *kobukiViewer* component, we have four ROS subscribers subscribing to each of the four ROS topics to receive sensor data and one ROS publisher to publish messages for updating the values of the motors. For each subscriber we implement a callback function where the ROS compatibility library is used to translate the ROS messages into data structures local to the *kobukiViewer* component. The ROS compatibility library is also used in the callback function in the motor driver which listens to the *kobukiViewer* component for messages in order to update the values of the actuator.

In Figure 10, we see the *kobukiViewer* component being run with ROS drivers. The component provides a Qt GUI to manually set the motor values or control which sensor readings to display. In this case, we see the 2D laser scan in the bottom right window and the left and right camera images at that instant in the window just next to it. In order to run the GUI and the ROS functions in parallel, we use a *ros::AsyncSpinner* object to run the ROS callback functions in the background thread. *ros::AsyncSpinner* unlike the single threaded *ros::spin()* function does not conform to abstract Spinner interface. Instead, it spins asynchronously when one calls *start()*, and stops when either one calls *stop()*, *ros::shutdown()* is called, or its destructor is called.

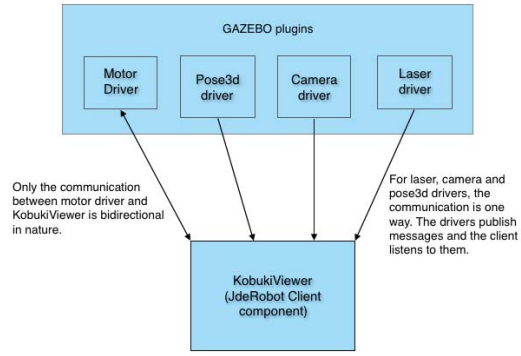


Fig. 11. Figure shows how the KobukiViewer component communicates with different ROS drivers.

Next, we illustrate a small code snippet to show the implementation of how the JdeRobot client component(s) is(are) modified to support ROS compatibility. The snippet includes two files. First is *roscompat.h*, an header file from the ROS compatibility library which contains methods responsible for message translation.

roscompat.h

```
class ros_compat {
  ...

/* public member functions */
public:
  void translate_image_messages(const sensor_msg::ImagePtr&
       msg, cv::Mat& image);
  void translate_laser_messages(const ros_compat::Num::ConstPtr
       & msg, std::vector<int>& laserdata);
  void translate_pose3d_messages(const ros_compat::Pose3d::
       ConstPtr& msg, std::vector<int>& pose);
  void translate_motor_messages(const ros_compat::Motors::
       ConstPtr& msg, std::vector<int>& motors);
  ...
}
```

In order to illustrate how these functions are used inside the *KobukiViewer* client component, we provide a snippet from the *Sensors.cpp* file from the component.

sensors.cpp

```
#include ''roscompat.h"
...

/* global variables */
ros_compat* rc;
std::vector<int> pose, laserdata, motors;
cv::Mat left_frame, right_frame;
...

/* callback functions */
```

```
/* Inside the callback functions */
/* the translation of ROS messages into */
/* local data structures takes place */
/* and the global variables are updated. */
void camera_left_callback(const sensor_msgs::ImageConstPtr&
      image_msg) {
  rc->translate_image_messages(image_msg, left_frame);
}
void camera_right_callback(const sensor_msgs::ImageConstPtr&
      image_msg) {
  ...
}
void pose3d_callback(const ros_compat::Num::ConstPtr& msg) {
  ...
}
/* so on */


Sensors::sensors(...) {
  ...

  /* ROS intialisation */
  ros::init(argc, argv, "kobukiclient");

  /* Create NodeHandles */
  ros::NodeHandle n_cam_left, n_cam_right, n_laser, n_pose,
       n_motors;

  /* Create ROS subscribers for each ROS topic */
  image_transport::ImageTransport it_left(n_cam_left);
  image_transport::ImageTransport it_right(n_cam_right);
  image_transport::Subscriber left_camera_sub = it.subscribe("
       leftcameratopic", 1000, camera_left_callback);
  image_transport::Subscriber right_camera_sub = it.subscribe("
       rightcameratopic", 1000, camera_right_callback);
  ros::Subscriber laser_sub = n_laser.subscribe("lasertopic", 1001,
       laserCallback);
  /* so on */

  /* Start the AsyncSpinner */
  ros::AsyncSpinner spinner(4);
  spinner.start();
  ros::waitForShutdown();

  ...

}

void Sensors::update() {
   /* update the private datamembers from the global variables */
   /* in stead of using the ICE proxies */
   /* an example is shown for the laser data*/
   /* Sensors::LaserData is a private data member */
   /* whereas laserdata is a global variable updated */
   /* when the callback function is called */

    mutex.lock();
    laserData.resize(laserdata.size());
    for (int i=0; i<laserdata.size(); i++) {
      laserData[i] = laserdata[i];
    }
    mutex.unlock();

   /* same for camera images, pose3d etc. */
   ...
}

/* other public member functions */
...
```

