Robótica 2012

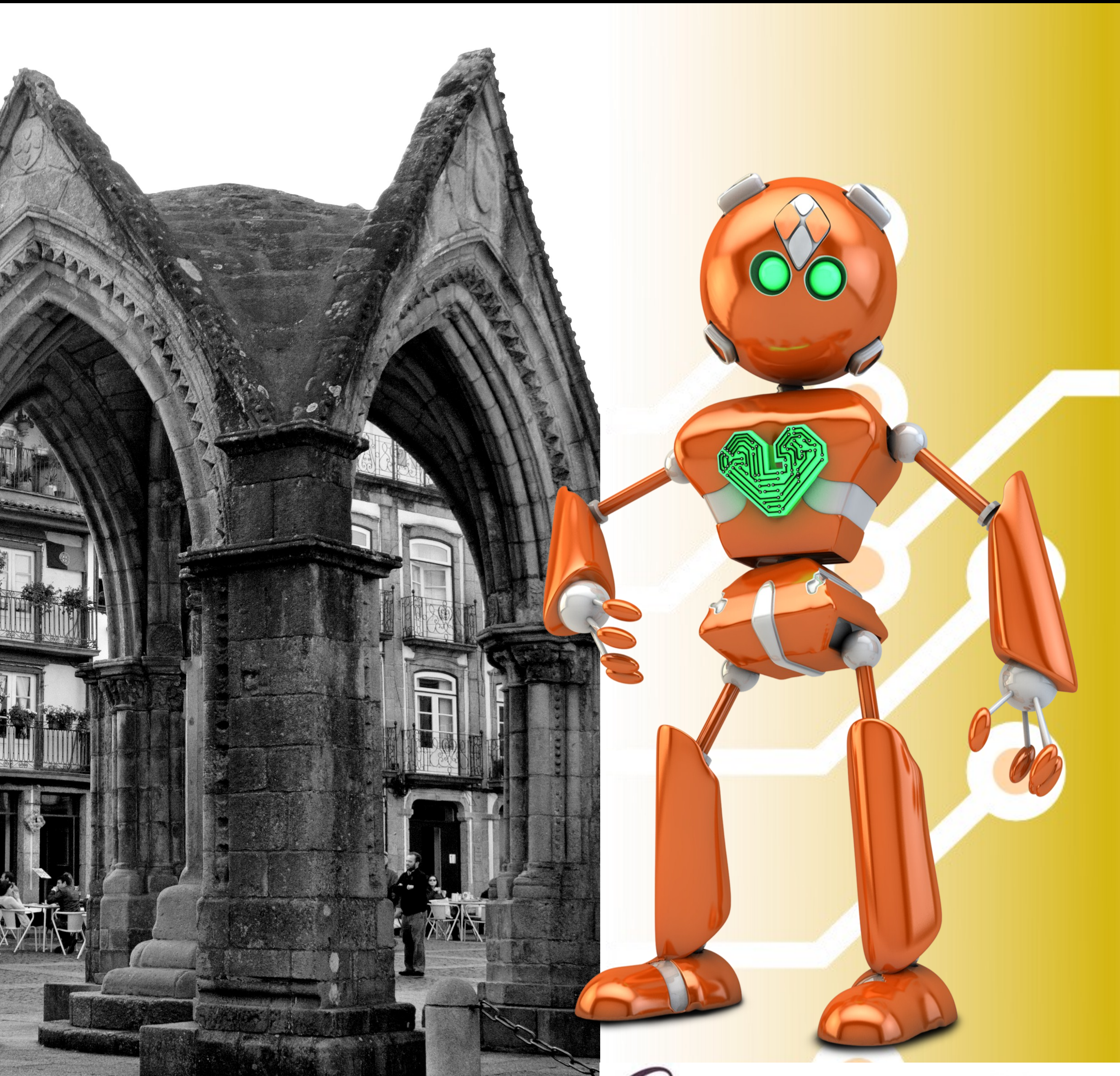
Proceedings of the 12th International Conference on

Autonomous Robot Systems and Competitions

University of Minho

http://www.robotica2012.org

Guimarães, 11th April, 2012



Robotics & Automation Society IEEE Portuguese Chapter

Autonomous Robot Systems and Competitions

Proceedings of the 12th International Conference

April 11, 2012, Guimarães, Portugal

Title: Autonomous Robot Systems and Competitions: Proceedings of the 12th International Conference, April 11, 2012, Guimarães, Portugal
Editors: Estela Bicho, Luís Louro, Fernando Ribeiro
Cover: View of Oliveira Square in Guimarães and Rogui, the official mascot of the event.
Graphical Design of Cover: Emanuel Sousa
ISBN: 978-972-98603-4-8

12th International Conference on Autonomous Robot Systems and Competitions

University of Minho, Guimarães, Portugal

April 11, 2012

Welcome to the 12th International Conference on Autonomous Robot Systems and Competitions. Welcome to the University of Minho and to Guimarães, 2012 European Capital of Culture!

This is the 2012's edition of the scientific meeting of the Portuguese Robotics Open (ROBOTICA'2012). It aims to disseminate scientific contributions and to promote discussion of theories, methods and experiences in areas of relevance to Autonomous Robotics and Robotic Competitions.

A total of 29 contributions were submitted in response to the call for papers. It was possible to accept 15 contributions for oral presentation at the conference. All accepted contributions are included in this proceedings book. The conference program also includes an invited talk by Dr.ir. Raymond H. Cuijpers, from the Department of Human Technology Interaction of Eindhoven University of Technology, Netherlands.

The conference is kindly sponsored by the IEEE Portugal Section.

We would like to thank the invaluable contribution of the IEEE-RAS, sponsors, International Program Committee members, external reviewers, invited speaker, session chair and authors. We also thank and appreciate the collaboration of Ana Maria Pereira from University of Minho, in managing registrations and some other local arrangements. Thank you all for participating in this event and we hope you enjoy the conference, and trust it will be a highly productive and sociable event.

Estela Bicho
Luís Louro
Fernando Ribeiro

Committee

Aníbal Matos	FEUP - PT
António Fernando Ribeiro	UMinho - PT
António Jesus Bandera Rubio	UM - ES
António Paulo Moreira	FEUP - PT
António Pascoal	IST - PT
Bernardo Cunha	UAveiro - PT
Carlos Cardeira	IST - PT
Carlos Carreto	IPG - PT
Daniele Nardi	URoma - IT
Dimos Dimarogonas	KTH-SE
Eduardo Silva	ISEP - PT
Estela Bicho	UMinho - PT
Federica Pascucci	URoma Ű IT
Fernando Lobo Pereira	FEUP - PT
Filipe Silva	UAveiro - PT
Gerhard Kraezschmar	Fh-Bonn-Rhein-Sieg - D
Gil Lopes	UMinho - PT
Helder Araújo	UCoimbra - PT
Hugo Costelha	IST - PT
João Caldas Pinto	IST - PT
João Miguel Sousa	IST - PT
João Sequeira	IST - PT
Johannes du Buf	UALG - PT
Jorge Dias	UCoimbra - PT
Jorge Ferreira	UAveiro - PT
José L. Lima	IPB - PT
José Sá da Costa	IST - PT
José Sá da Costa	IST - PT
José Santos-Victor	IST - PT
José L. Azevedo	UAveiro - PT
José Tenreiro Machado	ISEP - PT
Luís Almeida	FEUP - PT

Table of Contents

Robots that care: a need for decent human-robot interaction

Raymond H. Cuijpers,
Department of Human Technology,
Faculty of Industrial Engineering and Innovation Sciences,
Eindhoven University of Technology
Netherlands

"ROBOTS THAT CARE:
A NEED FOR DECENT
HUMAN-ROBOT INTERACTION"
Dr.ir. Raymond H. Cuijpers
Eindhoven University of Technology

***Abstract*— In an aging population robots could serve many purposes alleviating the care needs of older people and improving their quality of life. However, placing a robot in a domestic environment introduces many unsolved issues. The robot should be able to move about in cluttered and unknown environments, it should be able to approach a person, it should communicate, take context into account, respond and interact, make decisions and so on. Recent insights from cognitive psychology suggest that, on a low level, humans their own motor representations to simulate another person's actions. This would explain why robots are often considered unfriendly, threatening and irritating, because they are fundamentally unpredictable. It is expected more dynamic, interactive robot behaviour that takes this into account will facilitate human-robot interaction as the robot's actions become predictable and therefore understandable and trustworthy.**

A Domain Specific Language for Modeling Differential Constraints of Mobile Robots

Marco Guarnieri, Eros Magri, Davide Brugali, Luca Gherardi

***Abstract*— Kinematics and dynamics constraints of mobile robots can be modeled by means of differential equations. Simulation and sampling based path-planning algorithms need a model of these constraints in order to deal with non-holonomic mobile robots.**

Usually these models are hard-coded in the implementation of those algorithms and this makes hard their reuse. In order to design these algorithms in a modular and extensible way we have to explicitly represent the models of the robots and decouple them from algorithms implementation.

We propose DCML, a Domain Specific Language that can be used in order to describe differential models, and a tool that allows developers to automatically generate the code that implements the model. We also aim to show how this Model-Driven Engineering technique can be used with good results. As a demonstration of what can be done by means of our DSL, we present the differential model of an omnidirectional holonomic robot called BART, and we show how this model can be integrated in a framework for path planning.

I. Introduction

Differential equations are widely used for modeling kinematics and dynamics constraints of mobile robots, for example in simulation and sampling-based path planning algorithms. Differential models express relations between configuration variables. The possible states of a mobile robot are represented in the state space X and each state represents a particular configuration of the robot. For wheeled mobile robots each state $\vec{x} \in X$ is $\vec{x} = (x, y, \theta)$, where x and y represent the position of the robot in the plane and θ is its orientation. In the same way it is possible to define the action space U, which is the set of all the possible actions on all the possible states (an action is a response of the robot, which changes its current state, to an external input). Thus a differential model can be represented as $\dot{\vec{x}} = f(\vec{x}, \vec{u})$ where $\vec{x} \in X$ is the starting state, $\vec{u} \in U$ is the action applied to the model and f is a function, called state transition function, that defines the relation between state space and action space [1, Ch. 13]. The results are expressed in terms of velocities $\dot{\vec{x}}$ and the outcome of their integration represents the future states that satisfy the kinematics constrains.

These models are widely used in simulation algorithms (that, given the starting configuration and the action vector, compute the final configuration of the robot) and sampling-based motion planning algorithms (that sample collision free configurations that need to be compatible with the differential constraints). In order to compute final configurations it is necessary to solve the differential equations and this can be done by means of solvers, which use numerical approximation techniques. However solvers require that differential equations are implemented in the source code fulfilling specific interfaces, and implementing these equations is usually error prone and not trivial. Another problem is that differential models are usually hard-coded in the implementation of these algorithms, hence the algorithm implementation is hard-coupled to the specific robot.

In order to achieve higher flexibility, modularity, easier extensibility with respect to the current situation and to solve the problems presented before, a higher level representation of those models is needed. Domain specific languages (DSLs) provide this higher level representation. DSLs are simple formal languages, usually declarative, used to represent domain specific knowledge using some sort of syntax. DSLs let you describe easily a scenario in a specific domain.

The syntax and semantic of these DSLs are designed explicitly to describe only the knowledge of a specific domain and thus DSLs have a gain in terms of expressiveness and ease of use compared to general purpose languages for the specific domain. Conversely they are usually less expressive than general-purpose programming languages out of their domain. In this way they can also improve productivity and maintenance costs. More details on DSLs can be found in [2] and [3].

DSLs have also other advantages over general-purpose languages while expressing knowledge in the specific domain:

- being less expressive and complex than general purpose languages, DSLs can be used also by people that are not expert programmers;
- manual implementation of the model can require experience in computer programming and it is error prone while you can usually generate code from a DSL document in an in automated way;
- the model itself can be used as documentation;
- ease the communication between programmers and domain experts;
- ease the description of the scenario;
- can decouple the representation of the model from the technologies and interfaces used in the implementation.

In this paper we present DCML, Differential Constraints Modeling Language, a Domain Specific Language that allows the description of such kind of models with a high level of abstraction from implementation details. DCML, in addition to the advantages presented above, provides devel-

M. Guarnieri, E. Magri, D. Brugali and L. Gherardi are with the Dept. of Information Technology and Mathematics, University of Bergamo, 24044 Dalmine, Italy `0guarnieri.marco0@gmail.com`, `erosmagri@gmail.com`, `brugali@unibg.it`, `luca.gherardi@unibg.it`

opers with an automatic way to generate the code that implements the differential equations starting from the differential model, which describes the relations between state space and action space of a specific robot. This implementation can then be used by motion planning algorithms in order to do the simulation of the behavior of the robot itself. In order to develop DCML, we have followed a Model Driven Engineering (MDE) approach. MDE has already shown good results in robotics in terms of reusability and integration, as shown in [4] or [5], and thus we aim to demonstrate that this approach can be applied with good results also to the representation of differential models.

Section II presents some related work. Section III describes DCML more in detail, while Section IV shows how our language can be used for writing the differential model of an omnidirectional holonomic robot and how the generated code can be integrated in a framework for simulation and planning for mobile robots. Finally Section V presents our conclusions.

II. Related Work

Despite our research and this paper focus only on the use of differential models in sampling-based motion planning algorithms for the simulation of robots behaviour in response to specific actions, differential models are widely used also in other robotics fields. They can be used to describe several kinds of robots: [1] and [6] present differential models of some wheeled mobile robots under kinematics and dynamics constraints, while [7] shows a model of an hexapod robot. An extension of differential models, that can take into account also dynamics constraints, are phase-space models that consider also accelerations and can then be described as $\ddot{x} = f(\dot{x}, x, u)$. Each second order model can be converted in a first-order model, which is a differential model, using a *phase space*, that has more dimension than the *state space* of the second order model. In this way we can represent, using differential models, also dynamic constraints. In the same way a kth-order model can be expressed as a differential model using an adequate *phase space*. Thus differential models can be used for motion planning under kinematics and dynamics constraints, as shown in [1, Ch. 14].

A first way of defining differential models with an higher level of abstraction than hardcoded solutions is using Simulink [1]. It provides developers with a toolchain for defining, through block diagrams, differential models and generating from these diagrams C and C++ code that implements them. In our opinion this approach is not flexible enough because it does not allow the generation of code in other programming languages and also because it does not allow developers to customize the generated code, in terms of interface and optimization.

Another approach is using a dedicated Domain Specific Language. Literature presents, up to now, a few DSLs to describe differential equations. The MyFEM language, presented in [8], is a DSL for the definition of partial-differential equations using a subset of the Python language. It allows the generation of C++ code that implements the model defined in MyFem but it has not got an IDE. Scalation [9] is an embedded DSL defined over the Scala programming language, and it has a package that allows the representation of systems of differential equations. These approaches use subsets of existing programming languages to define the DSLs. This has some advantages, such as less learning time, but it has also the big drawback that the resulting DSL is too close to the general purpose language and thus it has less abstraction than a dedicated DSL and requires too much effort to be used by users that are not expert programmers. Another drawback of both approaches is that the syntax used to express differential equations is too far from the mathematical formalism because it is tied to the syntax of native programming languages.

Other approaches, such as the one in [10] that defines a specification language for partial differential equations on a union of rectangles, or [11] that defines differential equations using the arrow notation, are, in our opinion, too complex and difficult in order to be used as an effective aid to developers. A common disadvantage of all these approaches is that they are tied to work only with a fixed set of numerical solvers.

Given the fact that existing solutions for representing differential equations are too complex or do not offer enough flexibility in the code generation phase we decided to create a new DSL for representing differential models. DCML offers two advantages with respect to existing solutions. Firstly the syntax used to describe differential equations is close to the mathematical one, and secondly DCML is not tied to work with a fixed set of differential equations solvers.

III. Differential Constraints Modeling Language

DCML allows users to describe constraints that affect mobile robots by means of differential models. In this way users can focus on the description of the differential equations.

A simple model, taken from [1], that can be used to describe the constraints of a differential drive, a mobile robot with two independent wheels, is presented in 1.

$$\begin{aligned}\dot{x} &= \frac{r}{2}(u_l + u_r)cos\theta \\ \dot{y} &= \frac{r}{2}(u_l + u_r)sin\theta \\ \dot{\theta} &= \frac{r}{L}(u_r - u_l)\end{aligned} \tag{1}$$

The state vector (x, y, θ) represents the cartesian position of the robot while the action vector $u = (u_l, u_r)$ represents angular velocities of the wheels, r is the radius of each wheel while L is the distance between the two wheels.

Listing 1 shows the DCML document representing the differential drive presented above, and it will be used to describe how our language works. It describes the model of the differential drive presented above. A document written

[1]Simulink - www.mathworks.com/products/simulink/

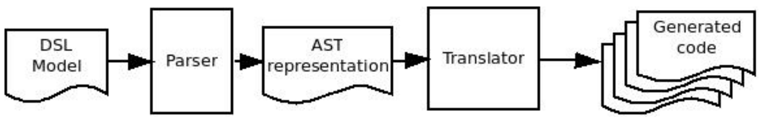


Fig. 1. Validation and Code generation process

by means of our DSL can describe several models and for each model the user can specify:

- The action space: after the keyword **ACTION** the user can specify all the actions. In the differential drive example we have two actions u_l and u_r.
- The configuration space: after the keyword **CONFIG** the user can specify the dimensions of each configuration. In the example each configuration can be expressed in terms of $x,\ y,\ \theta$.
- After the keyword **PARAM** the user can define the parameters of the model. In our example they are r and L.
- The state transition function of the model can be expressed by means of differential equations in an understandable way.
- After the keyword **VAR** the user can define some temporary variables that can be used to ease the definition of differential constraints.
- After the keyword **CONST** the user can define some constant values, different from the predefined ones, such as π and e.
- After the keyword **PACKAGE** the user can define the package in which the source code will be created. In the example we have decided that we want to create the source code in the package *robotics.models*.
- If the model definition isn't expressive enough, further comments can be added with a JavaDoc style notation.

```
BEGIN DifferentialDrive
 PACKAGE : robotics.models;
 ACTION : u_l, u_r;
 PARAM : L, r;
 CONFIG : x, y, theta;

 d(x) = r / 2 * (u_l + u_r) * cos(theta);
 d(y) = r / 2 * (u_l + u_r) * sin(theta);
 d(theta) = (r / L) * (u_r − u_l);
END;
```

Listing 1. Differential Drive model

While actions, configurations and differential equations are mandatory, the other elements are useful only for describing more complex models (see Section IV). We will describe, now, the structure of the grammar of our DSL, that is shown in Listing 2. A grammar has four main components, [12]:

1) a set Σ of terminals, which are the basic symbols that form valid instructions of our language;
2) a set V of non-terminals, that are syntactic variables that represent set of strings;
3) a non-terminal $s \in V$ that acts as start symbol;
4) a set P of productions, that define how terminals and non-terminals can be combined in order to generate valid strings.

For our grammar the set Σ is equal to {"**", "*\", *"BEGIN"*, *"END"*, *";"*, *"PACKAGE"*, *":'"*, *"ACTION"*, *"PARAM"*, *"CONST"*, *"CONFIG"*, *"VAR"*, *","*, *"+"*, *"-"*, *"*"*, *"\"*, *"("*, *")"*, *"d("*, *ID*, *PCKG_ID*, *NUM*, *COMMENT*} where ID represents an alphanumerical identifier, NUM is a numeric literal and $PCKG\_ID$ is a package identifier. The set V is composed by {*modelList, model, package, actions, params, constants, configurations, variables, varList, constList, varDef, constDef, assignments, assignment, var, expr, term, factor, paramList*} and the start symbol is $modelList$.

The grammar is expressed using the Extended Backus-Naur Form (EBNF) [13] that describes each production in the form $A \rightarrow f(V_1, \ldots, V_n, \alpha_1, \ldots, \alpha_m)$ where $A, V_1, \ldots, V_n \in V$, $\alpha_1, \ldots, \alpha_m \in \Sigma$ and f is a function that concatenates symbols using regular expressions. A production means that the non-terminal on the left hand side can be replaced by the regular expression on the right hand side of the $\rightarrow$ operator.

```
 1 modelList −> model(model)*
 2 model −> ["/**" COMMENT "*/"] BEGIN ID [package]
       actions [params] [constants] configurations
       [variables] assignments END";"
 3 package −> PACKAGE ":" PCKG_ID";"
 4 actions −> ACTION ":" varList ";"
 5 params −> PARAM ":" varList ";"
 6 constants −> CONST ":" constList ";"
 7 configurations −> CONFIG ":" varList ";"
 8 variables −> VAR ":" varList ";"
 9 varList −> varDef ("," varDef)*
10 constList −> constDef ("," constDef)*
11 varDef −> ID
12 constDef −> ID "=" ("+" | "−")NUM
13 assignments −> assignment (assignment)*
14 assignment −> var "=" expr";"
15 var −> ID | "d(" ID ")"
16 expr −> term ( ("+" | "−")term )*
17 term −> factor ( ("*" | "/")factor )*
18 factor −> NUM | "(" expr ")" | var["("paramList")"]
19 paramList −> expr("," expr)*
```

Listing 2. DSL Grammar

The first production (row 1) involves the $modelList$ terminal, and means that a document of our DSL must contain at least one model. The second production describes the syntax of each model, it must be enclosed between a *"BEGIN"* instruction and an *"END"* instruction and the ID must be unique in the document. Symbols enclosed between square brackets are optional. In the differential drive example the ID is *DifferentialDrive*.

The productions at rows 4,5,7,8 define that, after the specific keywords, a list of variable declarations is needed. These lists represent, respectively, actions, parameters, configurations, and temporary variables. Each variable list, represented

```
package robotics.models;

public class DifferentialDrive implements IFirstOrderModel {

        private double L, r, u_l, u_r;

        public void setAction(double[] actions) {
                if (actions.length != 2)
                        throw new IllegalArgumentException("Actions must have size 2.");
                u_l = actions[0];
                u_r = actions[1];
        }
        public void setParameters(double[] parameters) {
                if (parameters.length != 2)
                        throw new IllegalArgumentException("Parameters must have size 2.");
                L = parameters[0];
                r = parameters[1];
        }
        public void computeDerivatives(double t, double[] y, double[] yDot) throws DerivativeException{
           yDot[0] = r / 2 * (u_l + u_r) * java.lang.Math.cos(y[2]);
                yDot[1] = r / 2 * (u_l + u_r) * java.lang.Math.sin(y[2]);
                yDot[2] = (r / L) * (u_r − u_l);
        }
}
```

Listing 3. Differential Drive implementation

by the non-terminal $varList$, is made up of one or more variable declarations (row 9), each one consisting in an ID, as shown in the production at row 11, that must be unique in the model. In a similar way the production that has as head the non-terminal $constants$ (row 6) defines that, after the keyword *"CONST"*, a list of constant declarations $constList$ (row 10) is needed. Each constant declaration is made up of an ID, unique in the model, and a numeric literal, as shown in the production 12.

The non-terminal $assignments$ can be replaced by a list of differential equations. Each equation is defined as $var = expr$, where var is a non-terminal that represents, as shown in production 15, either a differential variable of the first order, or an already defined identifier. $expr$ represents an algebraic expression composed by predefined functions, such as sin or cos, parenthesized expressions, numeric literals, predefined constants, such as π, or instances of the var non-terminal and also the usual mathematical operators $+, -, *, /$.

In order to validate and generate the code that implements the models expressed using our DSL we have defined the process shown in Figure 1. It can be divided in two phases. In the first one, the parsing phase, the document is validated. The parser checks that the document is correct, both from a syntactic point of view (it must respect the syntactic rules) and also from a semantic point of view (e.g. the parser checks that the document does not contains undeclared variables, non unique identifiers or function invocations with a wrong number of parameters). In this phase the parser builds, starting from the document, the Abstract Syntax Tree (AST) that is an intermediate representation of the model. The AST is a tree representation of the syntactic structure of the model enriched with some useful semantic information elaborated during the parsing phase.

Taking the AST as input, we can start the second phase, i.e. the translation phase, which creates the code that implements the differential model by means of a general purpose programming language. This can be done by simply visiting the AST, because it is a tree structure bearing all the information needed for the translation.

The decision of generating an intermediate representation by means of ASTs, instead of performing directly the translation during the parsing phase has some advantages:

- allows the validation of the model without performing the translation;
- by decoupling the translation form the parsing phase we can develop and use several translators, which target several programming languages and/or numerical solvers, without modifying the parser. This is possible because the parsing phase is completely separated by details regarding the generation of the code.

Despite this solution is a bit less efficient than performing the translation during the parsing phase, it has great advantages in terms of extensibility and flexibility. The Java code generated from the differential drive example is shown in Listing 3. This code is written to be compatible with numerical solvers provided by the *Apache Commons Math* library[2]. The generated class implements the interface *IFirstOrderModel*, which extends the interface *FirstOrderDifferentialEquation* defined in the *Apache Math* library. It defines three methods: 1) *computeDerivatives*, called by the solver, contains the definition of the state transition function (the state $\vec{x}$ is mapped on the array y, while the velocities $\dot{\vec{x}}$ are mapped on $yDot$), 2) *setParameter* that can be used to set the parameters of a specific robot, 3) *setAction* that can be used to set the actions. The methods *setParameter* and *setAction* are called by the simulator.

Using our DSL users can focus only on modeling the state transition function of the robot. The code can be

[2]Apache Commons Math - http://commons.apache.org/math/

automatically generated by translators optimized accordingly to both the destination programming language and the model interface. In this way the details related to the model implementation (e.g. the numerical solver used to solve the equations) can be completely hidden to the user and the task of creating optimized code can be delegated to the writer of the translator. Our approach allows developers to define new translators in order to generate code optimized accordingly to real-time and computational requirements.

The grammar of our DSL was defined by using AntLR3[3]. We used it also for the definition of the semantic actions and for building the AST tree. AntLR is a parser generator that reduces the time and effort needed to build and maintain language processing tools.

One of the problems of MDE is that developers, in order to use MDE techniques, usually need tools that support them in the management and development of models. Thus, in order to create such a tool we have decided to use the Xtext framework[4], which provides a simple way for creating textual DSLs and to automatically generate a full-featured Eclipse Text Editor from the grammar. The grammar implemented in the Xtext editor is the same used for the parser, without semantic actions.

We choose to implement the parser separately from the editor for two reasons. First, thanks to AntLR we can have better control on the definition of the syntax and the semantic of our language and on the AST creation phase than using Xtext. Second, by using AntLR we have created a tool that can be used also in a stand-alone way, or can be integrated in others IDEs.

In this way we can use Xtext in order to integrate our DSL in the Eclipse IDE, that is by now one of the de facto standards in terms of IDEs and has several plugins related to model driven engineering. This integration gives to developers useful features such as auto-completion and syntax highlighting, while expressing the grammar using AntLR give us the power of expressing complex semantic rules.

In order to integrate the parser in the editor we have added a button that allows the invocation of the parser, which takes the model as input and validate it. Then, in the case that the model is correct, the Java translator is invoked. It takes the AST produced by the parser and translate it into the Java class that implements the differential equations of the model and fulfills the interface for differential models.

IV. Case Study

Using DCML we can describe models of simple robots, such as the differential drive described in Section III, or models of more complex robots like BART.

BART is an omnidirectional holonomic wheeled robot, developed by the Software for Experimental Robotics Lab (SERL) at the University of Bergamo. It is made up of two steering blocks and two free wheels. Each block is a differential drive and the rotational joint is not on the axis of its wheels. The mechanical structure of the robot is presented in Figure 2. BART is slightly similar to the robots presented in [14] and [15].

As a case of study we will show how the kinematic model of BART can be expressed by means of our DSL. While modeling the kinematics of BART robot defining the rigid bodies that made it up can be quite difficult, modeling the general kinematics of the robot can be done quite easily using differential models, and thus in our DSL, as shown in Listing 4.

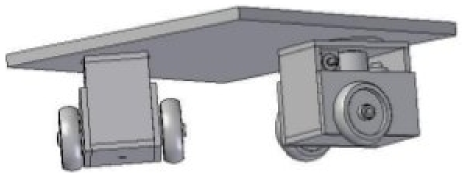


Fig. 2. BART robot

We choose to use as configuration space of the BART robot the variables x, y and $theta$ (cartesian position and the orientation of the center of the robot), the variables x_front, y_front and phi_front (cartesian position and orientation of the joint that connects the base of the robot to the frontal differential drive, related to the absolute reference system), and the variables x_rear, y_rear and phi_rear (position and orientation of the rear steering block).

The parameters of the model are $tt\_wheel\_half\_axis$, that represents the half length between the wheels in one of the differential drives, tt_wheel_radius, that is the radius of each wheel of the steering blocks, and tt_steer_offset, that is the distance between the steering axis and the axis of the wheels of the robot. We introduce a variable k that represents the ratio between tt_steer_offset and $tt\_wheel\_half\_axis$ to simplify the writing of the differential equations. The actions accepted by the robot are the values $left\_f\_s$ and $right\_f\_s$, that represent the angular speeds of left and right wheels of the frontal steering block, and the values $left\_r\_s$ and $right\_r\_s$, that represent the angular speeds of left and right wheels of the rear differential drive.

The differential model of BART can be divided in three parts. The first one, that involves x_front, y_front and phi_front variables, expresses the differential equations needed to compute the position of the joint of the front steering block. It is an extension of the differential model for a standard differential drive that considers also the fact that the rotational joint is not on the axis of the differential drive. The second part, involving variables x_rear, y_rear and phi_rear, is quite similar to the first one but it is related to the rear steering block. The last part of the model, involving variables x, y and $theta$, computes the position of the center of the BART robot. This part is not made up of differential equations because these values can be computed with algebraic equations from the values of the two steering blocks.

We developed a Java framework that implements some well known algorithms for sampling-based path planning.

[3]ANother Tool for Language Recognition - http://www.antlr.org/

[4]Xtext - http://www.xtext.org/

```
BEGIN Bart
 PACKAGE : robotics.models;
 ACTION : left_f_s, right_f_s, left_r_s, right_r_s;
 PARAM : tt_wheel_half_axis, tt_wheel_radius, tt_steer_offset;
 CONFIG : x, y, theta, x_front, y_front, phi_front, x_rear, y_rear, phi_rear;
 VAR : k;

 k = tt_steer_offset / tt_wheel_half_axis;

 d(x_front) = (tt_wheel_radius /2) * (((cos(phi_front) − k * sin(phi_front)) * right_f_s) + ((cos(phi_front
     ) + k * sin(phi_front)) * left_f_s));
 d(y_front) = (tt_wheel_radius /2) * (((sin(phi_front) + k * cos(phi_front)) * right_f_s) + ((sin(phi_front)
     − k * cos(phi_front)) * left_f_s));
 d(phi_front) = (tt_wheel_radius / (2 * tt_wheel_half_axis)) * (right_f_s − left_f_s);

 d(x_rear) = (tt_wheel_radius /2) * (((cos(phi_rear) − k * sin(phi_rear)) * right_r_s) + ((cos(phi_rear) + k
     * sin(phi_rear)) * left_r_s));
 d(y_rear) = (tt_wheel_radius /2) * (((sin(phi_rear) + k * cos(phi_rear)) * right_r_s) + ((sin(phi_rear) − k
     * cos(phi_rear)) * left_r_s));
 d(phi_rear) = (tt_wheel_radius / (2 * tt_wheel_half_axis)) * (right_r_s − left_r_s);

 x = (x_front + x_rear)/2;
 y = (y_front + y_rear)/2;
 theta = atan2((y_front−y_rear),(x_front−x_rear)) + (pi / 4);
END;
```

Listing 4. BART model

All these algorithms depend on the model of the robot under simulation and thus, the differential model is, for all of them, an input parameter. Using DCML we can modify the implementation of the differential model without changing directly any line of source code. We simply have to modify the DCML document and regenerating from it the new code. In order to do this we have developed a DCML to Java translator that, taken as input the AST of the model, creates the class that implements the model itself.

V. Conclusions

In this paper we have shown how, using a Domain Specific Language, it is possible to describe the differential model of a mobile robot. We have also shown how an intermediate representation of the model by means of an AST is useful in order to use translators that can generate optimized code for any target platform. In this way the model is independent from the actual implementation. We have also presented the tool suite that can be used in order to define and generate implementations of DCML models.

This work shows that MDE can be used with good results in specific areas, such as the representation of differential models, in which the representation of the knowledge can be formalized in a defined model. The integration of the generated models in the path planning framework demonstrates that the technique can have useful application also in a real environment.

Appendix

The DCML Eclipse Tool and some example can be found at *http://robotics.unibg.it/software/dcml/*.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. FP7-ICT-231940-BRICS (Best Practice in Robotics).

References

[1] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[2] Arie van Deursen, Paul Klint, and Joost Visser. Domain-specific languages: an annotated bibliography. *SIGPLAN Not.*, 35, 2000.

[3] M. Mernik, J. Heering, and A.M. Sloane. When and how to develop domain-specific languages. *ACM Computing Surveys (CSUR)*, 37(4), 2005.

[4] C. Schlegel, T. Haßler, A. Lotz, and A. Steck. Robotic software systems: From code-driven to model-driven designs. In *Advanced Robotics, 2009. ICAR 2009. International Conference on*. IEEE, 2009.

[5] Christian Schlegel, Andreas Steck, Davide Brugali, and Alois Knoll. Design abstraction and processes in robotics: from code-driven to model-driven engineering. In *Proceedings of the Second international conference on Simulation, modeling, and programming for autonomous robots*, SIMPAR'10. Springer-Verlag, 2010.

[6] J.P. Laumond, S. Sekhavat, and F. Lamiraux. Guidelines in nonholonomic motion planning for mobile robots. *Robot motion planning and control*, 1998.

[7] E. Szádeczky-Kardoss and B. Kiss. Extension of the rapidly exploring random tree algorithm with key configurations for nonholonomic motion planning. In *Mechatronics, 2006 IEEE International Conference on*. IEEE, 2006.

[8] J. Riehl. Implementing the myfem embedded domain-specific language. In *Proceedings of the Second International Workshop on Domain-Specific Program Development*, DSPD'08, 2008.

[9] John A. Miller, Jun Han, and Maria Hybinette. Using domain specific language for modeling and simulation: Scalation as a case study. In *Winter Simulation Conference*, 2010.

[10] M.H. Hohn. A little language for modularizing numerical pde solvers. *Software: Practice and Experience*, 34(9), 2004.

[11] H. Liu and P. Hudak. An ode to arrows. *Practical Aspects of Declarative Languages*, 2010.

[12] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: principles, techniques, and tools*. Addison-Wesley Longman Publishing Co., Inc., 1986.

[13] N. Wirth. Extended backus-naur form (ebnf), 1996. *ISO/IEC*, 14977, 1996.

[14] Fuhua Han, Takaaki Yamada, Keigo Watanabe, Kazuo Kiguchi, and Kiyotaka Izumi. Construction of an omnidirectional mobile robot platform based on active dual-wheel caster mechanisms and development of a control simulator. *J. Intell. Robotics Syst.*, 29, 2000.

[15] Takaaki Yamada, Keigo Watanabe, Kazuo Kiguchi, and Kiyotaka Izumi. Dynamic model and control for a holonomic omnidirectional mobile robot. *Auton. Robots*, 11, 2001.

Towards a Mobile Three-dimensional Modelling System for Underground Structures

António Ferreira, José Almeida and Eduardo Silva
INESC TEC - INESC Technology and Science
(formerly INESC Porto) and ISEP/IPP - School
of Engineering, Polytechnic Institute of Porto

***Abstract*—This paper addresses the three-dimensional modelling of large scale underground galleries, such as traffic tunnels and mines. This work employs techniques from mobile robotics to achieve an autonomous mobile modelling system, adapted to general underground environments. So far, the state-of-the-art methods in underground modelling remain restricted to environments in which pronounced geometric features are abundant. This limitation is a consequence of the scan matching algorithms used to solve the localization and registration problems.**

This work aims to extend the modeling capability to structures characterized by uniform geometry and smooth surfaces, as is the case of road and train tunnels.

A visual monocular Simultaneous Localization and Mapping (MonoSLAM) approach based on the Extended Kalman Filter (EKF) and complemented by the introduction of inertial measurements in the prediction step, allows our system to build three-dimensional models and localize himself over long distances, using exclusively sensors carried on board a mobile platform.

By feeding the Extended Kalman Filter with inertial data we were able to overcome the major problem related with MonoSLAM implementations, known as scale factor ambiguity, which emerges from the absence of metric measurements in monocular images. The monocular visual features used in MonoSLAM were extracted by the SIFT algorithm, and inserted directly in the EKF mechanism according to the Inverse Depth Parametrization. Through the 1-Point RANSAC (Random Sample Consensus) wrong frame-to-frame feature matches were rejected.