VI. Conclusions

The preliminary work on the third approach to allow compatibility between ROS nodes and JdeRobot components has been presented in this paper. It consists of a compatibility library that extends the capability of the components to connect to ROS nodes exchanging ROS messages. The components may connect with other JdeRobot units using ICE or with ROS units using that library. The ROS message processing is put on a library so any component dealing with the same messages may share it.

The communication is bidirectional, as sensors ('get' operations) and actuators ('set' operations) are supported. The processing for every ROS message has to be explicitely coded. The compatibility is intended just for the common sensors and actuators messages, so the JdeRobot application may use the ROS Hardware Abstraction Layer. The ROS communication side matches the same local API for sensors and actuators that the ICE communication side does, and so, the logic of the JdeRobot component is not changed at all.

The compatibility library has been validated with two experiments connecting two standard JdeRobot applications to ROS nodes. First, the cameraViewer to a ROS node that serves camera images. Second, the kobukiViewer has been connected with a Kobuki robot with motors, cameras, encoders and laser served through ROS.

The main development is focused now is to extend the compatibility library to support a drone robot like 3DR Solo drone and RGBD sensors like Kinect-2. In addition, the compatibility has been tested so far on C++ components, further work is needed to support the same extension on Python JdeRobot components.

Acknowledgment

This research has been partially sponsored by the Community of Madrid through the RoboCity2030-III project (S2013/MIT-2748), by the Spanish Ministerio de Economía y Competitividad through the SIRMAVED project (DPI2013-40534-R) and by the URJC-BancoSantander. Authors also would like to thank Google for accepting JdeRobot on its program GSoC-2015.

References

[1] http://jderobot.org/Mmoya-tfm
[2] http://jderobot.org/Militaru92-colab
[3] Brian P. Gerkey, Richard T. Vaughan, Andrew Howard. *The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems*. In Proceedings of the International Conference on Advanced Robotics (ICAR 2003), pp 317-323, Coimbra, Portugal, June 30 - July 3, 2003.
[4] J. M. Cañas and M. González and A. Hernández and F. Rivas, *Recent advances in the JdeRobot framework for robot programming*. Proceedings of RoboCity2030 12th Workshop, Robótica Cognitiva, pp 1-21, UNED, Madrid, July 4, 2013. ISBN:978-84-695-8175-9
[5] A.Makarenko, A.Brooks, T.Kaupp. *On the Benefits of Making Robotic Software Frameworks Thin*. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007). Workshop on Evaluation of Middleware and Architectures.
[6] A.Makarenko, A.Brooks, B.Upcroft. *An Autonomous Vehicle Using Ice and Orca*. ZeroC's Connections newsletter, issue 22, April, 2007.
[7] A. Makarenko, A. Brooks, T. Kaupp. *Orca: Components for Robotics*. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006). Workshop on Robotic Standardization.

[8] A. Brooks, T. Kaupp, A. Makarenko, S. Williams, and A. Oreback. *Orca: a component model and repository*. In D. Brugali, editor, Software Engineering for Experimental Robotics. Springer Tracts in Advanced Robotics, 30, p. 231-251, 2007.

[9] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. *ROS: An Open-Source Robot Operating System*. ICRA workshop on open source software. Vol. 3. No. 3.2. 2009.

[10] L. Manso, P. Bachiller, P.Bustos, P. Nuñez, R.Cintas, L.Calderita. *RoboComp: A tool-based robotics framework*. In Proceedings of Simulation, Modeling, and Programming for Autonomous Robots: Second International Conference, SIMPAR 2010, Darmstadt, Germany, November 15-18, Springer Berlin Heidelberg, pp 251–262, 2010.

[11] Marco A. Gutiérrez, A. Romero-Garcés, P. Bustos, J. Martínez. *Progress in RoboComp*, Journal of Physical Agents 7(1), pp 39–48, 2013.

[12] Kurt Konolige, Karen Myers. *The saphira architecture for autonomous mobile robots*. In book "Artificial intelligence and mobile robots", pp 211–242. MIT Press Cambridge, MA, 1998.

VisualHFSM 5: recent improvements in programming robots with automata in JdeRobot

Samuel Rey and José M. Cañas

***Abstract*—A visual programming tool, named VisualHFSM, has been improved in the JdeRobot robotics software framework. This tool provides Hierarchical Finite State Machines to program robot behaviors. The particular automaton is designed visually, with nodes for the states, links for the transitions and specific source code on them. It automatically generates a C++ or a Python JdeRobot component that connects to the drivers and implements the automaton. It uses multithreaded templates for that. Such component dynamically shows the current state of the automaton while running. This tool speeds up the development time of robot behaviors, reducing the code that has to be created from scratch for new behaviors. VisualHFSM has been experimentally validated creating several autonomous behaviors for drones.**

***Index Terms*—Visual languages, robot programming, automata**

I. Introduction

Most of the robot intelligence lies on its software. Its functionality resides in its programming, in the software that manages hardware resources like sensors and actuators. There is no universally standardized methodology to program robots. In the last few years the robotics community has begun to apply software engineering methodologies to its field, making more emphasis in code reuse, distributed software design, etc. Several robot programming frameworks that simplify the development of applications have emerged.

These frameworks (1) provide a more or less portable hardware access layer (HAL); (2) offer a concrete software architecture to the applications; (3) include tools and libraries with already ready-to-use functionality and building blocks. Many of the emerged platforms are component oriented, such as ROS, Orca, Microsoft Robotics Studio, RoboComp, JdeRobot, etc..

In several frameworks, automata have been used for robotic software development. Finite State Machines (FSM) have been largely and succesfully employed in many fields and they can also be used in robotics to symbolize the robot behaviors, representing them in a compact and abstract form. With FSM the robot behavior is defined by a set of states, each of which performs a particular task. The system can then switch from one state to another through transitions, which are conditions of state change or stay depending on certain events or sensor conditions that may happen, both internal or external. FSMs provide one smart way to organize the control code and perception on-board a mobile robot. They have been explored in research and also incorporated in recent robotic frameworks with tools that let the programmer to focus on the behavior logic more than in implementation details. With these tools most of the code is then generated automatically from the abstract description of the FSM. This diminishes the chance of bugs, reduces the development time and allows the programming of complex behaviors in a robust way.

JdeRobot is the component oriented robot programming framework developed in Universidad Rey Juan Carlos. In this paper we present the new release of the VisualHFSM tool in JdeRobot, which supports the visual programming of robot behavior using hierarchical Finite State Machines. Now it can generate Python components, not only C++ ones. The generated components now show in a runtime GUI the active states in the hierarchical FSM. Its usability has been improved and support for drones has been included.

The reminder of this paper is organized as follows. In the second section we review related works on FSM tools in different frameworks. The third section presents the current visual programming tool emphasizing the improvements from the previous release. The fourth section describes two example applications generated with VisualHFSM for simulated and real drones. Finally, conclusions are summarized.

II. Related works

Automata have been successfully used in videogame programming for generating the behavior of automatic players. For instance, Xait [1] enterprise commercializes tools that facilitates automaton programming to videogames developers, as its xaitControl (Figure 1). This tool allows the programmer to easily develop behaviors with hierarchical finite state machines. It has a main canvas in which the automaton is displayed, a tree view on the left shows the created structure and other panels show different information about auxiliary procedures, execution flow control, etc.. Another example is the successful best seller Halo 2 game of Bungie, where several automata were used to deploy more than one hundred different behaviors.

Regarding finite state machines in robotics, in ROS framework there is SMACH [2] [6], [7]. This tool is a task-level architecture for rapidly creating complex robot behaviors. At its core, SMACH is a ROS-independent Python library to build hierarchical finite state machines. To develop a hierarchical finite state machine you have to write the code needed to create and describe each state and transition, it is not a visual tool.

Samuel Rey and José M. Cañas are with Universidad Rey Juan Carlos

E-mail: s.reye@alumnos.urjc.es, josemaria.plaza@urjc.es

[1] http://www.xaitment.com

[2] http://www.ros.org/wiki/smach

Fig. 1. An instance of xaitControl

The package also comes with the SMACH-viewer (Figure 2), a tool that shows the developed finite state machine at runtime. In that tool, we can see either a graph or a tree view of the current automaton, but not both at the same time. It also shows every state and transition, as well as the active state and a debug window where we can see data of the active state.