To build the model, vertical cross-sections of the gallery, acquired by a laser range finder sensor, are placed on a common reference frame using the estimated localization.

The system was tested based on a dataset acquired inside a real road tunnel. Results from the localization strategy and the modelling process are presented.

I. Introduction

Over the last few years some successful underground mobile modelling implementations were documented [1] [2] [3]. These approaches, designed specifically to operate in mines, are characterized by one common aspect: they all use laser range finder sensors as the main (and in some cases the only) source of information. The model is built by placing laser range finder scans in a virtual three-dimensional world, process called registration. For this purpose, relative position and orientation between scans have to be determined. In previous approaches, this task is accomplished via a scan matching algorithm [7], which restricts the systems to non-uniform structures, since this technique requires that notorious and well-differentiated geometric features stand out along overlapping scans.

Our work extends the underground mobile modelling systems to galleries characterized by very uniform and smooth surfaces, as is the case of traffic tunnels. In this type of scenario the scan matching approaches are condemned to failure, so the previous state-of-the-art systems become ineffective. Without artificial landmarks and no access to global positioning systems, self-localization becomes an hard problem. In inertial based localization the errors accumulated over time cause a monotonic growth in localization uncertainty. On the other hand, a vision based approach may be affected by the lighting conditions, additionally, the parametrization of landmarks far from the cameras raises extra difficulties due to the depth uncertainty.

Similarly to [3], our solution uses 2D laser range finders to gather a sequence of vertical scans along the gallery. Absolute position and orientation of each scan is computed by an independent localization process, that estimates the systems' trajectory based on inertial measurements and a sequence of images.

We employ an alternative localization solution to overcome both the structural monotony and the lack of global positioning systems, adopting the SLAM (Simultaneous Localization and Mapping) concept [8] [9] to estimate the platforms localization in 6DoF (Six Degrees of Freedom). Following the traditional approach, the probabilistic SLAM algorithm is based on the EKF (Extended Kalman Filter). Since for landmarks far from the cameras, stereoscopic systems do not provide satisfactory depth measurements, a visual monocular algorithm was implemented instead, ensuring tracking of landmarks at any depth.

In order to identify visual landmarks to be used in the SLAM algorithm, highly distinctive visual features, invariant to scale, rotation and linear illumination variations, are extracted from the images using the SIFT algorithm [11]. To each feature is assigned at least one descriptor, that embodies the image properties in the features' neighborhood. The descriptors are used to establish the frame-to-frame feature matches.

Our system combines another advanced state-of-the-art methods such as Inverse Depth Parametrization [5], and the 1-Point RANSAC algorithm [6], for outlier rejection.

Through the Inverse Depth Parametrization, undelayed initialization of landmarks within the EKF framework becomes possible. However another major problem of monocular SLAM applications still needs to be solved. A single camera

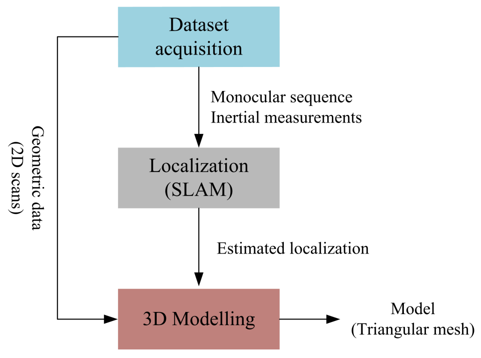
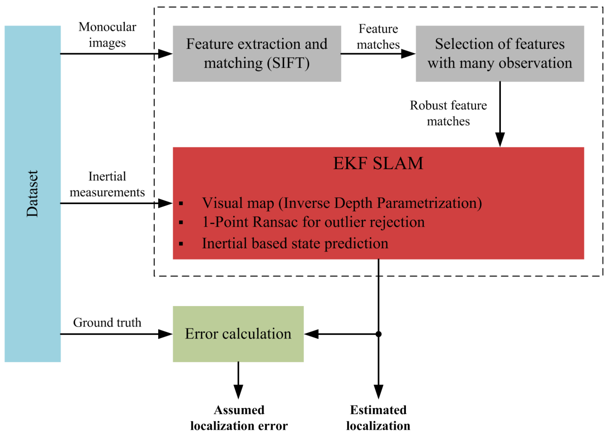


Fig. 1: High level system architecture



Dataset
Monocular images
Feature extraction and matching (SIFT)
Feature matches
Selection of features with many observation
Robust feature matches
EKF SLAM
Inertial measurements
Visual map (Inverse Depth Parametrization)
1-Point Ransac for outlier rejection
Inertial based state prediction
Ground truth
Error calculation
Assumed localization error
Estimated localization

Fig. 2: Localization algorithm overview

moving through the scene does not provide metric measurements, leading to scale ambiguity in the estimated map and motion. As suggested in [4] inertial measurements, provided by a low-cost IMU, feed the filter with metric data in order to prevent the scale factor degeneration. This strategy keeps the map and motion estimates constrained to the meaningful metric system, in our case for distances over more than one hundred meters.

To build the model, all vertical cross sections are placed on a common reference frame according to the localization estimates, resulting in a point cloud model, which is finally converted into a triangular mesh through the Ball Pivoting Algorithm [10], to reach a more realistic representation without information losses.

This document is organized as follows: Section II presents a brief architecture description with emphasis on the localization and modelling algorithms. Section III is devoted to the dataset acquisition that takes places inside a road tunnel. We then present and discuss our implementation results (Section IV) and finally, Section V, provides a conclusion and sets some future goals.

II. System Architecture

Our system is divided in three main blocks, executed by the following order: data acquisition, localization and three-dimensional modelling (see Fig. 1).

In the first step, a sensor platform mounted on board a car is used to collect a wide range of synchronized measurements inside the underground galleries, including images captured by two CCD cameras, 2D scans from two laser range finders and inertial measurements provided by a low cost inertial measurement unit. The platform carries also a INS/GPS system that gives accurate ground truth information, used to measure the performance of our localization strategy.

The localization estimation and modelling tasks are preformed offline based on this data, according to the methods described next.

A. Localization Algorithm

In underground galleries it is expected to find reliable visual features that can be used as reference points to build the SLAM map. The process starts with a feature pre-selection stage (see Fig. 2) to fulfill the following objectives:

- Reduce the computational complexity of the SLAM cycle, by performing feature extraction and frame-to-frame matching in advance. The feature extraction is accomplished by the SIFT algorithm [11], that produces descriptors invariant to scale, orientation, and linear illumination changes, used to compute the frame-to-frame feature matches;
- Identify features with large number of observations and use only those to build the map. By doing so, we pretend to minimize the computational demands, ensuring that all landmarks in the map persist over an acceptable frame interval.

1) State Vector: The SLAM cycle is implemented according to the EKF method. The state vector stores the localization and map states. Since the system does not have prior information about the environment, the initial state vector includes only 9 states related to the platforms' localization: position x^n, orientation Θ^n (expressed in terms of Euler angles) and velocity v^n, all defined in the local level reference frame.

$$x(k) = (x_b)^n(k) = \begin{bmatrix} x^n(k) \\ \Theta^n(k) \\ v^n(k) \end{bmatrix} \tag{1}$$

As new landmarks are observed, the state vector is expanded to accommodate the respective states (equation 2).

$$x(k) = \begin{bmatrix} (x_b)^n(k) \\ L_1(k) \\ L_2(k) \\ \vdots \\ L_n(k) \end{bmatrix} \tag{2}$$

Initially, each landmark L_i is coded in the SLAM map using the Inverse Depth Parametrization [5], which requires six parameters (Fig. 3): position of the cameras' optical center at the moment of first observation $[x_i^n \; y_i^n \; z_i^n]$, azimuth θ_i and elevation ϕ_i angles of the projection ray that passes through the optical center and the landmark, and finally the inverse of the distance ρ_i between the optical center and the landmark in the world (inverse depth).

$$L_i = [x_i^n,\ y_i^n,\ z_i^n,\ \theta_i,\ \phi_i,\ \rho_i]^T \tag{3}$$

The state uncertainty of this overparameterized representation can be modelled by Gaussian distributions, regardless to the distance between the landmark and the camera, therefore this is an efficient and accurate solution for undelayed initialization of new landmarks within the EKF. The EKF computational complexity grows quadratically with respect to the state vector dimension, so when the uncertainty in the landmark's location reveals a Gaussian behavior, indicated by the linearity index introduced in [13], the conversion to the standard Cartesian representation is accomplished applying the formula below:

$$\begin{bmatrix} L_{xi} \\ L_{yi} \\ L_{zi} \end{bmatrix} = \begin{bmatrix} x_i^n \\ y_i^n \\ z_i^n \end{bmatrix} + \frac{1}{\rho_i} m(\theta_i, \phi_i) \tag{4}$$

being $[L_{xi},\ L_{yi},\ L_{zi}]$ the Cartesian coordinates of the landmark and $m(\theta_i, \phi_i)$ a unitary vector calculated from the azimuth and elevation angles:

$$m(\theta_i, \phi_i) = \begin{bmatrix} -cos(\phi_i)sin(\theta_i) \\ sin(\phi_i) \\ cos(\phi_i)cos(\theta_i) \end{bmatrix} \tag{5}$$

2) Landmark Initialization: From the six parameters that define an Inverse Depth landmark, only the azimuth and elevation angles need to be computed, since the camera position is already defined in the state vector, and the initial inverse depth consists on a fixed value defined in advance. To compute the angles, the feature is first projected from the image to the camera reference frame, using the pinhole camera model. A distortion model is applied next to compensate for the lens distortion. From this operation results a three-dimensional non-unitary vector h^c with the same orientation as the projection ray. The vector expressed in the navigation frame is given by:

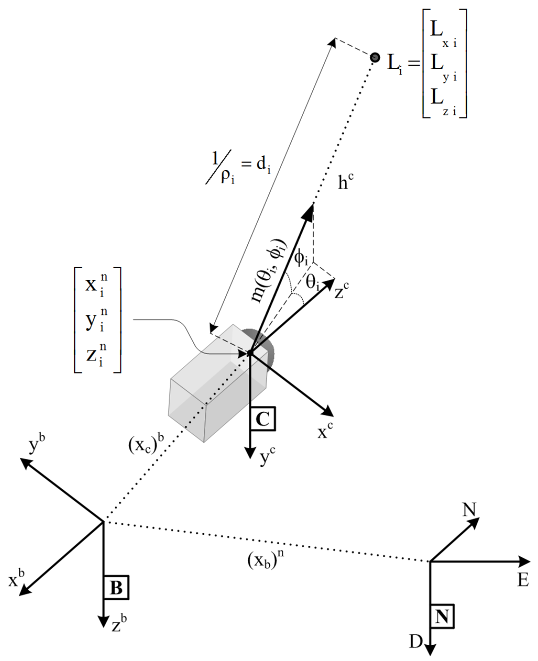
$$h^n = C_b^n C_c^b h^c \tag{6}$$

where C_b^n and C_c^b are the rotations matrices from the body frame to the navigation frame and from the camera frame to the body frame, respectively.

From h^n, the orientation angles can be finally computed as follows:

$$\begin{bmatrix} \theta_i \\ \phi_i \end{bmatrix} = \begin{bmatrix} arctan(-h_x^n,\ h_z^n) \\ arctan\big(h_y^n,\ \sqrt{(h_x^n)^2 + (h_z^n)^2}\big) \end{bmatrix} \tag{7}$$

3) Landmark Prediction and Outliers Rejection: At the update step of the Extended Kalman Filter the position of the features observed in the image is compared to the expected projection of the map landmarks in the image. The projection of a landmark in the map to the image starts with the projection from the navigation frame to the camera frame:



$L_i = \begin{bmatrix} L_{xi} \\ L_{yi} \\ L_{zi} \end{bmatrix}$
$1/\rho_i = d_i$
h^c
$m(\theta_i, \phi_i)$
ϕ_i
θ_i
z^c
$\begin{bmatrix} x_i^n \\ y_i^n \\ z_i^n \end{bmatrix}$
C
x^c
y^c
$(x_c)^b$
y^b
x^b
B
z^b
$(x_b)^n$
N
E
N
D

Fig. 3: Reference frames and Inverse Depth Parametrization

$$h^c = C_b^c C_n^b \left(\rho_i \left(\begin{bmatrix} x_i^n \\ y_i^n \\ z_i^n \end{bmatrix} - (x_b)^n - C_b^n (x_c)^b \right) + m(\theta_i, \phi_i) \right) \tag{8}$$

The distortion model is then applied to h^c, followed by the pinhole model, to determine the projection in the image.

Finally, wrong feature matches are rejected through the 1-Point RANSAC algorithm [6], that takes into account the prior probabilistic distributions maintained by the EKF to reduced the minimal sample size to only one feature match, significantly reducing the computational complexity associated with the standard RANSAC algorithm.

4) Inertial Based State Prediction: To avoid the scale factor ambiguity, the main limitation of monocular SLAM caused by the absence of metric information, inertial measurements from a low cost IMU are injected in the EKF prediction step. Since the map landmarks are static, only the platform localization states are subjected to the motion model, that consists on the inertial mechanization in the local level reference frame, respecting the following equations:

$$\begin{bmatrix} x^n(k) \\ \Theta^n(k) \\ v^n(k) \end{bmatrix} = \begin{bmatrix} x^n(k-1) + v^n(k)\Delta t \\ \Theta^n(k-1) + E_b^n w^b(k)\Delta t \\ v^n(k-1) + \big(C_b^n a^b(k) + g^n\big)\Delta t \end{bmatrix} \tag{9}$$

where the IMU inputs are identified by a^b and w^b, respectively the linear accelerations and angular velocities, measured in the body reference frame. C_b^n is the direction cosine matrix obtained from the platform orientation and E_b^n is a 3 by 3 matrix that converts the angular velocities into the Euler angles rate of change:

$$E_b^n = \begin{bmatrix} 1 & sin(\phi)tan(\theta) & cos(\phi)tan(\theta) \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi)sec(\theta) & cos(\phi)sec(\theta) \end{bmatrix} \quad (10)$$

B. *Modelling Algorithm*

The three-dimensional model is constructed by placing all gallery cross-sections, taken by the vertical laser range finder, into a common coordinate system.

First, laser range finder scans, initially expressed in polar coordinates, are converted to the Cartesian coordinate system with origin matching the center of the laser range finder. Next, specific position and orientation of each scan is derived from the two closest localization points in time. Given the calibration parameters that describe the spatial relationship between sensors, and using the calculated scan localization, all vertical cross-sections are transformed to the local level frame according to the formula below:

$$P^n = C_b^n\left(C_c^b\left(C_l^c\left(P^l - (x_l)^c\right) - (x_c)^b\right) - (x_b)^n\right) \quad (11)$$

where P^n is the final point in the local level frame, whereas P^l refers to the original point in the sensor Cartesian system. The rotation matrices C_c^b and C_l^c establish the rotation from camera to body and laser to body reference frames, respectively. Whereas $(x_c)^b$ define the camera position in the body frame and $(x_l)^c$ the laser position with respect to the camera frame. Finally C_b^n and $(x_b)^n$ enclose the rigid body transformation from the body to the local level reference frame.

After applying formula (11) to all points of all scans, a point cloud model is achieved (see Fig. 4). Usually, the interpretation of point clouds is not easy due to lack of surfaces. To improve the scene's perception, original surfaces are reconstructed by converting the point cloud into a triangular mesh, using the Ball Pivoting Algorithm (BPA) [10]. Through BPA, a realistic world representation can be attained without data losses (see Fig. 4).

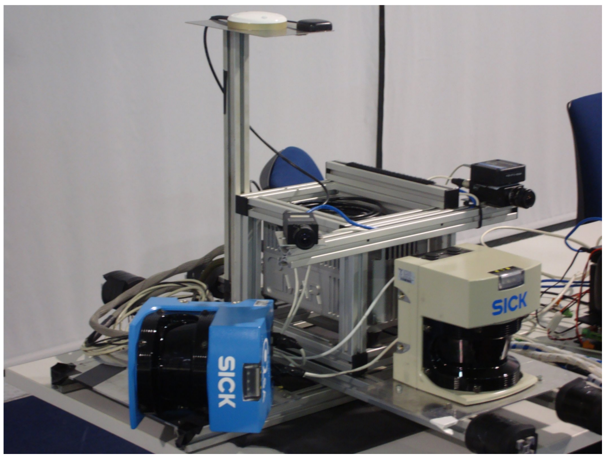


Fig. 5: Sensor platform used for data acquisition

To reduce the noise and produce smoother surfaces, a Laplacian filter is applied to the whole triangular mesh, computing a new position for each vertex according to local information given by adjacent points.

Both the point cloud model and the triangular mesh are coded in the VRML format to be displayed in a virtual reality application.

III. Dataset Acquisition

Solving the localization and modelling problems demands previous acquisition of a variety of measurements. To this purpose different types of sensors where assembled in a rigid platform (see Fig. 5), which in turn is mounted on top of a car.

The vertical cross-sections are taken by the vertical laser range finder (SICK LMS-200) at 75Hz with an angular resolution of 1°. There are two pointing-forward cameras (JAI CB-080GE), arranged in a stereoscopic configuration, with a resolution of 1032(h)x778(v) and controlled by an external trigger at a frame rate of 7 fps. Only the images from the left camera are used in our SLAM system.

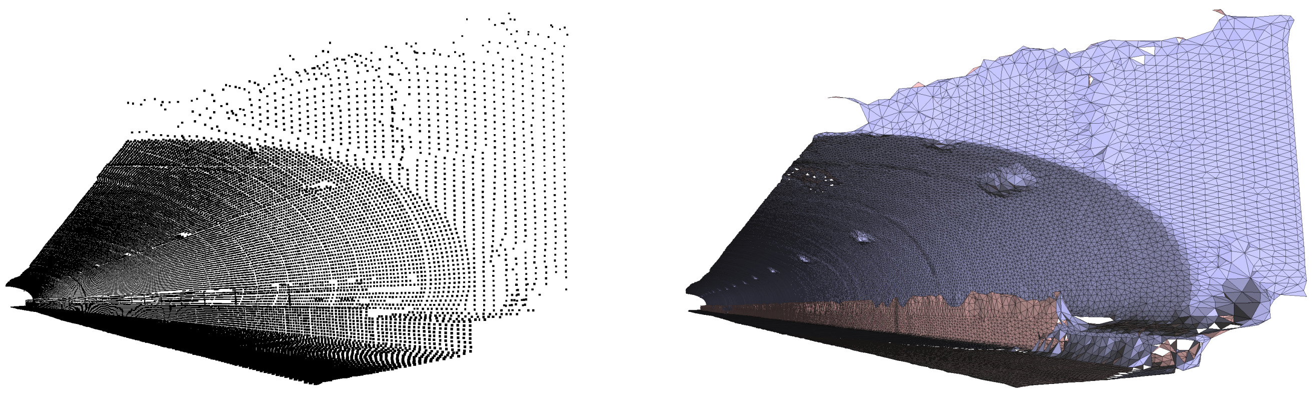


Fig. 4: Three-dimensional models of a road tunnel: point cloud (left) and triangular mesh produced by the Ball Pivoting Algorithm (right).