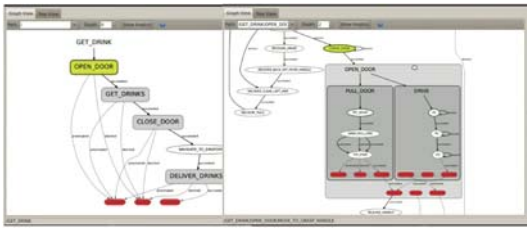


Fig. 2. An instance of SMACH-viewer

One of the most powerful frameworks that use HFSM in robotics is MissionLab, developed in Georgia Tech by the professor R. Arkin. This environment includes a graphical editor of configurations (CfgEdit) [5] as a tool, similar to automaton, that allows to specify missions with its states, transitions, etcetera. It allows to generate applications following the AuRA architecture, developed by the same group. In order to represent the automaton they developed their own language, the Configuration Description Language. A more recent example of FSM is the automaton editor inside the component-oriented platform RoboComp, from Universidad de Extremadura. For instance, in [8], they used it to program the behavior of a forklift. Another sample is the behaviors graphical editor Gostai Studio [9], inside the Urbi platform. This graphical editor generates urbiscript code as its output, includes time-execution visualization of the state of the automaton, allows to stop and continue the execution of the generated automaton and offers the possibility of creating hierarchical finite state machines.

In the RoboCup scenario finite state machines are frequently used to generate behaviors for the standard SPL league humanoid robots. Several teams use the tool and language XABSL [10], [11] to specify behaviors, around the influential German team B-Human. In addition, the TeamChaos team [12] used an HFSM tool to handle hierarchical finite state machines for its architecture ThinkingCap, allowing even behavior hot-edition in each state. In SPIteam a tool named Vicode is used to generate finite state machines for BICA architecture.

III. Improvements in VisualHFSM 5

VisualHFSM [1] is a tool created for the programming of robot behaviors using hierarchical finite states machines. It generates as output a component in JdeRobot framework that implements the robot behavior. It represents the robot's behaviour graphically on a canvas composed of states and transitions.

The source text code to be executed in each state or transition must be introduced. This tool decreases the development time for new applications, providing the developer with a higher level of abstraction. It also increases the quality of these applications, automatically generating most of the code using a clean and well organized template. The tool allows the engineer to focus on specific parts of her application, writing only the actual code that will be executed by the automaton and the conditions that will make the robot to go from one state to another. The final result is a more robust code and less prone to failure.

VisualHFSM is divided in two parts: a *graphical editor* and the *automatic code generator*. The graphical editor displays the main window where the automaton structure is created. It also contains internal structures and saves all the data into an XML file. The automatic code generator reads that XML file and delivers the source code of a JdeRobot component implementing the HFSM. It also generates a configuration file for it. The compilation can be launched from the graphical editor or outside.

The previous release of VisualHFSM had several shortcomings and limitations. Most of them have been addressed in the fifth release and are described next.

A. Better usability in the graphical editor

The graphical editor allows the developer to see, edit and add states and transitions in a clear and simple way. The component is represented by a finite state machine in a canvas, where all elements are visualized. It allows to manipulate states (create, remove, move, rename...) and to define the behavior that will be executed in them. It is also possible to set the conditions for regulating the transitions between states, and the code that will be executed when the transitions occur.

As shown in the Figure 3, the GUI in VisualHFSM 5.0 is now divided into two parts: the `Tree View` at the left and the `Scheme View` at the right. The buttons part of Figure 4, present in previous releases, is now placed in the `menu bar`, so the space of the window for creating the automaton is bigger and more comfortable. Another usability improvement is that now the canvas, the `Tree View` and all of the popup elements are scrollable.

In the `Tree View` (red border area in Figure 3) the whole automaton is text represented in the hierarchical mode with two columns for identifying the states: one for the ID and

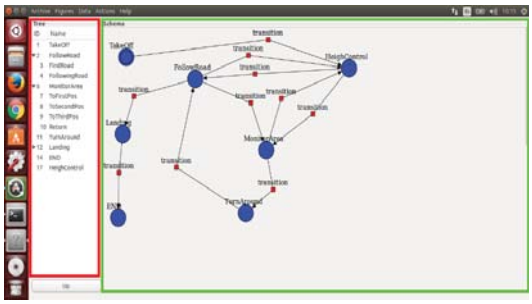


Fig. 3. Graphical editor of VisualHFSM 5.0

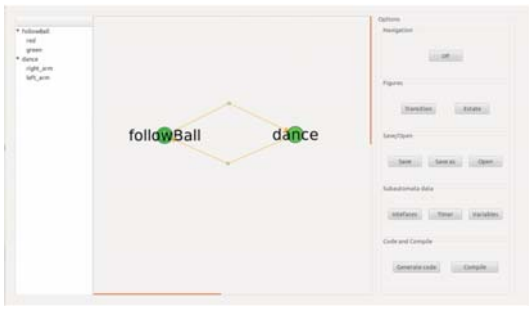


Fig. 4. Graphical editor of previous version of VisualHFSM

other for the name given to this state. It has now the option of collapsing or expanding the children of some state, so the developer can choose to see all the levels at the same time or focus on some specific levels by collapsing the rest. The children and levels of the hierarchy are represented under their fathers by using different tabulations. It allows a simple and intuitive navigation through the automaton, double clicking in one state for representing the subautomaton that it contains.