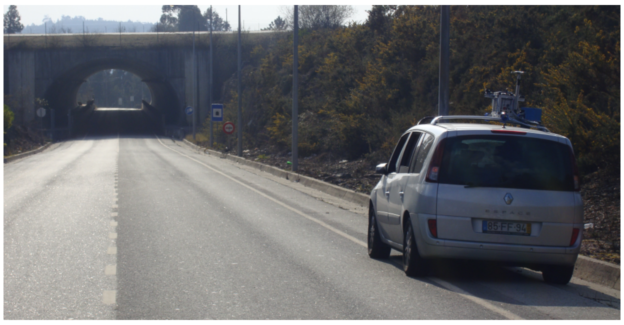


Fig. 6: Preparation for the data acquisition experiment in the tunnel area

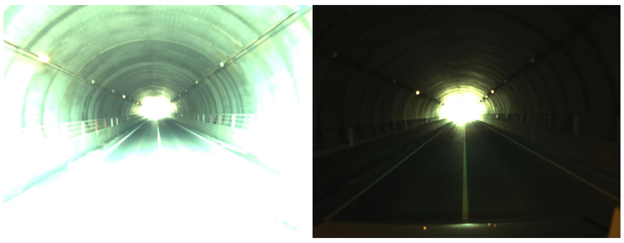


Fig. 7: Image instability as consequence of the illumination variations along the tunnel.

The low cost IMU (MicroStrain 3DM-GX1), placed above the left camera, gives the linear acceleration and angular velocity measurements used in the EKF prediction step, at a frequency of 100Hz.

Ground truth with a 400 Hz rate is obtained by the INS/GPS system (iMAR iNAV-FMS-E) placed in the center of the platform. This system provides raw inertial data and GPS measurements acquired outside the gallery, that are post-processed in a commercial software (Waypoint Inertial Explorer) to produce an accurate trajectory estimation. This trajectory is only used as ground truth to evaluate the SLAM performance.

All system reference clocks are synchronized with respect to GPS clock, to assure a consistent time base.

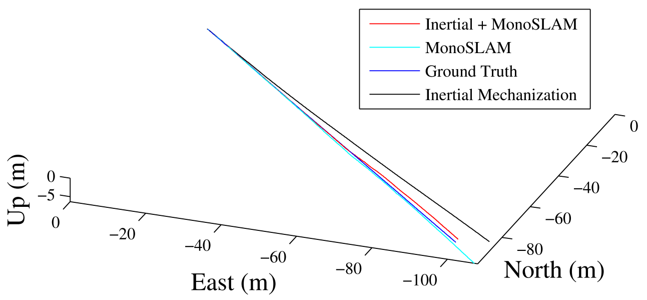


Fig. 8: Three-dimensional representation of the trajectories computed by the following methods: SLAM fusing inertial and visual data (red line), inertial mechanization (black line), monocular SLAM (light blue line) and ground truth (dark blue line)

The data acquisition experiment took place on a road tunnel with approximately 140 meters located at Vilar de Luz – Porto (see Fig. 6). To ensure the acquisition of equally spaced cross-sections of the tunnel walls, the vehicle moved at a nearly constant velocity of 35 Km/h. All data were correctly logged. However the images reflect the huge lighting variations between the interior and exterior of the tunnel, as a consequence of using fixed gain and non HDR (High Dynamic Range) cameras (see Fig. 7).

IV. Results

An accurate localization estimate is crucial to obtain a reliable model reproducing the real gallery characteristics. Since the ground truth is very close to the real trajectory, we were able to determine the error associated with our estimated localization. Furthermore, to realize the benefits of fusing inertial and visual measurements, both inertial navigation and MonoSLAM approaches were implemented, and the results are compared with the ones achieved by our approach.

The trajectories computed by these methods are outlined in Fig. 8. Although the path calculated by MonoSLAM apparently overlaps the ground truth, this approach shows the worst results due to the scale ambiguity, accumulating an error of 11.7 meters. As expected, inertial navigation drifts with time due to error integration, resulting in a total drift of 8.7 meters. Our approach produces the smallest error, showing the advantage of inertial and visual data fusion, with a maximum value of 1.29 meters and an error of 0.95 meters at the final position, equivalent to 0.7% of the total displacement. The insertion of inertial measurements in the MonoSLAM mechanism successfully prevents the scale factor ambiguity, whereas visual data contributes to the inertial drift compensation, particularly to the orientation states correction.

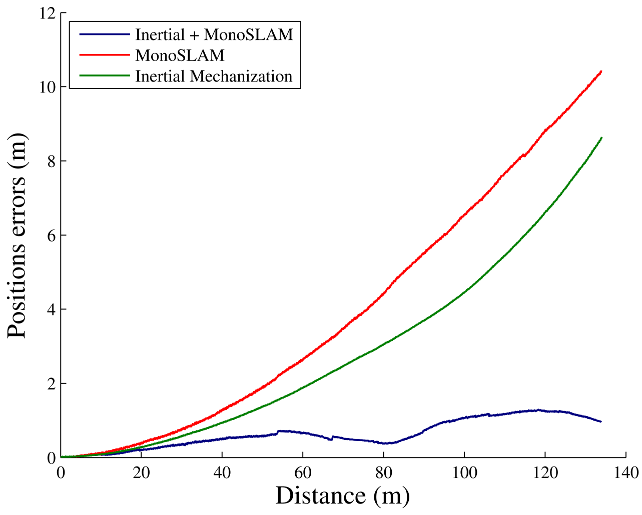


Fig. 9: Position errors produced by each localization strategy: SLAM fusing inertial and visual data (blue line), inertial mechanization (green line) and monocular SLAM (red line)

It can be seen in Fig. 9 that, at some instants, the inertial MonoSLAM position errors momentarily increase. This behavior coincides with a considerable number of landmarks being converted from the inverse depth to the Cartesian representation. As documented in [13], the conversion induces errors in landmark states, that are propagated to the localization states. Nevertheless, in the moments after is visible an error attenuation, which indicates the ability of the SLAM mechanism to filter this perturbation. The aleatory oscillations exhibited in the inertial MonoSLAM position error are characteristic of a random walk situation.

The point cloud model in Fig. 4 was built using the localization estimates. As previously mentioned, the point cloud models can become really hard to understand, depending on the view point and scale. In order to reach a more explicit and realistic representation, a triangular mesh is constructed from the point cloud without data losses, through the Ball Pivoting Algorithm. In the final step the surfaces are filtered by a Laplacian smoother (Fig. 10).

V. Conclusion

The development of a mobile modelling system for large scale underground environments raises some difficult challenges, especially when dealing with monotonous geometry. Based on inertial and visual data we have implemented a localization method that does not depend on the geometric properties of the environment, thus it is specifically suited to operate inside smooth shape galleries like traffic tunnels.

Through localization results the benefit of fusing inertial data within the MonoSLAM strategy became evident. In the most aggressive configuration, with a pointing forward camera, forward motion and large illumination variance, our localization estimate reached an error of 0.95% of the total displacement, which constitutes a quite impressive accomplishment given the low cost sensors used.

Despite the poor image quality, reliable visual features and descriptors where extracted by the SIFT algorithm, exploiting the algorithm's immunity to rotation scale and linear illumination variations, enabling robust frame-to-frame feature matching.

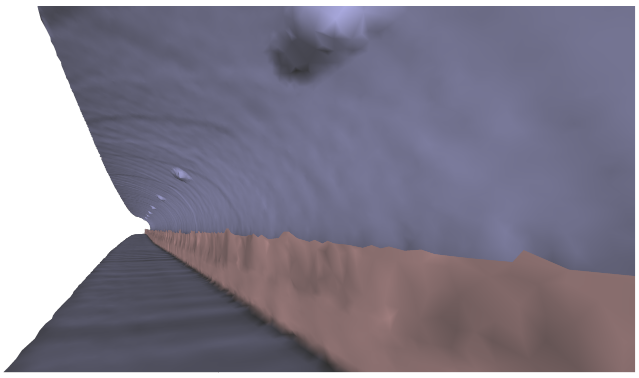


Fig. 10: Triangular mesh model after Laplacian filtering.

In the future, localization accuracy could be improved by adding other types of information, for instance, laser range finder measurements to provide a better approximation of the landmarks initial depth. A stereo vision system will also be implemented to enable instant computation of close landmark coordinates. The use of cameras with larger field of view will also be beneficial, enabling the observation of landmarks with high parallax and hence low depth uncertainty.

In order to enhance the models' realism, the mesh triangles will be filled with texture captured by the cameras.

Acknowledgment

This work was supported by QREN – Project n° 7865 "FlexiMap3D". The authors would like to thank the FlexiMap3D project members, affiliated with LSA-ISEP (Laboratório de Sistemas Autónomos - Instituto Superior de Engenharia do Porto), that were involved in the data acquisition stage.

References

[1] Andreas Nüchter, Hartmut Surmann, Kai Lingemann, Joachim Hertzberg and Sebastian Thrun, *6D SLAM with an Application in Autonomous Mine Mapping*, IEEE International Conference on Robotics and Automation (ICRA), pages 1998–2003, 2004.

[2] Daniel Huber and Nicolas Vandapel, *Automatic Three-dimensional Underground Mine Mapping*, The International Journal of Robotics Research, volume 25, pages 7–17, January 2006.

[3] Sebastian Thrun, Dirk Hähnel, David Ferguson, Michael Montemerlo, Rudolph Triebel, Wolfram Burgard, Christopher Baker, Zachary Omohundro, Scott Thayer and William Whittaker, *A System for Volumetric Robotic Mapping of Abandoned Mines*.

[4] Pedro Pinies, Todd Lupton, Salah Sukkarieh, Juan D. Tardós, *Inertial Aiding of Inverse Depth SLAM Using a Monocular Camera*, IEEE International Conference on Robotics and Automation (ICRA), pages 2797–2802, 2007.

[5] Javier Civera, Andrew J. Davison and J. M. M. Montiel, *Inverse Depth Parametrization for Monocular SLAM*, IEEE Transactions on Robotics, volume 24, number 5, pages 932–945, October 2008.

[6] Javier Civera, O. Garcia, Andrew J. Davison and J. M. M. Montiel, *1-Point RANSAC for EKF-Based Structure from Motion*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), October 2009.

[7] Paul J. Besl and Neil D. McKay, *A Method for Registration of 3-D Shapes*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(2), pages 239–256, 1992.

[8] R. Smith, M. Self and P. Cheeseman, Estimating Uncertain Spatial Relationships in Robotics, In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, Springer-Verlab, 1990.

[9] J. J. Leonard and Durrant H. Whyte, *Simultaneous Map Building and Localization for an Autonomous Mobile Robot*, IEEE International Conference on Intelligent Robots and Systems (IROS), Osaka, Japan, 1991.

[10] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, *The Ball-Pivoting Algorithm for Surface Reconstruction*, IEEE Transactions on Visualization and Computer Graphics (TVCG), 5(4), pages 349–359, 1999.

[11] David G. Lowe, *Distinctive Image Features from Scale-Invariant Keypoints*, International Journal of Computer Vision (IJCV), 60(2), pages 91–110, 2004.

[12] Martin A. Fischler and Robert C. Bolles, *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*, Readings in Computer Vision: Issues, Problems, Principles, and Paradigms, pages 726–740, 1987.

[13] Javier Civera, Andrew J. Davison, J. M. M. Montiel, *Inverse Depth to Depth Conversion for Monocular SLAM*, International Conference on Robotics and Automation (ICRA), pages 2778–2783, 2007.

Learning Velocity Kinematics: Experimental Comparison of On-line Regression Algorithms

Alain Droniou, Serena Ivaldi, Patrick Stalph, Martin Butz and Olivier Sigaud

***Abstract*—The increasing complexity of tasks addressed by humanoid robotics requires accurate mechanical models which are difficult to obtain in practice. One approach is to let the robot learn its own models. In [16], two algorithms were compared on learning the velocity kinematics model of iCub: XCSF and LWPR. This comparison was based on simulated data. In this paper, we propose to extend this study to data recorded from the iCub robot. We analyze the behavior of these algorithms in presence of large noise in real conditions of use. We also add the study of a third algorithm, iRFRLS. After a detailed study on the tuning of the three algorithms, we show that the results obtained in [16] are still valid on real data: XCSF converges more slowly, but to a lower error than LWPR. However, we show that iRFRLS outperforms these two algorithms.**

I. Introduction

For the few last decades, robot control was based on hard-coded mechanical models. Even if it can still be suitable for complex robots with many degrees of freedom [10], such methods usually do not take into account perturbations, like viscous and solid friction (which vary with mechanical wear), contacts with unknown objects, ... For instance, when interacting with new objects, autonomous robots require new mechanical models to adapt their behavior: playing with a bottle is different depending on its size, its material, whether it is filled or empty... Learning techniques are expected to provide efficient and fast generalization to transfer knowledge between related situations.

In [16], two algorithms, LWPR and XCSF, were compared on learning velocity kinematics of the humanoid robot iCub from visual inputs. In order to introduce uncertainties and address a situation where the model is unknown, the end-effector was modeled by a green ball at the top of a stick held by the robot. The study was carried on simulated data. In this paper, we refine the study with data recorded from the real robot. In fact, as soon as we work in real conditions, we must face multiple sources of noise. In these conditions, a good algorithm must be robust and its generalization capability is crucial.

Alain Droniou (PhD candidate in Robotics), Serena Ivaldi (postdoctoral researcher in Robotics), and Olivier Sigaud (Professor in Computer Science) are with: Université Pierre et Marie Curie, Institut des Systèmes Intelligents et de Robotique - CNRS UMR 7222, Pyramide Tour 55 - Boîte Courrier 173, 4 Place Jussieu, 75252 Paris CEDEX 5, France, Contact: `firstname.name@isir.upmc.fr`
Patrick Stalph (PhD candidate in Computer Science) and Martin Butz (Professor in Computer Science) are with: Universität Tübingen, Sand 14, 72076 Tbingen, Germany
Contact: `name@informatik.uni-tuebingen.de`

We first describe the learning task and algorithms in section II. In addition to XCSF and LWPR, we also study a third algorithm, iRFRLS, and we highlight its differences with respect to XCSF and LWPR. Then, we present the experimental set-up in section III. In section IV, we compare the performances of the three algorithms and show that iRFRLS converges faster and to a lower error than the other algorithms. We discuss this result in section V, as well as the influence of the parameters of the algorithms.

II. Methods

In this section, we first describe the learning task. Then, we present the three studied algorithms.

A. Learning velocity kinematics

Our approach to learning robot control is to learn the forward velocity kinematics model and then invert it for control. This provides more flexibility than directly learning an inverse model [13].

1) Forward kinematics: The simplest model of a robot is given by its kinematics, i.e. the correspondence between joint space coordinates (articular position of each joint) $\mathbf{q}$ and task space coordinates ξ, usually 3-D coordinates of the end-effector: $\xi = f(\mathbf{q})$.

When the joint space has more dimensions than the task space, the system is redundant and there is no simple method to span the set of solutions at the kinematics level. Therefore, the model is usually derived to obtain the velocity kinematics model through the expression of a Jacobian matrix $J(\mathbf{q}) = \frac{\partial f}{\partial \mathbf{q}}$:

$$\dot{\xi} = J(\mathbf{q})\dot{\mathbf{q}} \tag{1}$$

2) Inverse velocity kinematics: In order to be able to control the robot, we need to inverse (1) using

$$\dot{\mathbf{q}} = J^{\sharp}(\mathbf{q})\dot{\xi} \tag{2}$$

where $J^{\sharp}$ is a pseudo-inverse of J. With (2), we can compute joint velocities corresponding to the desired end-effector velocity.

However, in the redundant and non singular case, there is an infinite number of solutions $\dot{\mathbf{q}}$ providing the desired velocity $\dot{\xi}^{\star}$. Among all these possibilities, the Moore-Penrose pseudo-inverse J^{+} provides the minimum norm solution [4]. To avoid critical effects at singular configurations, regularization terms can be added, like in Damped Least Square Pseudo-inverse (DLS - PINV) [3]. Further information on control in our experiments can be found in [16].

B. Regression algorithms

We focus our study on three regression algorithms: LWPR [19], XCSF [21] and iRFRLS [6]. A survey of these three algorithms can be found in [17].

1) LWPR: The Locally Weighted Projection Regression (LWPR[1]) algorithm [19] is a recursive function approximator, which provides accurate approximation in very large spaces at low computational cost.

It uses a sum of linear models weighted by normalized Gaussians. These Gaussians, also called receptive fields (RFs), define the domain of influence of each corresponding linear model. The RFs and their linear models are both updated incrementally to match the training data. LWPR reduces the input dimensionality using the Partial Least Squares (PLS) algorithm [22]. The global algorithm provides as output $\hat{y}$ the weighted sum of all outputs $\hat{y}_k$ of each RF

$$\hat{y}(x) = \frac{\sum_{k=1}^{K} w_k \hat{y}_k(x)}{\sum_{k=1}^{K} w_k} \tag{3}$$

where K is the number of receptive fields.

We refer the reader to [15] or [19] for a further presentation of the incremental version of the algorithm.

2) XCSF: XCSF[2] [21] is another function approximator that shares some similarities with LWPR but comes from Learning Classifier Systems (LCSs) [7]. As any LCS, XCSF manages a population of rules, called *classifiers*. These classifiers contain a *condition part* and a *prediction part*. In XCSF, the condition part defines the domain $\phi_i(z)$ of a local model whereas the prediction part contains the local linear model β_i itself. From classifiers, XCSF predicts a local output vector y_i relative to an input vector x_i. The β_i are updated using the Recursive Least Squares (RLS) algorithm [8], the incremental version of the Least Squares method. The classifiers in XCSF form a population P that clusters the condition space into a set of overlapping prediction models. XCSF uses only a subset of the classifiers to generate an approximation. Indeed, at each learning iteration, XCSF generates a match set M that contains all classifiers in the population P whose condition part $\mathcal{Z}$ matches the input data z i.e., for which $\phi_i(z)$ is above a threshold ϕ_0.

In XCSF, the output $\hat{y}$ is given for a (x, z) pair as the sum of the linear models $\hat{y}_i$ of each matching classifier i weighted by its fitness F_i

$$\hat{y}(x,z) = \frac{\sum_{k=1}^{n_M} F_k(z)\, \hat{y}_k(x)}{\sum_{k=1}^{n_M} F_k(z)} \tag{4}$$

where n_M is the number of classifiers in the match set M. In all other respects, the genetic algorithm (GA) that drives the evolution of the population of classifiers is directly inherited from XCS [20].

An important process in the context of this study is *compaction*. At the end of a learning process, the final population is composed of highly overlapping classifiers. To reduce the size of the population, XCSF uses a Closest Classifier Matching (CCM) rule [2] to get a fixed size match set M.

[1]We use the *liblwpr* C/C++ library available on http://www.ipab.inf.ed.ac.uk/slmc/software/lwpr/index.html.

[2]We use the Java *XCSFServer* developed by M. Butz and P. Stalph [18].

3) iRFRLS: iRFRLS[3] is based on a regularized least square method with random features [6]. In this paper, we present this algorithm from a different point of view, based on Fourier transform.

Almost any usual function can be calculated by inversion of its Fourier transform. Approximating the Fourier transform of a function instead of the function itself allows to exploit some regularity properties to make the learning process more robust and simpler.

Since we work on real, continuous functions defined on finite intervals (and so virtually periodic), the partial sums of the Fourier series of f,

$$\begin{aligned} S_n(f(x)) = \frac{a_0(f)}{2} &+ \sum_{k=1}^{n} a_k(f)\cos\left(kx\frac{2\pi}{T}\right) \\ &+ \sum_{k=1}^{n} b_k(f)\sin\left(kx\frac{2\pi}{T}\right) \end{aligned} \tag{5}$$

converge towards f when $n \to \infty$, with

$$\begin{aligned} a_n(f) &= \frac{2}{T}\int_{-T/2}^{T/2} f(t)\cos\left(nt\frac{2\pi}{T}\right)\,\mathrm{d}t \\ b_n(f) &= \frac{2}{T}\int_{-T/2}^{T/2} f(t)\sin\left(nt\frac{2\pi}{T}\right)\,\mathrm{d}t \quad \forall n \geq 0 \end{aligned} \tag{6}$$

To approximate the function, we can set some (actually most of) coefficients to 0, and learn the others. The quality of predictions depends in that case on the number D of learned coefficients (features). Writting $x_1, ..., x_k$ the points from the training set and

$$\begin{aligned} f &= [f(x_1), ..., f(x_k)]\,,\ w = [a_{i_0}, b_{i_0}, ..., a_{i_D}, b_{i_D}]^T \\ z(x) &= [cos(\omega_{i_0}x), sin(\omega_{i_0}x), ..., cos(\omega_{i_D}x), sin(\omega_{i_D}x)]^T \\ Z &= [z(x_1), ..., z(x_k)]^T \end{aligned}$$

the prediction can be written $\hat{f}(x) = w^T z(x)$ and a classical single vector regression method [5] can be used to learn the coefficients. The solution is then given by

$$w = (\lambda I + Z^T Z)^{-1} Z^T f \tag{7}$$

where λ is a regularization factor. This solution w can be computed incrementally and easily inverted with a Cholesky decomposition of $(\lambda I + Z^T Z)$ [1], [14], [6].

Finally, iRFRLS requires to tune only three parameters: the regularization factor λ, the number of features D and the distribution of pulsations ω. Because of a strong relationship with Gaussian kernel methods, one usually chooses normal distributions whose only parameter γ is the variance. In these conditions, iRFRLS behaves as a Gaussian kernel regression algorithm. Features $z(x)$ then approximate Gaussian kernel values [11].

[3]We use the C++ library developed by A. Gijsberts, which can be found in the official iCub repository https://robotcub.svn.sourceforge.net/svnroot/robotcub/trunk/iCub.

III. Experiments

We use the 104cm high and 53 degrees of freedom humanoid robot iCub [9]. In [16], algorithms were tested on an iCub simulator. For this study, we use the real robot and hence face real conditions of use. As a consequence, we must address multiple sources of noise: imprecision of sensors, power supply noise, mechanical vibrations and backlashes, inaccurate vision algorithms, imperfect calibration or non-simultaneous measures.

A. Data

We tune and evaluate the algorithms on data recorded according to the setup of [16]. For those experiments, iCub is performing an asterisk reaching task. As we do not want to address vision process difficulties, the end-effector is represented by a green ball at the top of a stick held by the robot (see [16] for further details). As a results, the kinematics model is unknown. The task involves 6 degrees of freedom: 2 for the head and 4 for the arm. During these experiments, the explored region consists of a cube of about 10 cm side length. Compared with the 40 cm of the iCub arm, it represents roughly 20% of the useful reachable space[4]. This can bias some results because this emphasizes some linearities.

We recorded 4 million points $(\mathbf{q}, \dot{\mathbf{q}}, \xi, \dot{\xi})$ corresponding to 30 experiments, sampled at approximately 20Hz.

B. Tuning the algorithms

One of the main points when working with learning methods is the tuning of each algorithm. In order to perform a fair comparison, we use a grid-search method: for each parameter, a set of values is chosen (Tables I, II, III), and we test all possible combinations of values.

To get statistically significant results using paired Student's t-tests [12], each experiment is repeated on the 30 recorded datasets. These datasets are the same for the three algorithms.

We measure the performance as the mean squared error in the prediction $\hat{\dot{\xi}}$ of $\dot{\xi}$: $MSE = \frac{1}{3}\mathbb{E}\left[||\hat{\dot{\xi}} - \dot{\xi}||^2\right]$ where coefficient $\frac{1}{3}$ comes from the dimension of $\dot{\xi}$. Test sets consist of 1000 points, different from learning sets.

Thanks to the distinction between condition and prediction spaces, XCSF can directly learn the velocity kinematics Jacobian matrix. On the other hand, LWPR and iRFRLS cannot provide directly this matrix. We use them to learn the kinematics model, from which we can compute the velocity kinematics Jacobian matrix by derivation.

IV. Results

In this section, we carry out experiments to study both the influence of each parameter and the influence of the data order on the algorithms. We focus on the convergence of the algorithms and their computation time.

[4]One might approximate the reachable space as a half sphere of radius 40 cm. This would lead to a ratio explored/reachable of 0.7%. Actually, due to mechanical constraints and joints limits, the reachable space is much smaller.

A. Influence of the parameters

In order to study the influence of parameters of each algorithm, we compute the average error obtained on all simulations by setting the value of one parameter at a time. The 28 XCSF parameters cannot be exhaustively tested. We select six of them, identical to those tested in [16]. The results are presented in Table I. We do the same for LWPR, in Table II. Since iRFRLS has only three parameters, it is possible to perform an exhaustive search (Table III).

TABLE I
INFLUENCE OF EACH PARAMETER FOR XCSF. THE NOTATION $v_1 \overset{c}{>} v_2$ MEANS THAT THE VALUE v_1 OF THE PARAMETER LEADS TO BETTER PERFORMANCE THAN THE VALUE v_2, WITH A CONFIDENCE OF $c\%$. VALUES IN BOLD (ITALIC) CORRESPOND TO THE BEST (WORST) SET OF PARAMETERS. GRID-SEARCH TIME CORRESPONDS TO THE TIME NEEDED TO TEST EVERY SET OF PARAMETERS ON 30 DATASETS, WITHOUT PARALLELIZATION.

XCSF		
Parameter	Ranking	Time
Δ	$0.01 \overset{76}{>} 0.05 \overset{99.2}{>} \boldsymbol{\mathit{0.1}} \overset{84}{>} 0.5$	constant
converConditionRange	$\boldsymbol{\mathit{0.995}} \overset{99.9}{>} 0.7$	constant
minConditionStretch	$0.001 \overset{98}{\approx} \boldsymbol{\mathit{0.005}} \overset{81}{\approx} 0.1$	constant
ϵ_0	$0.01 \overset{98}{>} \mathbf{0.005} \overset{99.4}{>} \mathit{0.001}$	constant
maxPopulationSize	$\boldsymbol{\mathit{1500}} \overset{90}{>} 500 \overset{100}{>} 100$	increasing
startCondensation	$\boldsymbol{\mathit{500}} \overset{67}{\approx} 1000 \overset{99.9}{>} 2000 \overset{99.9}{>} 5000$	increasing
Grid-search time		9 days

TABLE II
INFLUENCE OF EACH PARAMETER FOR LWPR.

LWPR		
Parameter	Ranking	Time
useMeta	$\boldsymbol{\mathit{true}} \overset{89}{\approx} \text{false}$	constant
updateD	$\text{false} \overset{99.7}{>} \boldsymbol{\mathit{true}}$	*true* longer
penalty	$\mathbf{0.001} \overset{67}{>} \mathit{0.01} \overset{93}{\approx} 0.1$	constant
wGen	$0.9 \overset{92}{\approx} \mathbf{0.2} \overset{64}{>} 0.1 \overset{93}{>} \mathit{0.01}$	increasing
setInitAlpha	$\mathbf{0.01} \overset{58}{>} 0.1 \overset{67}{>} 200 \overset{98}{\approx} \mathit{500}$	constant
setInitD	$\mathbf{50} \overset{75}{>} 40 \overset{66}{>} 30 \overset{84}{>} \mathit{20}$	increasing
Grid-search time		11 hours

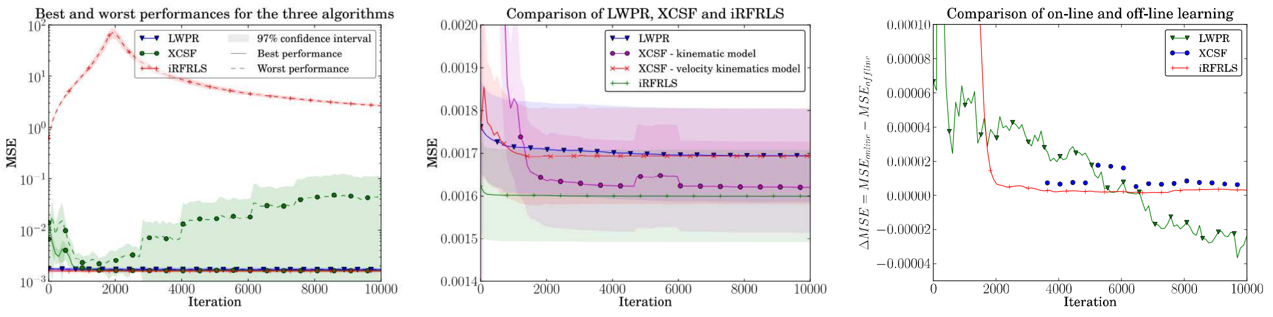
TABLE III
INFLUENCE OF EACH PARAMETER FOR iRFRLS.

iRFRLS		
Parameter	Ranking	Time
γ	$\mathbf{10^{-6}} \overset{99.9}{>} 10^{-12} \overset{96}{\approx} 10^{-3} \overset{100}{>} 1 \overset{100}{>} \mathit{10}$	constant
λ	$10 \overset{99.9}{>} 1 \overset{99.9}{>} \mathit{10^{-3}} \overset{97}{\approx} 10^{-6} \overset{96}{\approx} \mathbf{10^{-12}}$	constant
D	$50 \overset{100}{>} 500 \overset{99.9}{>} \mathbf{1000} \overset{99.9}{>} \mathit{2000}$	$\mathcal{O}(D^2)$
Grid-search time		12 days

Plots in Fig. 1(a) show the worst and best performances obtained by each of the three algorithms.

B. Comparison of learning algorithms

In [16], parameters were optimized for learning a velocity kinematics model for which data were generated randomly



(a) Error plots obtained with the best and the worst set of parameters for LWPR, XCSF and iRFRLS. Note logarithmic scale.

(b) Comparison of the best sets of parameters for each algorithm. XCSF was tested on both kinematics and velocity kinematics models.

(c) Comparison between on-line and off-line learning. We represent the difference $\Delta MSE = MSE_{online} - MSE_{offline}$.

Fig. 1. Empirical study of the behavior of LWPR, XCSF and iRFRLS

(no temporal relationship between them, excluding any notion of trajectory). Therefore, we start with a similar approach, before testing the algorithms when the data is ordered along the trajectories of the robot.

1) Offline Learning: Initially, training and testing sets are generated by drawing random points from the database. Fig. 1(b) compares the performance of the three algorithms with the best sets of parameters.

First, we note that the final performance of XCSF is better when learning the velocity kinematics Jacobian matrix than when learning the kinematics model (with a confidence higher than 99.9%). In order to compare the best performance of the algorithms, we evaluate XCSF on learning the velocity kinematics model.

The convergence is almost instantaneous for LWPR and iRFRLS (models are correctly learned in less than 500 iterations). However, the convergence of XCSF is much longer. That confirms a result on simulated data in [16]: XCSF converges slower than LWPR, but the final performance is better (confidence of 98.8%). Similarly, iRFRLS converges as fast as LWPR with a better final performance (confidence higher than 99.9%). But if it converges faster than XCSF, the difference in final performance is significant up to only 84%.

2) Online Learning: We now use sets of consecutive points instead of randomly selected ones. We thus confront the algorithms in real conditions, where learning is constrained by the movements of the robot. Fig. 1(c) represents the difference between the error obtained with off-line and on-line training $\Delta MSE = MSE_{online} - MSE_{offline}$. Though the convergence is slower, the final performance is similar for iRFRLS and XCSF, while it is better for LWPR. However, the final ranking is identical to the one for off-line training: iRFRLS is always the fastest to converge and the best in terms of final performance.

V. Discussion

From the results presented in the previous section, iRFRLS appears superior in all respects. However, some points require further analysis.

A. *Performance*

The results in section IV-B induce two discussions: one on the interest of optimization, the other on the influence of the data order. Table IV provides a synthesis of these discussions.

1) Advantages of optimization: If the impact of parameter optimization is tangible with the three algorithms (Fig. 1(a)), its interest is limited for LWPR: neither the final performance (gain less than 2 %) nor the speed of convergence are impacted significantly. However, this optimization is the fastest of the three algorithms. In contrast, optimization is essential for iRFRLS (gain by a factor of 1000) as well as XCSF for which final performance, speed of convergence and stability are impacted.

2) Data order: Obviously, the convergence speed of the algorithms depends on the time needed to cover the space with enough training data. Hence, the convergence is slower when data order follows the trajectory, compared to a random order. Studying the impact on the final performance is more relevant.

Theory ensures that the final performance of iRFRLS does not depend on the data order, except for rounding errors that may accumulate around the calculation of the incremental Cholesky factorization. Similarly, the GA in XCSF reduces the dependence of the algorithm on the data order. Fig. 1(c) indeed shows a fast convergence of iRFRLS to the same level of error when learning off-line. XCSF is slower to converge, but also reaches the same level of error.

Unlike XCSF, LWPR does not have any mechanism to optimize the position of the RFs centers. Since the data order determines the RFs creation order and hence their positions, it changes the final performance, as shown in Fig. 1(c). However, contrary to expectations, it must be emphasized that the final performance of LWPR is better when data are presented along trajectories. This can be because it is easier to identify and properly filter noise when data correspond to consecutive points of the trajectory, building RF overlapping along these trajectories and whose combination gives a more reliable prediction. However, if the signal to learn is not as noisy, this mechanism may work against LWPR by smoothing inner variations of the estimated function. This remains to be studied.

TABLE IV
SYNTHESIS OF THE PERFORMANCE COMPARISON OF LWPR, XCSF AND iRFRLS.

Algorithm	LWPR	XCSF	iRFRLS
Convergence	Fast	Slow	Fast and unstable
Final performance	-	+	++
Sensitivity to tuning	Low	High	Extreme
Computation time	Fast	Variable, sometimes slow	Accuracy / time compromise
Main advantages	Fast and stable	Distinction condition / prediction	Constant cost, few parameters and good performances
Main drawbacks	Many parameters to optimize	Sometimes unstable and slow convergence, many parameters	Slow for large values of D, sensitive to tuning

B. Comparative study

In this section, we analyse the influence of the parameters with respect to the specificities of our experimental set-up, in order to highlight some rules which simplify tuning.

1) Noise and regularity of estimated functions: In our study, the function to learn is very regular, with strong linearities. Therefore this impacts the tuning of parameters and explains the fast convergence of the algorithms when learning the kinematics model. Because of its higher dimensionality, learning the Jacobian matrix with XCSF is slower (Fig. 1(b)).

However, learning the kinematics model to compute the velocity kinematics model increases the error variance (since it uses twice the model to calculate the velocity), which can justify the lower final performance of XCSF with the kinematics model compared to the velocity kinematics Jacobian.

Besides the regularity of the function to learn, data are very noisy. Hence, learning regular models is even more important to counter over-fitting. This favor larger areas of validity for XCSF classifiers, which is reflected in the value of *coverConditionrange* which hides *minConditionStretch* values. Similarly, a higher value for ϵ_0 smooths predictions, taking into account more classifiers for each prediction. To avoid over-fitting, the tuning of the GA is crucial: one must avoid to specialize the population on noise, and favor a high turnover. This is achieved through *startCondensation* and Δ parameters. Working in Fourier space, iRFRLS filters noise more efficiently, which may explain its superiority. Note that a choice of a too high γ for iRFRLS, i.e. an over-representation of high frequencies, degrades the performance.

On the contrary, LWPR seems to favor smaller RFs, through the high value of *setInitD*. This seems contradictory to the previous analysis, but unlike XCSF, LWPR does not optimize the center of its RFs, and smaller RFs then allow to achieve better accuracy and adaptivity thanks to a finer partitioning of the space.

2) Models complexity: LWPR and XCSF experiments highlight the existence of a minimum population size for both algorithms. For XCSF, it is between 100 and 500 classifiers, as shown by the large difference in performance for these two values of *maxPopulationSize*, while the gain is much more limited from 500 to 1500 classifiers. For LWPR, the effect is noticeable on *wGen*, the threshold that determines the creation of a new RF. The gain is much more important between 0.01 and 0.1, than with further increases.