In the `Schema View` (green border area in Figure 3) the automaton is graphically drawn, showing the name of each state or transition. States are represented as blue circles and for each subautomaton it also marks with a double circle the state which must be active at the beginning. For each state, the user can rename it; edit it, allowing to change the code of the selected state; mark it as the initial state of its subautomaton; copy it, allowing to paste the selected state into another or the same subautomaton; and delete it. Any state can be connected to another by an unlimited number of transitions or to itself by auto-transitions. Transitions are represented as arrows that go from the origin state to the destiny, with a red square in the middle for applying them different actions, as move them, rename them, editing its condition, adding them code and delete them. The condition added by editing a transition is the condition that must happen for the transition to occur. The Schema View also allows to graphically navigate through the automaton double clicking in the desired state or in the `Up` button.

In the `menu bar`, menus are structured in five groups: Archive, Figures, Data, Actions and Help. The Archive menu allows to create new projects, open existing ones, saving the current project or exit VisualHFSM. Figures menu contains two buttons, for adding new states or transitions. Data menu has Timer, for choosing the frequency of the iterations, and Variables, for adding variables and functions that the developer may need for better organizing and structuring its code, giving more flexibility to the tool. Last, the Actions menu allows to add additional libraries that the automaton code may need, edit the parameters for the "Config file" that will be auto-generated with the code, generate C++ or Python code and compile the C++ code by using the CMake file generated with the code.

B. Runtime GUI in the generated C++ component

For running the automaton, the XML file is parsed and a new JdeRobot component in C++ is generated. Such C++ component implements the automaton and is generated using a carefully designed *C++ template*. Each subautomaton is implemented as a single thread, so the full automaton is a collection of concurrent threads which interact among them, sleep, are resumed, etc. as some states of the automaton are activated or deactivated. More details of this template can be found in [1].

The new VisualHFSM 5 includes some code to graphically show the (hierarchical) automaton state at runtime. If the user wants it, the generated JdeRobot component displays a runtime GUI that shows which states are active or not while running. This is very convenient for debugging.

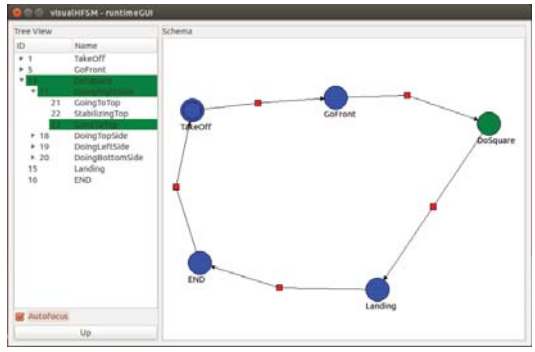


Fig. 5. Runtime GUI in C++ component using the Autofocus feature

Figure 5 shows one runtime GUI similar to the graphical editor. The current active states are displayed with green background color in the TreeView, in all levels of the hierarchy including the root. The user may expand or collapse the different hierarchical levels at will. In addition, a check box named "Autofocus" has been added. When selecting it, the Tree View will automatically expand and collapse itself for showing the active states and the rest of the automaton will be collapsed, as shown in Figure 5. In the Schema View of this runtime GUI, the active nodes are presented in green and the others in blue. When a transition takes place, the old active state is painted in blue in the Schema View and with

white background in the Tree View, and the new active state is painted in green and with green background in the Tree View.

This runtime GUI feature is off by default, because the graphical window is useful only for debugging. In normal robot operation, once its behavior programming has been refined, there is no need to spend computational resources in this additional GUI. Nevertheless, it is very useful in the developing process. Enabling this feature is as easy as executing the component with the flag `--displaygui=true`.

The runtime GUI has been implemented as an additional thread and a new library called `automatagui`. When the code or the C++ component is being generated, a C++ function is written for constructing a list containing all subautomata with its states and transitions. An object of the class `AutomataGui` is created and inserted in the component. The component will use at runtime that object for dinamically showing the automaton.

C. Generation of Python component with its runtime GUI

In the new release of VisualHFSM, the robot behavior may be programmed in Python too. Adding support for this language, VisualHFSM increases its flexibility and the component does not requires to be compiled. The code inserted for the states and transitions in the graphical editor must be in Python and a new template in Python has been developed to generate the component's final code.

The code is now organized following an object oriented model. There will be the main class `Automata`, which will contain all the information related to the automaton. This approach makes easier the communication between different subautomata, or different threads, by using elements of the `Automata` class instead of global variables. It also provides a threading lock, in case some code is sensitive to race conditions and it allows to create inner classes, in addition to more functions or variables that could be needed. The additional features will also be created inside the `Automata` class. This implementation provides more robust, better organized and cleaner code. In addition, as Python do not need to compile, it is faster to make any change on the program. The Python code is generated as an executable.

For unfolding the runtime GUI the Python component must be launched with the `--displaygui=true` parameter too. It has been programmed using the PyQt4 graphic library in Python. The way of communicating that the color of a state needs to change is simpler than in C++. It is implemented by the GUI thread again to avoid race conditions, but this time the notification is done using a handler. The thread that is going to change its active node will emit a signal with the node name, which will be handled by a handler in the GUI thread.

In this release it is possible to create several runtime GUI windows (one for one subautomaton in detail), if convenient. To activate a new window the user only has to right click over one state of the desired subautomaton and then select "Open Subautomaton". Figure 6 shows several subwindows.

D. Support for drones

Several real robots as Pioneer from ActivMedia, Kobuki (Turtlebot) from Yujin Robot, Nao from Aldebaran and their counterparts in Gazebo simulator were supported in previous releases of VisualHFSM. In the new release support for ArDrone-1 and ArDrone-2 from Parrot and simulated drones in Gazebo has been included. Their corresponding JdeRobot interfaces and their local data structures are now available for use from the code of the states and transitions.

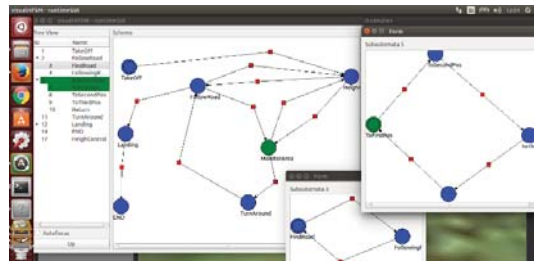


Fig. 6. State diagram of the monitor-an-area application shows three subautomata at the same time

E. Shutdown

Another feature added in this release, both in Python and in C++, is the `Shutdown` function. This function ends the loop of all the subautomata by setting the correspondent variables to false, so the code will reach the end and finish, in contrast with previous releases of VisualHFSM, where the execution never ended unless the process was manually interrupted from the terminal.

IV. Experiments

In order to validate the new features introduced with this version of VisualHFSM, we have performed two different experiments, both of them using a simulated ArDrone robot. The experiments are two robot applications developed using visualHFSM: monitor-an-area application and follow-colored-objects applications. Both of them are written using the Python code generator of VisualHFSM.

A. Monitor-an-Area application

For this first application, we have used a Gazebo scenario with a road, and some victims of a car accident laying in the ground around the location were the accident has occured. This simulated scenario is an example of an application where drones could be useful: go fast to the accident place and check with its camera the status of the victims, for the emergency service to give a better, faster and more accurate response.

We identified several states to unfold different behaviors of the robot in this example. Their combination into a HFSM fulfils the whole application. First, the drone has to take off. When it has reached the desired height, it goes to the next state `following-the-road`. Figure 7 shows the ArDrone following the road in that state. This state has a child subautomaton, for following it and considering other aspects at the same time. For instance, if the drone lost the road, this child subautomaton tooks control to search and recovers the road

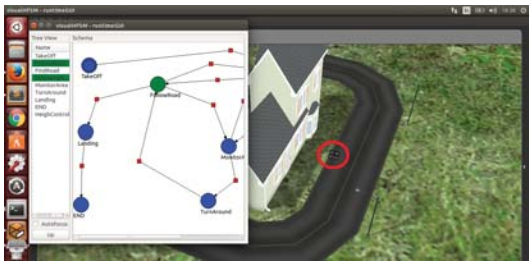


Fig. 7. ArDrone following the road

again. After finding it, it returns to the state of following the road.

It will keep in the `following-the-road` state until it gets to the point where the accident has happened. Then, it switches to the `monitor-area` state, which is responsible of looking for and locate the victims, as shown in Figure 8, where the drone has found a victim. This state has a child subautomaton for specifying phases of this activity in a simpler and more understandable way. When it has finished searching in the area, the drone will come back to the point of the road where it started to search for victims, it will turn around, and again it will go to the `following-the-road` state, following it until it reaches the point where the drone took off, and then lands there.