The study of the features number D of iRFRLS is much more surprising at first sight. Indeed, this number is directly related to the quality of the Fourier transform approximation and a high value should always be preferred. But the study shows an average performance decreasing with the number of features. Because of the strong linearities of the kinematics model, a small number of frequencies is sufficient (and preferable, if one wants to avoid learning the discontinuities due to noise). In addition, the number of coefficients to be learned corresponds to the number of features and a small number of features thus facilitates learning. This explains why on average, with random values for the other parameters, it is easier to have better performances with only 50 features. Nevertheless, though the better performance on average is obtained with 50 features, the best performing model was obtained with 1000 features.

3) Computation Time:

C. Computation Time

In the robotics context, computation time is an important feature for learning algorithms, allowing or not real-time applications. Table V compares training and prediction time for each of the three algorithms[5].

TABLE V
MEAN (MAX) COMPUTATION TIME PER SAMPLE IN TRAINING AND PREDICTION.

Algorithm \ Time	Training	Prediction
LWPR	50μs (1ms)	50μs (1ms)
XCSF	300 μs (20ms)	300μs (20ms)
iRFRLS, D=50	20μs (25μs)	20μs (25μs)
iRFRLS, D=2000	50ms (50ms)	450μs (500μs)

LWPR is the fastest of the three algorithms, making it easily usable in real-time applications. The influence of the parameters (Table II) is thus not crucial.

For XCSF, the computation time depends mainly on the population size and on the starting time of condensation (table I). However, the latter effect is noticeable only in the learning phase: once the population is stable after condensation, the prediction time becomes constant. On average, training and prediction time are around 300μs, with a maximum of 20ms typically achieved just before condensation.

The computation time of iRFRLS depends on only the number D of features (Table III). This allows to easily search

[5]Experiments were carried out on an Intel Core i7 960 processor (4 cores, hyper-threading, 3.2GHz) with 6Gb RAM.

for a compromise between computation time and accuracy. One reaches 50 ms in learning process with $D = 2000$. Such computation time may require off-line training or significant computing resources for real-time applications.

D. Tuning cookbook

Through this discussion, we identified some rules thanks to which it is easier to tune the algorithms depending on the learning problem. They are presented on Table VI. The table must be read as follows: to deal with differences in the function regularity, for LWPR the best parameters to tune are those controlling RFs size and overlapping. To prevent over-fitting with LWPR, one should tune parameters controlling RFs optimization and meta-learning, etc.

TABLE VI
AN EASIER TUNING OF THE ALGORITHMS CAN BE ACHIEVED BY IDENTIFYING SOME RECURRENT ISSUES IN LEARNING PROBLEMS AND THEIR RELATED PARAMETERS.

Algorithm	Function regularity	Over-fitting	Time complexity
LWPR	RFs size and overlapping	RFs optimization and meta-learning	Always fast
XCSF	Classifiers size and number	Condensation and GA	Population size
iRFRLS	Sampling distribution		Number of features

VI. Conclusion and perspectives

In this work, we have learned models from iCub and shown that LWPR, XCSF and iRFRLS perform well in the presence of large noise sources. iRFRLS outperforms XCSF and LWPR, in terms of speed of convergence and performance. However, computation time can become prohibitive for iRFRLS for real-time applications. Nevertheless, it remains possible to ensure real-time computation at design level, thanks to a computation time / accuracy trade-off, by a relevant tuning of the number of features. We also highlighted that the optimization of the parameters is crucial for iRFRLS and XCSF, while the gain in performance is much lower for LWPR.

In addition, if on-line learning is slower, which is mainly due to the time required to cover the whole space, the impact on the final performance is minor for the three algorithms.

In the future, we want to validate our results on-line on iCub. This will provide two improvements to this study: it will be possible to test the algorithms on larger spaces and performances will be evaluated not only on the mean squared error of predictions, but also on the quality of the obtained trajectories. On a longer term, we also want to study the effects of the introduction of artificial curiosity mechanisms on the algorithms and learn the dynamics of the robot.

Acknowledgments

This work is supported by the French ANR program (ANR 2010 BLAN 0216 01), more at http://macsi.isir.upmc.fr

References

[1] Å. Björck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, 1996.

[2] Martin V. Butz and Oliver Herbort. Context-dependent predictions and cognitive arm control with XCSF. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, GECCO '08, pages 1357–1364, New York, NY, USA, 2008. ACM.

[3] Arati Deo and Ian Walker. Overview of damped least-squares methods for inverse kinematics of robot manipulators. *Journal of Intelligent and Robotic Systems*, 14:43–68, 1995. 10.1007/BF01254007.

[4] K. L. Doty, C. Melchiorri, and C. Bonivento. A theory of generalized inverses applied to robotics. *Int. J. Robotics Research*, 12(1), 1993.

[5] Harris Drucker, Chris Kaufman, Burges L., Alex Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in Neural Information Processing Systems 9*, volume 9, pages 155–161, 1997.

[6] Arjan Gijsberts and Giorgio Metta. Incremental learning of robot dynamics using random features. In *ICRA*, pages 951–956, 2011.

[7] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press, April 1992.

[8] T. Kailath, A.H. Sayed, and B. Hassibi. *Linear estimation*. Prentice-Hall information and system sciences series. Prentice Hall, 2000.

[9] Giorgio Metta, Giulio Sandini, David Vernon, Lorenzo Natale, and Francesco Nori. The iCub humanoid robot: an open platform for research in embodied cognition. In *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, PerMIS '08, pages 50–56, New York, NY, USA, 2008. ACM.

[10] Ugo Pattacini, Francesco Nori, Lorenzo Natale, Giorgio Metta, and Giulio Sandini. An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots. In *IROS*, pages 1668–1674. IEEE, 2010.

[11] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1177–1184. MIT Press, Cambridge, MA, 2008.

[12] Graeme D. Ruxton. The unequal variance T-test is an underused alternative to Student's T-test and the Mann-Whitney U test. *Behavioral Ecology*, 17(4):688–690, 2006.

[13] C. Salaun, V. Padois, and O. Sigaud. Control of redundant robots using learned models: an operational space control approach. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 878–885, Saint-Louis, USA, oct 2009. doi:10.1109/IROS.2009.5354438.

[14] Ali H. Sayed. *Adaptive Filters*. Wiley-IEEE Press, 2008.

[15] Stefan Schaal, Christopher G. Atkeson, and Sethu Vijayakumar. Scalable techniques from nonparametric statistics for real time robot learning. *Applied Intelligence*, 17:49–60, June 2002.

[16] G. Sicard, C. Salaun, S. Ivaldi, V. Padois, and O. Sigaud. Learning the velocity kinematics of iCub for model-based control: XCSF versus LWPR. In *Proceedings of the 11th IEEE-RAS International Conference on Humanoid Robots - Humanoids 2011*, Bled, Slovenia, 2011.

[17] Olivier Sigaud, Camille Salaün, and Vincent Padois. On-line regression algorithms for learning mechanical models of robots: a survey. *Robotics and Autonomous Systems*, 59:1115–1129, July 2011.

[18] Patrick O. Stalph and Martin V. Butz. Current XCSF capabilities and challenges. In *IWLCS 2008/2009*, LNCS (LNAI), vol. 6471, pages 57–69. Springer, 2010.

[19] Sethu Vijayakumar and Stefan Schaal. Locally weighted projection regression: An o(n) algorithm for incremental real time learning in high dimensional space. In *in Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000*, pages 1079–1086, 2000.

[20] S. W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[21] Stewart W. Wilson. Function approximation with a classifier system. *Genetic and Evolutionary Computation Conference, GECCO 2001*, pages 974–981, 2001.

[22] H. Wold. Soft Modeling by Latent Variables; the Nonlinear Iterative Partial Least Squares Approach. *Perspectives in Probability and Statistics. Papers in Honour of M. S. Bartlett*, 1975.

Towards fast and adaptive optimal control policies for robots: A direct policy search approach

Didier Marin and Olivier Sigaud

Abstract—**Optimal control methods are generally too expensive to be applied on-line and in real-time to the control of robots. An alternative method consists in tuning a parametrized reactive controller so that it converges to optimal behavior. In this paper we present such a method based on the "*direct Policy Search*" paradigm to get a cost-efficient control policy for a simulated two degrees-of-freedom planar arm actuated by six muscles. We learn a parametric controller from demonstration using a few near-optimal trajectories. Then we tune the parameters of this controller using two versions of a *Cross-Entropy Policy Search* method that we compare. Finally, we show that the resulting controller is 20000 times faster than an optimal control method producing the same trajectories.**

I. Introduction

The mission of robots is changing deeply. In contrast with factory robots always repeating the same task in a perfectly known environment, service robots will have to deal with complex tasks in the presence of humans, while being subject to many unforeseen perturbations. As a result of this evolution, a whole set of new control principles is needed. Among such principles, the study of Human Motor Control (HMC) suggests to call upon optimality and adaptation. However, Optimal Control (OC) methods are generally too expensive to be applied on-line and in real-time to the control of robots.

A straightforward solution to this cost problem that also comes with the benefits of continuous adaptation consists in calling upon incremental, stochastic optimization methods to improve the behavior of the system all along its lifetime through its interactions with its environment. This generates intensive research and, in particular, the *direct Policy Search* methods are providing more and more interesting results (e.g. [7], [8], [6], [1]). Since such methods are generally sensitive to local minima problems, the optimization process is generally preceded by a *Learning from Demonstration* (LfD) stage that drives the process to the preferred local optimum.

In [9], the potential of such an approach was demonstrated through a simple reaching experiment with a 2D arm controlled by 6 muscles. An OC method based on a computationally expensive variational calculus process was replaced by the combination of two adaptive components. First, a parametric controller was learned from demonstration using a few near-optimal trajectories, but the resulting policy was suboptimal. Thus, in a second step, the parametric controller was improved by a stochastic optimization process acting on the parameters of the controller.

However, the empirical study presented in [9] showed that, despite a good generalization capability and global improvement in performance, the stochastic optimization process was detrimental to performance in the area where the initial controller learned from demonstration was already nearly optimal. In this paper, we focus on this specific issue. We show that the global optimization can be replaced by a more local approach that only acts where the performance must be improved.

The paper is organized as follows. In Section II, we present our LfD and stochastic optimization methods, together with some related work. In Section III, we present the design of the experiments. Results of the comparison between the global method and its local counterpart are given in Section IV. Finally, Section V summarizes the results and presents the perspectives of this work.

II. Methods

In this section, we describe the methods used in our work, summarized in Fig. 1.

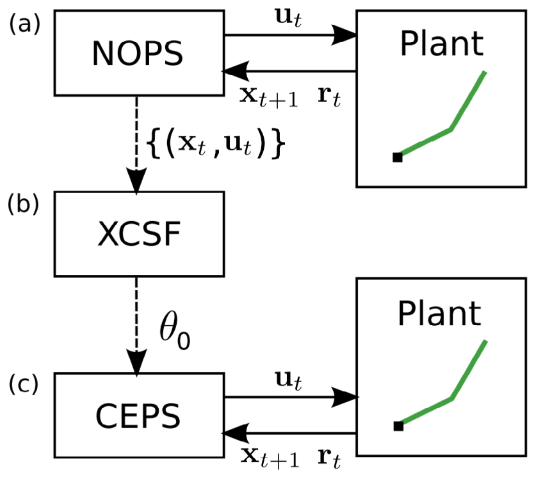

(a)
NOPS
$\mathbf{u}_t$
Plant
$\mathbf{x}_{t+1}$ $\mathbf{r}_t$
$\{(\mathbf{x}_t, \mathbf{u}_t)\}$
(b)
XCSF
θ_0
Plant
(c)
CEPS
$\mathbf{u}_t$
$\mathbf{x}_{t+1}$ $\mathbf{r}_t$

Fig. 1. General architecture of the experiments. The nature and role of each box is detailed in the text.

A. Generating near optimal control with the NOPS

In [12], the authors proposed a model of human reaching movements based on the OC paradigm. The model is based on

Didier Marin (PhD candidate in Robotics) and Olivier Sigaud (Professor in Computer Science) are with:
Université Pierre et Marie Curie
Institut des Systèmes Intelligents et de Robotique - CNRS UMR 7222
Pyramide Tour 55 - Boîte Courrier 173
4 Place Jussieu, 75252 Paris CEDEX 5, France
Contact: `firstname.name@isir.upmc.fr`

the assumption that HMC is governed by an optimal feedback policy computed at each visited state given a cost function that involves a trade-off between muscular effort and goal-related reward

$$r(\boldsymbol{s}_t, \boldsymbol{a}_t) = \alpha\|\boldsymbol{a}_t\|^2 - \beta g(\boldsymbol{s}_t) \tag{1}$$

where α is a weight on the effort term, g is a function that equals 1 at a target state $\boldsymbol{s}^*$ and is null everywhere else, and β is a weight on the reward term.

The corresponding near optimal deterministic policy is obtained through a computationally expensive variation calculus method. The feedback controller, resulting from the coupling of this policy with an optimal state estimator, drives the plant towards the rewarded state. Given that the policy does not take the presence of noise in the model into account, the actions must be computed again at each time step depending on the new state reached by the plant. Overall, generating a trajectory with this method is extremely costly. Hereafter, this controller is called Near-Optimal Planning System (NOPS). Indeed, the trajectories are not optimal in the strict sense, given the presence of non-modeled noise. A crucial feature of this model is that the generated movements do not depend on time. This results in the possibility to learn stationary policies from the model. In our framework, such policies are learned from demonstration with XCSF.

B. Learning from demonstration with XCSF

Learning Classifier Systems (LCSs) is a Machine Learning family of rule-based systems [16]. The XCS [2], [20] is an efficient accuracy-based LCS designed to solve classification problems and sequential decision problems. XCSF [21], [22] is an evolution of XCS towards function approximation.

As any LCS, XCSF manages a population of rules, called *classifiers*. These classifiers contain a condition part and a prediction part. In XCSF, the condition part defines the region of validity of a local model whereas the prediction part contains the local model itself. XCSF is a generic framework that can use different kinds of prediction models (linear, quadratic, etc.) and can pave the input space with different families of regions (Gaussian, hyper-rectangular, etc.). In the context of this paper, we only consider the case of linear prediction models and Gaussian regions.

A classifier defines a domain $\phi_i(\boldsymbol{z})$ and uses a corresponding linear model β_i to predict a local output vector $\boldsymbol{y}_i$ relative to an input vector $\boldsymbol{x}_i$. The linear model is updated using the Recursive Least Squares (RLS) algorithm, the incremental version of the Least Squares method.

The classifiers in XCSF form a population P that clusters the condition space into a set of overlapping prediction models. XCSF uses only a subset of the classifiers to generate an approximation. Indeed, at each iteration, XCSF generates a match set M that contains all reliable classifiers in the population P whose condition space $\mathcal{Z}$ matches the input data $\boldsymbol{z}$ i.e., for which $\phi_i(\boldsymbol{z})$ is above a threshold ϕ_0[1].

In XCSF, the output $\hat{\boldsymbol{y}}$ is given for a $(\boldsymbol{x}, \boldsymbol{z})$ pair as the sum of the linear models of each matching classifier i weighted by its fitness F_i

$$\hat{\boldsymbol{y}}(\boldsymbol{x}, \boldsymbol{z}) = \frac{1}{F(\boldsymbol{z})} \sum_{i=1}^{n_M} F_i(\boldsymbol{z})\, \hat{\boldsymbol{y}}_i(\boldsymbol{x}) \tag{2}$$

where $F(\boldsymbol{z}) = \sum_{i=1}^{n_M} F_i(\boldsymbol{z})$ and n_M is the number of classifiers in the match set M. In all other respects, the mechanisms that drive the evolution of P are directly inherited from XCS. In sum, XCSF is designed to evolve maximally accurate approximations of the learned function. A more complete description of XCSF can be found in [3], [4].

In our architecture, XCSF is used to learn a policy from demonstration. More precisely, a set of near-optimal state-action trajectories generated by the NOPS provides supervised learning samples, using the state of the plant as the condition and prediction space input and the action as the output on which regression is performed. By feeding XCSF with such samples (Fig. 1 (b)), we generate an action for any state within the range of the population of classifiers. Using a default action $\boldsymbol{a}_{default}$ for states that are not covered by the population (that is for which XCSF does not predict anything), we get a mapping from states to actions i.e., a deterministic policy. We call it the "XCSF policy" and note it $\pi_{\boldsymbol{\theta}_0}$. In a second step, it is improved by a direct policy search method called Cross-Entropy Policy Search (CEPS).

C. Improving the XCSF policy with CEPS

The XCSF policy $\pi_{\boldsymbol{\theta}_0}$ is parametric since each classifier has parameters in its condition and prediction parts. A standard way to optimize a parametric policy $\pi_{\boldsymbol{\theta}}$ given an objective function $J(\boldsymbol{\theta})$ consists in performing a gradient descent over $\boldsymbol{\theta}$ giving rise to *Gradient Policy Search* methods. This class of methods has attracted a lot of attention in Reinforcement Learning (see [11] for an overview). However, despite convincing results in robotics [7], [8], such methods are complex and sensitive to many hyper-parameters. An alternative family that gives better results comes from *probability-weighted averaging* methods such as CMA-ES, Cross-Entropy and PI^2 [1].

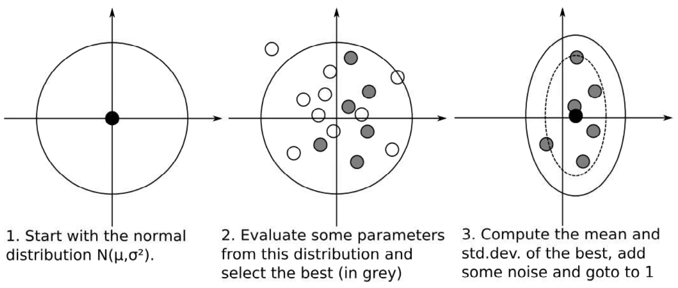

1. Start with the normal distribution N(μ,σ²).
2. Evaluate some parameters from this distribution and select the best (in grey)
3. Compute the mean and std.dev. of the best, add some noise and goto to 1

Fig. 2. Schematic view of the Cross-Entropy method.

By contrast with Gradient Policy Search methods, Cross-Entropy Methods (CEMs) [13], [14] do not assume that the objective function is differentiable or even continuous. CMA-ES is a more general evolutionary framework that contains CEMs as a special case [5]. More recently, PI^2 has been

[1]This threshold is named θ_m in [3]

proposed as another probability-weighted averaging method that is very similar to CEMs though it derives from very different first principles [19]. The general CEM is given in Alg. 1 and illustrated in Fig. 2. For more details, see [9].