During all the execution, the height of the drone has been watched. If it went too hight or too low, the automaton would have switched to another state until it reaches the desired height. The state diagram of this behavior is shown in the Figure 6, where we can see the root subautomaton, and those of `following-the-road` and `monitor-area` states.

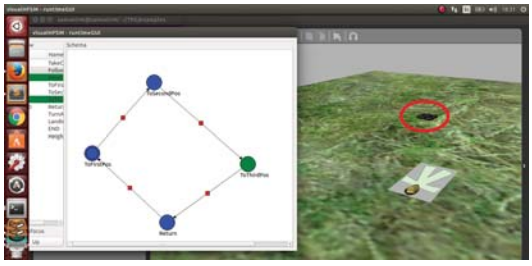


Fig. 8. ArDrone recording the location of a car accident's victim with its camera

B. Follow-colored-objects application

The motivation for this application is to use visualHFSM with a real robot, an ArDrone2 from Parrot. In this experiment there are several different colored moving objects, and the ArDrone must follow them using its ventral camera following a sequence. First, it will follow the green object until it finds something blue, then it will start following this new blue object until it finds a green object again. Finally, it will follow such green object until it finds a red one, which it will follow until the application finishes.

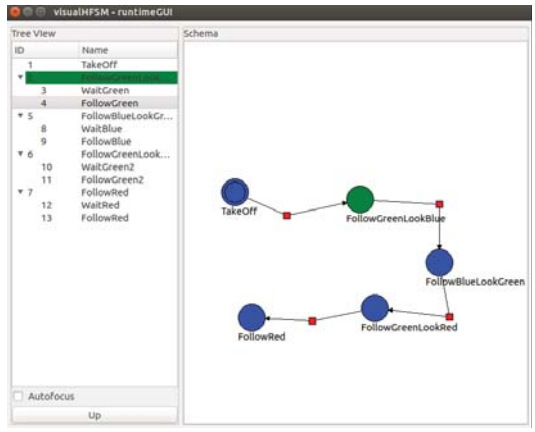


Fig. 9. State diagram of the follow-colored-objects application from its runtime GUI

This kind of behaviour perfectly matches a HFSM, as each of the four stages of following an object while looking for another can be modelled with states and transitions. The state diagram is shown in Figure 9 when the Robot is following the green object and looking for a blue object. The whole automaton is expanded on the Tree View.

As it is shown, the drone starts in the `take-off` state. When the desired height is reached, the drone will go to the state `FollowGreen-LookForBlue`. There it will be filtering two colours: green and blue. It will be following the green object until it detects a blue contour, and then it will change to `FollowBlue-LookForGreen` state. Then, to avoid the drone immediately detecting the green object that it has been following in the previous state, it will wait a blanking interval. During this interval it will only follow the blue. When such interval is over, in case of finding a green contour it will change to `FollowGreen-LookForRed` state. This state works like the other two, and when the drone finds the red object it will change to `FollowRed` state until the program finishes. Figure 10 shows the real drone following this last red object.

All the color following states are implemented with a child subautomaton. The father is in charge of detecting the colors and decides if it should continue in the current state or go to the next, while the children is responsible for following the color. We have followed this approach because the laboratory where the experiments have been performed is small, and so, when the drone loses the color it is following, it should stop. In other scenarios like open spaces it would be more convenient that when the drone loses the object, it would start a looking for manouver. With this approach it would be easier to have a new child subautomaton performing such manouver.

This experiment has been performed both using Gazebo simulator and real robots. In Gazebo it works perfectly, but we have found some difficulties while migrating it to the real

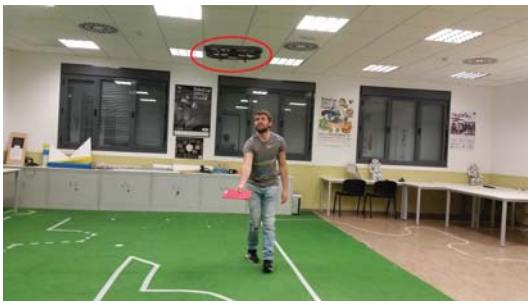


Fig. 10. ArDrone robot following a red object in the `FollowRed` state.

world. First, we had to fine tune the color filters, as making robust color filters with real light is complex. In addition, the real ArDrone flight is not as stable as in the simulator. A limitation in current visualHFSM release was detected: once the generated component with the automaton is started, it cannot be stopped until it reaches the final state. This, combined with the previous problems, is annoying when the drone does not behave as expected.