A CEM can be applied straightforwardly to Policy Search, using the policy parameters $\boldsymbol{\theta}$ as solutions and a performance criterion J over targets as objective function. This results in the CEPS method [9]. $\pi_{\boldsymbol{\theta}_0}$ is adapted using CEPS (Fig. 1 (c)) where J is a discounted sum of costs $r(\boldsymbol{s}_t, \boldsymbol{a}_t)$ over trajectories (see 1). Each resulting policy $\pi_{\boldsymbol{\theta}}$ corresponds to a dot (i.e. a CEPS sample) in Fig. 2. In practice, $\boldsymbol{\theta}$ only contains the weights of each local model β_i.

Algorithm 1 CEM_iter

```
Require: $\{(\boldsymbol{\theta}_{(i)}, J_{(i)})\}_{i=0\cdots N}$: set of $N$ solution-value pairs
  $\rho$: proportion of the best solutions to use for the update
  $\sigma^2_{noise}$: additional noise term

  Sort the $\boldsymbol{\theta}_{(i)}$ according to $J_{(i)}$
  Compute the set $\mathcal{S}_\rho$ of the $\max(1, N \times \rho)$ best solutions
  $\boldsymbol{\mu} \leftarrow \text{mean}(\mathcal{S}_\rho)$
  $\boldsymbol{\sigma}^2 \leftarrow \text{std.dev}(\mathcal{S}_\rho) + \sigma^2_{noise}$
  return mean $\boldsymbol{\mu}$ and std.dev. $\boldsymbol{\sigma}^2$
```

Algorithm 2 "Global" Cross-Entropy Policy Search (GCEPS)

```
Require: $\{\boldsymbol{s}^*_{(j)}\}_{j=1\cdots M}$: set of target states
  $(\boldsymbol{\mu}_0, \boldsymbol{\sigma}^2_0)$: initial mean and std.dev of the $\boldsymbol{\theta}$ distribution
  $\rho$: proportion of the best samples to use for the update
  $\sigma^2_{noise}$: additional noise term
  $N$: number of sample policies to draw
  $K$: number of iterations

  for $k = 1 \cdots K$ do
    for $i = 1 \cdots N$ do
      Draw a sample $\boldsymbol{\theta}_{(i)} \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\sigma}^2_k)$
      for $j = 1 \cdots M$ do
        Perform an episode $\tau_j$ for target $\boldsymbol{s}^*_{(j)}$ following $\pi_{\boldsymbol{\theta}_{(i)}}$
      end for
      Compute the global performance of $\pi_{\boldsymbol{\theta}_{(i)}}$:
      $J_{(i)} = \frac{1}{M}\sum_{j=1}^{M}\sum_{t=0}^{|\tau_j|-1}\gamma^t r_{j,t}$ where $r_{j,t}$ is the reward at time $t$ for the episode $\tau_j$
    end for
    Perform a CEM iteration (Alg.1):
    $\boldsymbol{\mu}_{k+1}, \boldsymbol{\sigma}^2_{k+1} = CEM\_iter(\{(\boldsymbol{\theta}_{(i)}, J_{(i)})\}_{i=0\cdots N}, \rho, \sigma^2_{noise})$
  end for
  return optimized $\boldsymbol{\theta} = \boldsymbol{\mu}_{K+1}$
```

D. From global CEPS to local CEPS

In [9], CEPS was optimizing a global criterion: the mean performance over all targets. This "global" CEPS (Alg. 2), called GCEPS hereafter, induced a local loss of performance in the demonstration region where $\pi_{\boldsymbol{\theta}_0}$ was already good. We concluded that, to circumvent this effect, we should train $\pi_{\boldsymbol{\theta}_0}$ on one target at a time using a more local criterion

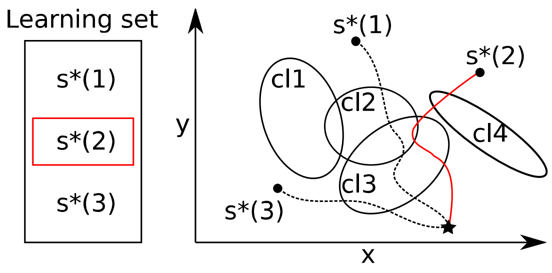


Fig. 4. Schema of the local policy parameters weighting process. The selected target is $\boldsymbol{s}^* = \boldsymbol{s}^*_{(k)}$ with $k = 2$ here. The central figure represents a two-dimensional state-space with some XCSF classifiers $cl_i, i \in \{1, \cdots, 4\}$. The weights w are computed as follows: first, for each target $\boldsymbol{s}^*_{(j)}$ in the learning set (left part), we perform an episode and get a state-space trajectory τ_j. Then, for each trajectory, LCEPS computes $n_{i,j}$ the number of times the classifier cl_i matches a state from the trajectory τ_j. Finally, the weight for all parameters associated which a classifier cl, noted w_{cl} (right part), is computed such that $w_{cl_i} = 1$ if $n_{i,k} > 0$, 0 else.

that would improve the performance only with respect to the current target. This local method should act preferentially on the classifiers that most need it, without disrupting P.

Thus, we propose here a "local" variant of CEPS (Alg. 3), called LCEPS hereafter, which is implemented as follows: each iteration begins by choosing a target state $\boldsymbol{s}^*$ for which a local update is performed using the CEM.

Algorithm 3 "Local" Cross-Entropy Policy Search (LCEPS)

```
Require: as in the global case
  for $k = 1 \cdots K$ do
    Choose a target $\boldsymbol{s}^*$ from the set $\{\boldsymbol{s}^*_{(j)}\}_{j=1\cdots M}$
    Perform an episode $\tau$ for target $\boldsymbol{s}^*$ following $\pi_{\boldsymbol{\mu}_k}$
    Compute weights $w[j] \in [0, 1]$ for each parameter $j$ based on their relevance during episode $\tau$
    Compute the std.dev. $\bar{\boldsymbol{\sigma}}^2$ such that $\bar{\boldsymbol{\sigma}}^2[j] = \boldsymbol{\sigma}_k$ if $w[j] > 0$ and $0$ otherwise
    for $i = 1 \cdots N$ do
      Draw a sample $\boldsymbol{\theta}_{(i)} \sim \mathcal{N}(\boldsymbol{\mu}_k, \bar{\boldsymbol{\sigma}}^2)$
      Perform an episode $\tau_{(i)}$ for target $\boldsymbol{s}^*$ following $\pi_{\boldsymbol{\theta}_{(i)}}$
      Compute the local performance of $\pi_{\boldsymbol{\theta}_{(i)}}$:
      $J_{(i)} = \sum_{t=0}^{|\tau_{(i)}|-1}\gamma^t r_t$ where $r_t$ is the reward at time $t$ for episode $\tau_{(i)}$
    end for
    Perform a CEM iteration (Alg.1):
    $\boldsymbol{\mu}', \bar{\boldsymbol{\sigma}}'^2 = CEM\_iter(\{(\boldsymbol{\theta}_{(i)}, J_{(i)})\}_{i=0\cdots N}, \rho, \sigma^2_{noise})$
    Update the $\boldsymbol{\theta}$ distribution using the local CEM result
    $\boldsymbol{\mu}_{k+1} = \boldsymbol{w} \times \boldsymbol{\mu}' + (1 - \boldsymbol{w}) \times \boldsymbol{\mu}_k$
    $\boldsymbol{\sigma}^2_{k+1} = \boldsymbol{w} \times \bar{\boldsymbol{\sigma}}'^2 + (1 - \boldsymbol{w}) \times \boldsymbol{\sigma}^2_k$
  end for
  return optimized $\boldsymbol{\theta} = \boldsymbol{\mu}_{K+1}$
```

In order to determine which parameters should be updated to improve the performance for $\boldsymbol{s}^*$, without a loss of performance elsewhere, LCEPS computes a weight vector $\boldsymbol{w}$ such that $w_j \in [0, 1]$ reflects the importance of the j^{th} policy parameter for the task associated to $\boldsymbol{s}^*$. The specific implementation of this weighting process for $\pi_{\boldsymbol{\theta}}$ can be found in Fig. 4. Then, $\boldsymbol{\theta}$ samples are drawn using the current normal distribution except

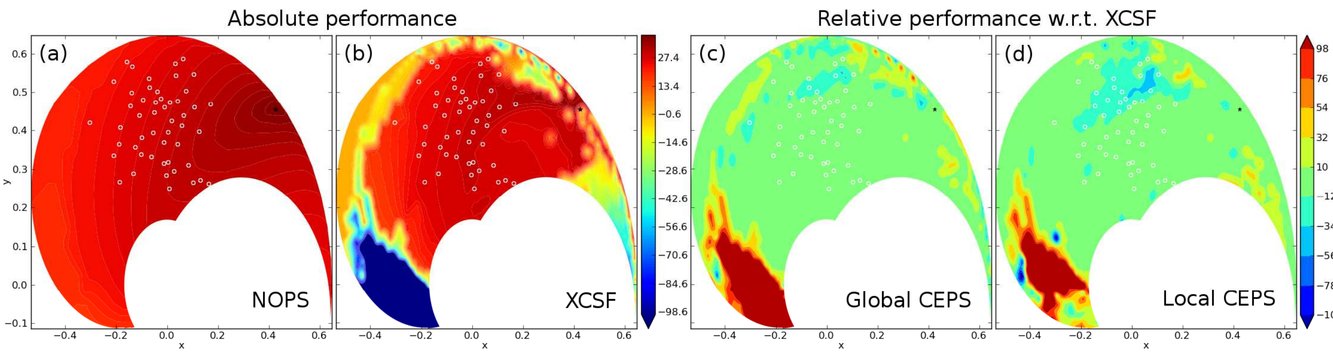


Fig. 3. Absolute performance of (a) the NOPS and (b) π_{θ_0}, and relative performance of π_{θ} optimized by either (c) GCEPS and (d) CEPS with respect to (b) π_{θ_0}. The performance is represented given the target position, obtained by interpolating the performances for a large grid of targets. The learning target positions are indicated by white circles, and the starting position by a star. The performance is computed according to (1) and represented as a color according to the right-hand side scale.

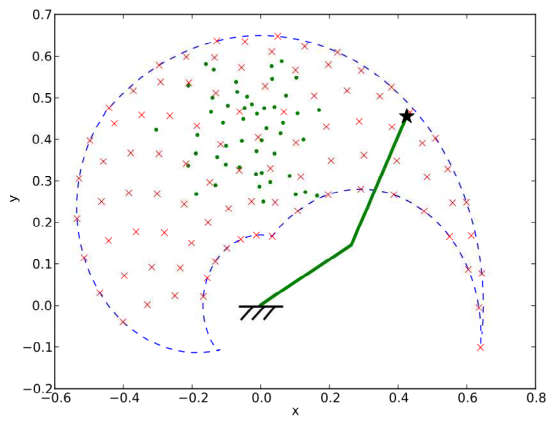


Fig. 5. The arm workspace. The reachable space is delimited by a spiral-shaped envelope. The two segments of the arm are represented by two bold lines. The initial configuration is the one represented. Learning targets are indicated by dots and testing targets by crosses.

for the parameters of null weight, for which the variance is set to 0. The corresponding π_{θ} are evaluated by performing an episode and fed to the CEM. Finally, the result of this "local" CEM iteration is used to update the θ distribution proportionally to the weight vector: the more relevant the parameter, the more it is taken into account in the new distribution.

III. Experimental design

We now illustrate the use of XCSF and CEPS for learning to control an arm to reach a given point with its end-effector. The plant is a simulated two degrees-of-freedom planar arm controlled by 6 muscles. The equations of the model and the specification of the control problem are given in [9].

A. Experiments

We perform three experiments.

1) Generalization with XCSF: First, we create two sets of target positions, $\mathcal{L}$ for learning versus $\mathcal{T}$ for testing. The $\mathcal{L}$ set contains 50 targets drawn according to a normal distribution centered on $x = -0.059$ and $y = 0.44$ and with standard deviation 0.1^2. The $\mathcal{T}$ set contains 94 targets in a 10×10 grid over the articular space (NOPS diverged for 6 of them, which were excluded). All trajectories start from the same point in the upper-right part of the reachable space (see Fig. 5). The NOPS is used to generate one trajectory for each target in $\mathcal{L}$. Then, XCSF learns from the actions of the NOPS for targets in $\mathcal{L}$, using the generated trajectories, and is tested as a policy on both sets. The condition and prediction spaces contain states $\boldsymbol{s}$. The performance and the trajectories obtained with π_{θ_0} are compared to those generated by the NOPS, to see how good π_{θ_0} generalizes to other targets.

2) Adaptation to a new target: Second, starting back from π_{θ_0}, we compare CEPS variants for improving the policy on a single target. This experience is performed for two targets: target A is located at the top-right of the workspace ($x = 0.2, y = 0.5$) and target B in the bottom-left corner ($x = -0.3, y = -0.5$).

3) Adaptation over the workspace: Third, we apply both CEPS variants for improving π_{θ_0} over $\mathcal{T}$. For LCEPS, we select a different target from $\mathcal{T}$ after each iteration.

B. Experimental set-up and Parameters

We use the JavaXCSF [18] implementation of XCSF, and all algorithms are implemented in Java. Computation times are measured on an Intel Core 2 Duo E8400 @ 3 GHz with 4 GB RAM. XCSF is tuned as follows. The number of iterations is set to $1,000,000$ and the maximum size of the population to 500. The input are normalized: the target and current positions are bounded by the reachable space and the speed is bounded by $[-100, +100]$ $rad.s^{-1}$. The default action $\boldsymbol{a}_{default}$ is set to a vector of zeros i.e., no muscular activation. After tuning empirically the parameters, the learning rate α (named `beta` in JavaXCSF) is set to 1.0, and compaction is disabled. For both CEPS variants, the number of iterations K is such that the method runs up to a time limit of 230 minutes. Each CEM iteration contains $N = 100$ policies. The proportion of selected episodes ρ is set to 0.8, i.e., the $N \times \rho = 80$ best policies are used for the update. The initial variance $\boldsymbol{\sigma}^2$ is set to 0.1 and the additional noise $\sigma^2_{noise} = 10^{-3}$.

IV. Results

In this section, we first study whether the policy learned with XCSF is similar to the one obtained with the NOPS for

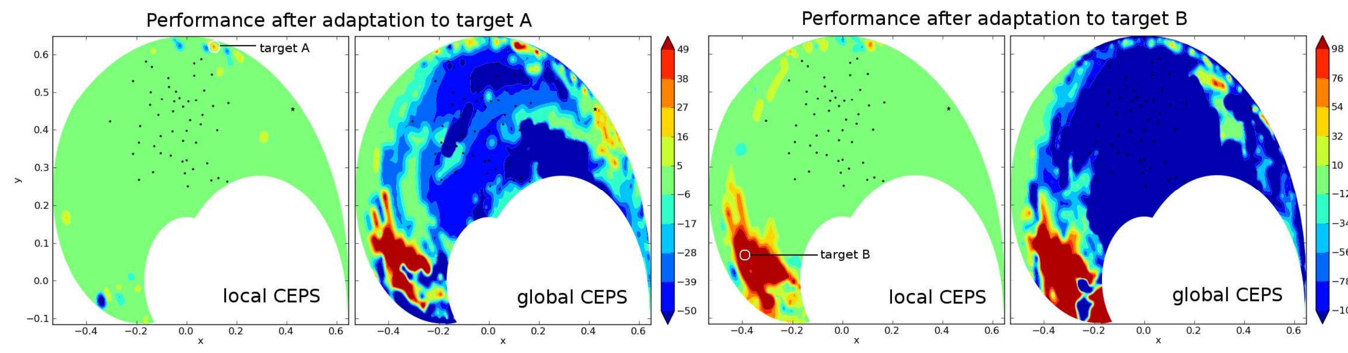


Fig. 6. Relative performance after applying GCEPS and LCEPS for adaptation of the XCSF policy to either target A or B.

TABLE I
MEAN AND STANDARD DEVIATION OF THE PERFORMANCE OVER TARGETS, BOTH FOR $\mathcal{L}$ AND $\mathcal{L}$

	$\mathcal{L}$	$\mathcal{T}$
NOPS	28.22 ± 2.06	27.46 ± 3.73
XCSF	27.45 ± 2.35	2.44 ± 46.03
GCEPS	22.96 ± 11.50	12.76 ± 22.28
LCEPS	25.19 ± 7.97	7.52 ± 38.76

TABLE II
MEAN AND STANDARD DEVIATION OF THE PERFORMANCE OVER TARGETS A AND B AND OVER $\mathcal{L}$

Algo.	target A	$\mathcal{L}$	target B	$\mathcal{L}$
NOPS	28.01	28.22 ± 2.06	21.17	28.22 ± 2.06
XCSF	-30.59	27.45 ± 2.35	-204.3	27.45 ± 2.35
GCEPS	27.98	-14.32 ± 12.49	22.88	-293.2 ± 169.7
LCEPS	26.14	27.40 ± 2.39	16.68	27.63 ± 2.01

the same targets, how well XCSF generalizes over different targets, and whether GCEPS is able to improve the performance of the learned policy. Then we compare the performance of LCEPS versus GCEPS, focusing on the improvement where generalization resulted in a poor performance and on the impact on the performance where it was already good.

A. *Performance of XCSF policy and generalization*

Results presented in this section are similar to those presented in [9], though we used only 500 classifiers instead of 6400 in [9]. The average running time to get one trajectory from the NOPS is $\sim$ 10 minutes. From $\pi_{\boldsymbol{\theta}_0}$, it is $\sim$ 30 milliseconds. Fig. 3(a) shows the performance of the NOPS as a function of the target position, obtained by interpolating the performance of the NOPS trajectories for all targets in $\mathcal{T}$. Fig. 3(b) shows the performance of one representative $\pi_{\boldsymbol{\theta}_0}$, which is very close to the NOPS performance for targets located near $\mathcal{L}$. One can see that the relative performance decreases with the distance to $\mathcal{L}$.

Fig. 3(c) shows the performance of $\pi_{\boldsymbol{\theta}}$ after applying GCEPS to optimize its parameters. One can see that the performance is globally improved over the whole workspace. In particular, the region around $(x = -0.3, y = 0)$ where the performance of $\pi_{\boldsymbol{\theta}_0}$ was very poor is improved a lot. Results in Table I confirm quantitatively this visual feeling.