V. Conclusions

This paper has presented the new release of visualHFSM tool in JdeRobot framework. It allows the programming of robot behaviors using automata. The developer creates a graphical representation of the automaton, fills the code to be executed on each state and on each transition, and the tool automatically generates the output JdeRobot component. The use of *hierarchical* automata provides power to the tool to represent complex behaviors. The new release automatically generates Python or C++ components, and the generated component dynamically shows the state of the (hierarchical) automaton in execution while running. In addition the usability of the tool has been improved.

The tool was previously tested with Pioneer, Kobuki and Nao humanoid robots. The new release has also been validated generating two example behaviors for a drone robot.

As future lines we would like to make the generated automaton safely interruptible and to promote the use of visualHFSM among JdeRobot users, to get feedback from them using the tool in different scenarios. In this way, the tool has recently been included in the official release of the framework. In addition, we are exploring the extension of the tool to generate ROS nodes and support for Petri nets as richer robot behavior model.

Acknowledgment

This research has been partially sponsored by the Community of Madrid through the RoboCity2030-III project (S2013/MIT-2748), by the Spanish Ministerio de Economía y Competitividad through the SIRMAVED project (DPI2013-40534-R) and by the URJC-BancoSantander.

References

[1] Borja Menéndez and Rubén Salamanqués and José M. Cañas, *Programming of a Nao humanoid in Gazebo using Hierarchical FSM*. Proceedings of XIV Workshop on Physical Agents, WAF-2013, pp 15-22. ISBN: 978-84-695-8319-7. 2013.

[2] David Yunta and José M. Cañas, *Programación visual de autómatas para comportamientos en robots*. Proceedings of XIII Workshop on Physical Agents, WAF-2012, Santiago de Compostela, 3rd-4th september 2012. pp 65-71, ISBN 978-84-940469-0-2, 2012.

[3] J. M. Cañas and M. González and A. Hernández and F. Rivas, *Recent advances in the JdeRobot framework for robot programming*. Proceedings of RoboCity2030 12th Workshop, Robótica Cognitiva, pp 1-21, UNED, Madrid, July 4, 2013. ISBN:978-84-695-8175-9

[4] Michalis Foukarakis and Asterios Leonidis and Margherita Antona and Constantine Stephanidis, *Combining Finite State Machine and Decision-Making Tools for Adaptable Robot Behavior*, Universal Access in Human-Computer Interaction. Aging and Assistive Environments, Volume 8515 of the series Lecture Notes in Computer Science pp 625-635. Springer International Publishing, 2014.

[5] D C MacKenzie and R C Arkin. *Evaluating the usability of robot programming toolsets*. The International Journal of Robotics Research, 17(4):381, 1998.

[6] Jonathan Bohren and Steve Cousin, *The SMACH High-Level Executive*. IEEE Robotics & Automation Magazine, 17(4), pages 18-20, 2011

[7] Jonathan Bohren, Radu Bogdan Rusu, Edward Gil Jones, Eitan Marder-Eppstein, Caroline Pantofaru, Melonee Wise, Lorenz Mösenlechner, Wim Meeussen, and Stefan Holzer. *Towards Autonomous Robotic Butlers: Lessons Learned with the PR2*. Proceedings of ICRA, pages 5568-5575. IEEE, 2011.

[8] R. Cintas, L. Manso, L. Pinero, P. Bachiller, and P. Bustos. *Robust behavior and perception using hierarchical state machines: A pallet manipulation experiment*. Journal of Physical Agents, 5(1):35–44, 2011.

[9] Gostai studio suite. *http://www.gostai.com/products/studio/gostaistudio/*, 2012.

[10] M. Lötzsch, J. Bach, H.D. Burkhard, and M. Jüngel. *Designing agent behavior with the extensible agent behavior specification language xabsl*. In D. Polani, B. Browning, A. Bonarini, and K. Yoshida, editors, RoboCup 2003: Robot Soccer World Cup VII, volume 3020 of Lecture Notes in Artificial Intelligence. Springer, 2004.

[11] Max Risler. *Behavior Control for Single and Multiple Autonomous Agents Based on Hierarchical Finite State Machines*. PhD thesis, Fachbereich Informatik, Technischen Universitat Darmstadt, 2009.

[12] D. Herrero-Perez, F. Bas-Esparza, H. Martinez-Barbera, F. Martin, C.E. Aguero, V.M. Gomez, V. Matellan, and M.A. Cazorla. *Team chaos 2006*. In Proceedings of the IEEE 3rd Latin American Robotics Symposium, 2006. LARS '06, pages 208–213, 2006.

Author Index