B. *Performance of the GCEPS versus LGCEPS*

The rationale behind proposing LCEPS was to provide a way to improve the performance locally, where it is needed, without disrupting the performance over targets for which the controller is already good. In order to evaluate this specific property, we performed the experiments described in Section III-A2. The results of these experiments can be visualized in Fig. 6, that gives the relative performance of the controller resulting from LCEPS with respect to $\pi_{\boldsymbol{\theta}_0}$. In Fig. 6(a), one can see that the performance on target A is increased without major changes elsewhere, despite some local decrease of performance in areas that are not close in the Cartesian workspace but in fact correspond to related articular configurations. Fig. 6(c) show the same for target B, with a much larger increase in performance around the target. This can be explained by the fact that $\pi_{\boldsymbol{\theta}_0}$ was particularly far from optimality around target B. The numerical results corresponding to this adaptation are shown in Table IV-B.

Interestingly, with GCEPS, one cannot train the controller over a specific target while measuring the performance over the whole workspace. If we evaluate the controller just on one target, then GCEPS optimizes the whole classifier population so as to maximize the performance on that specific target, which results in a very large disruption of performance over the workspace, as shown in Fig. 6(b) and (d).

Thus LCEPS is capable of improving the performance where needed, with only minor effects on the performance elsewhere. Furthermore, performing iterations of LCEPS over a set of targets is much faster than an iteration of GCEPS over the same set because LCEPS optimizes for a restricted set of parameters.

Given these positive results, we now compare the performance of GCEPS and LCEPS over $\mathcal{T}$. In order to do so, we apply LCEPS by choosing each target in $\mathcal{T}$ in sequence, as we do when performing a GCEPS iteration. Results can be visualized in Fig. 7 and extracted from a comparison between Fig. 3(c) and Fig. 3(d), as well as from Table I.

As one can see, LCEPS is not significantly better than GCEPS over the whole workspace, even if the good performance in the training region is more preserved. This is mainly because iterating over all targets as in GCEPS does not make full profit of the capabilities of LCEPS. In the future, instead of iterating over all targets with LCEPS, we plan to combine it with an *active learning* strategy that will spend more computational

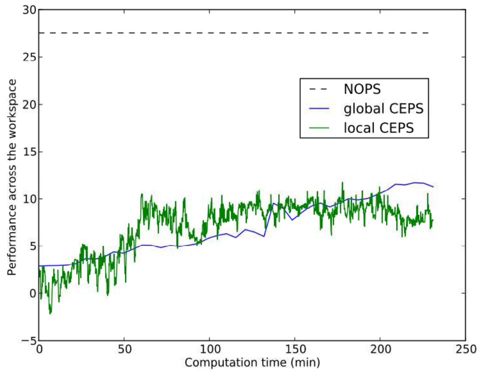


Fig. 7. Performance of the NOPS and $\pi_{\boldsymbol{\theta}}$ adapted by either GCEPS or LCEPS, as a function of time.

resources on targets where the performance can be more improved, and less resources where the performance is already good. Defining this *active learning* strategy is the matter of immediate future work.

V. Conclusion

In this paper, we have shown that learning and optimizing a parametric controller is a convincing alternative to using optimal control methods that are generally too expensive for real-time application in robotics. Here, using $\pi_{\boldsymbol{\theta}_0}$ was about 20000 times faster than using the NOPS. The resulting controller is fast and the generalization capability of LfD methods makes the learning process reasonably easy in practice. Furthermore, we have shown that using a stochastic optimization algorithm could further improve the policy, particularly in the areas where the performance is far from optimal.

However, from our empirical results, one can see that the obtained performance after a limited stochastic optimization period is still not close enough to the performance one gets with an OC method. There are two possible answers to this limitation. One consists in considering a *life-long learning* approach, where the system is optimizing its controller parameters throughout his life-time, yielding the slow convergence to a better optimum. Another option consists in learning a model of the plant, as presented in [15], [17] for instance, and use this model to improve the policy offline with virtual experiments based on the learned model. This will also help scaling the method to systems with more dimensions.

Our most immediate future work will consist in applying our approach to an assistance robot. We are designing a robot that will help motor-impaired patients to transfer from a seated position to a standing position with as few effort as possible. With the methods presented here, we will adapt in real-time the controller of the robot on-line to specific patients [10]. On a longer term, we would like to apply these techniques to the whole body control of humanoid robots like iCub, so that they perform more human-like movements.

Acknowledgments

This work was supported by the Ambient Assisted Living Joint Programme of the European Union and the National Innovation Office (DOMEO-AAL-2008-1-159), more at http://www.aal-domeo.eu.

References

[1] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal. Learning variable impedance control. *The International Journal of Robotics Research*, 30(7):820–833, 2011.

[2] M. V. Butz, D. E. Goldberg, and P.-L. Lanzi. Computational Complexity of the XCS Classifier System. *Foundations of Learning Classifier Systems*, 51:91–125, 2005.

[3] M. V. Butz and O. Herbort. Context-dependent predictions and cognitive arm control with XCSF. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1357–1364. ACM New York, NY, USA, 2008.

[4] M. V. Butz, T. Kovacs, P. L. Lanzi, and S. W. Wilson. Toward a theory of generalization and learning in XCS. *IEEE Transactions on Evolutionary Computation*, 8(1):28–46, 2004.

[5] V. Heidrich-Meisner and C. Igel. Similarities and differences between policy gradient methods and evolution strategies. In *Proceedings of the 16th European Symposium on Artificial Neural Networks (ESANN)*. Citeseer, 2008.

[6] S. Ivaldi, M. Fumagalli, F. Nori, M. Baglietto, G. Metta, and G. Sandini. Approximate optimal control for reaching and trajectory planning in a humanoid robot. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1290–1296. IEEE, 2010.

[7] J. Kober and J. Peters. Policy search for motor primitives in robotics. *Advances in Neural Information Processing Systems (NIPS)*, pages 1–8, 2008.

[8] P. Kormushev, S. Calinon, and D. G. Caldwell. Robot Motor Skill Coordination with EM-based Reinforcement Learning. In *Proc. of IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS-2010)*, 2010.

[9] D. Marin, J. Decock, L. Rigoux, and O. Sigaud. Learning cost-efficient control policies with xcsf: generalization capabilities and further improvement. In *Proceedings of the 13th annual Conference on Genetic and Evolutionary Computation*, pages 1235–1242. ACM, 2011.

[10] V. Pasqui, L. Saint-Bauzel, and O. Sigaud. Characterization of a least effort user-centered trajectory for sit-to-stand assistance user-centered trajectory for sit-to-stand assistance. In *Proceedings of the Iutam Symposium on Dynamics Modeling and Interaction Control in Virtual and Real Environments*, pages 196–203. IUTAM, june 2010.

[11] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks : the official journal of the International Neural Network Society*, 21(4):682–97, 2008.

[12] L. Rigoux, O. Sigaud, A. Terekhov, and E. Guigon. Movement duration as an emergent property of reward directed motor control. In *Proceedings of the Annual Symposium Advances in Computational Motor Control*, 2010.

[13] R. Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997.

[14] R. Y. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999.

[15] C. Salaün, V. Padois, and O. Sigaud. Control of redundant robots using learned models: an operational space control approach. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 878–885, 2009.

[16] O. Sigaud and S. Wilson. Learning classifier systems: a survey. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 11(11):1065–1078, 2007.

[17] P. Stalph, J. Rubinsztajn, O. Sigaud, and M. Butz. A Comparative Study: Function Approximation with LWPR and XCSF. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 1863–1870, 2010.

[18] P. O. Stalph and M. V. Butz. Documentation of JavaXCSF. Technical report, COBOSLAB, 2009.

[19] E. Theodorou, J. Buchli, and S. Schaal. Reinforcement learning of motor skills in high dimensions: a path integral approach. In *International Conference on Robotics and Automation*, pages 2397–2403. IEEE, 2010.

[20] S. W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[21] S. W. Wilson. Function approximation with a classifier system. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 974–981, San Francisco, California, USA, 2001. Morgan Kaufmann.

[22] S. W. Wilson. Classifiers that Approximate Functions. *Natural Computing*, 1(2-3):211–234, 2002.

Object Transportation by a Human and a Mobile Manipulator: a dynamical systems approach

Tiago Malheiro*, Toni Machado*, Sérgio Monteiro*, Wolfram Erlhagen† and Estela Bicho*
*Department of Industrial Electronics & Centre Algoritmi
†Department of Mathematics and Applications & Centre of Mathematics
University of Minho
Email: *{tmalheiro, tmachado, sergio, estela.bicho}@dei.uminho.pt †wolfram.erlhagen@mct.uminho.pt

***Abstract*—In this paper we address the problem of human-robot joint transportation of large payloads. The human brings to the task knowledge on the goal destination and global path planning. The robot has no prior knowledge of the environment and must autonomously help the human, while simultaneously avoiding static and/or dynamic obstacles that it encounters. For this purpose a dynamic control architecture, formalized as a coupled system of non-linear differential equations, is designed to control the behavior of the mobile manipulator in close loop with the acquired sensorial information. Verbal communication is integrated that allows the robot to communicate its limitations. Results show the robot's ability to generate stable, smooth and robust behavior in unstructured and dynamic environments. Furthermore, the robot is able to explain the difficulties it encounters and thus contribute to success of the task and to enhance the human-robot physical interaction.**

I. Introduction

Human-Robot cooperation is one of the key technologies to broaden the application field of robotics. By taking advantage of the strengths of both human and robot, more elaborated tasks, incapable of being performed by only one of the agents (robot or human) or even a team of robots, are now viable.

One of the problems addressed in Human-Robot cooperation concerns joint transportation of large objects [1], [2], [3]. In this task, the human may contribute by bringing intelligence and experience, global task knowledge, such as end goal and path planning, while the robot allows the execution of the task by autonomously helping the human to transport the object while avoiding static and/or dynamic obstacles that may appear in its navigation path.

The use of a mobile manipulator, instead of a simple mobile robot as in [2], allows a particular movement of the human partner to be followed by the robot by moving the manipulator, the mobile platform or a combined motion of both. Furthermore, the manipulator enables for a faster response of the robot to human movements, while at the same time allowing it to grasp and hold the object to be carried.

Despite the great potential of this Human-Robot cooperation for transportation tasks, the unknown and changing environment demands a complex dynamic behavior from the robot. Furthermore, this task also requires a close physical interaction with a human, thus the robot must exhibit a stable and smooth behavior that is influenced by the human's movements. On the other hand, the robot's movements may also influence the human partner. The so called Dynamics Approach to Behavior Generation [4] has been extensively and successfully implemented in mobile robot navigation [5], [6], multiple robots coordination for object transportation and formations [7], [8], mobile manipulator navigation [9], and Human-Mobile robot object transportation [2]. It is based on the mathematical theory of non-linear dynamics and provides a theoretical framework and tools that allow the design of control architectures that generate the behavior of the (cooperating) robots.

In this paper we aim to extend the work developed in [2], where the robot consisted of a mobile platform with a 2 DoF (rotational and prismatic passive joints) support on top of it and verbal communication was not considered, into the domain of Human-Mobile Manipulator cooperation, which must integrate non-verbal and verbal communication. The remainder of the paper is organized as follows. Section I-A provides a summary of the related work to control a robot in an object transportation task with a human. The mobile manipulator is described in section II, followed by a dynamical architecture for the control of the mobile manipulator in section III. In section IV we present some experiments, followed by the conclusions in section V.

A. Related Work

Passive robotics concepts were proposed in [10]. In [11], the authors extended it to accomplish a physical interaction between human and robot. Despite the continuous development [12] there are some scenarios were the necessary force to keep the object on track may have a certain direction and magnitude such that the passive robot does not have enough brake units to generate the desired force for supporting the human's motion.

In [13], the authors proposed a decentralized control algorithm to distributed robot helpers (DR Helpers). Each robot is controlled as if it has caster-like dynamics and interacts with the human through an intentional force/moment, that the human applies to the object. However, by using such approach, the human may not be able to precisely apply a moment to the human's grasping point of the object in order to adjust its orientation to keep it on track and avoid obstacles, [14]. To overcome this problem, in [14] it was suggested a system where each robot has a map information of the environment, thus, enabling it to generate a path on the known environment and move along with a path velocity based on the intentional force applied by the human. Despite the good performance, the applicability of such systems is very limited, since no changes are allowed to the environment or task, and dynamic obstacles could not be modeled.

Compliant motion has been addressed by several researchers [15], [16], [17] as an approach for robot-environment and human-robot interaction, where both the dynamic behavior and the position of the manipulator are controlled based on the concept of mechanical impedance [15]. By building on previous work [18], [17] propose a variable impedance control where the impedance characteristics were changed according to the speed of motion, and it was verified that the human was able to perform quick actions, with the same controller having a positioning action without oscillations.

The problem of side-slip in impedance control proposed in [17] was considered in [3], where a method that suppresses the side-slip of the object by assigning a virtual nonholonomic constraint at the robot hand is presented. This obliges the human to steer the manipulation as a cart or wheelbarrow.

In [19], the authors proposed an approach where no prior planning was assumed by the robot, and the human takes the lead while the mobile manipulator helps support the object and follows the human. They considered force control in the inertial z-axis direction to sustain the object while the manipulator is left free-floating inside the xy-plane. This way, the manipulator is dragged in the xy-plane by the friction force between the end effector and the object. They have used the manipulability measure of the manipulator as a criterion for optimal coordination of both platform and manipulator motions. However, the task may fail when the necessary force to drag the latter supersedes the friction force between the end-effector and the object.

A systematic derivation of the effort sharing policies was proposed in [1]. The authors considered three different policies where the effort (degree of assistance) is changed to tune the pro-activity of the robot. Besides the overall performance, a commonly known trajectory of the configuration was assumed.

Based on intention recognition, in [20] it was proposed an approach for active human-mobile manipulator cooperation, reducing the continuous human effort to maintain the movement, thus making transportation faster. Such is achieved by the search for spectral patterns in the force signal measured at the manipulator gripper. The system demonstrated satisfactory performance, nonetheless, it still needs an improvement in robustness [20] and robot's behavior does not integrate obstacles avoidance capabilities.

In active human-robot cooperation based on motion estimation [21], the position of the human hand is treated as the desired position of virtual compliance control. The human could manipulate the object as intended, however, the task was carried under limited movement of horizontal, one dimensional, transportation of the object.

A behavior-based approach was proposed in [2], where a dynamical control architecture, formalized as non-linear dynamical systems, controls the behavior of the autonomous mobile robot (without a manipulator) that must transport a large size object in cooperation with a human. In the proposed approach, the robot has no prior knowledge of the environment and the navigation is based on information of target orientation relative to the robot's heading direction, and obstacles orientation and distance. To note that, in the work reported in [2] verbal communication was not considered, which as we show here may enhance the physical interaction between human and robot. As the sensed world changes, the robot's behavior is adjusted accordingly. It was shown that the system is robust against perturbations, stable and the trajectories are smooth, while the robot helps the human to carry a long object in an unstructured indoor environment.

In this paper we extend that work in two ways: *i)* a mobile manipulator is used, for which a dynamic control architecture is developed; *ii)* verbal communication is integrated, that allows the robot to communicate its own difficulties, thus enhancing the execution of the task.

II. The Mobile Manipulator: Dumbo

A. Hardware

Dumbo robot is comprised of a differential mobile platform with a robotics arm coupled. The mobile platform makes use of 2 DOF for steering along with path velocity for translation, while the manipulator has 7 DOF (*amtec^{TM}lwa 7dof*) and its end-effector is a 1 DOF gripper. It features a laser range finder, URG-04LX, a 6 axis force/moment sensor between the end-effector and the arm, a speaker and a digital compass. All the hardware modules are connected to a computer, which supports all computational requirements. It is a Centrino M 1.7 GHz CPU, with 2 GB of RAM and 40 GB of hard disk drive. To suppress all connection requirements, a 4 port RS232 - PCI expansion card and a USB hub was added. Despite only having one computer, an Ethernet switch and wireless access point were added to allow connections from remote computers to debug and to monitor the task, while experiments are being performed.

B. Software

The on-board computer runs the Windows® Embedded Standard 7 operating system which has a network of modules, with an inter-process communication mechanism based on YARP [22], that manages the *sensorimotor* system. The software architecture is divided in: 1) a set of devices that comprise the Hardware Abstraction Layer and communicate directly with the hardware modules; 2) control application, which performs all the calculus required by the dynamical architecture, presented in section III, and connects to the devices to acquire sensory information and command the motors; 3) auxiliary software to monitor the task, such as the Matlab_Viewer, which is a GUI application developed under MATLAB® and allows the remote visualization and debug of the robot's internal dynamics.

III. Dynamical Architecture

A human operator and an autonomous mobile manipulator must, together, transport a long object in an unstructured and unknown environment. A *human-follower* motion control strategy is adopted where the human assumes the leadership of the task, since he/she knows the end goal of the task. The robot must be able to help him/her, for smooth and safe execution of the task, while avoiding static and/or dynamic obstacles that it may encounter.

As a first approach, the manipulator movements were not considered since these require inverse differential kinematics, which is very complex in redundant manipulators, like this one, and involves avoidance of singularities, whose calculations mean high computational costs. This implies that some flexibility is lost, however the benefit is that the control solution becomes simplified, easy to implement and fast to compute. A (sub)optimal posture was adopted which: 1) avoids joints limits and singularities; 2) places the end-effector at the height of the human partner and 3) allows free rotation on its wrist, parallel to ground. In this task, the grasping is non-compliant and it is assumed that the robot is already holding the object before the transportation starts and it is not unloaded at finish time. Such subtasks include: (a) the search for the object, e.g. with the vision system, (b) platform movement towards object location and (c) final grasping and lifting up maneuvers, using the manipulator.

The robot uses the direction to where the object is moving (read directly from the rotational joint on the wrist, $\Psi_{tar,R}$) and moment information from the force/moment sensor as non-verbal communication. Obstacles are sensed by the laser range finder. This information ($\Psi_{tar,R}$ and force/moment) is easily gathered with the adopted manipulator configuration. Nevertheless, it may also be acquired recurring to manipulator kinematics transformations if the manipulator is moving. Comparing Fig. 1 with Fig. 2 we can see that $\Psi_{tar,R}$ may be replaced by θ.

The platform's navigation is expressed in terms of angular velocity, $w = d\phi/dt$, and path velocity, v, [4]. Behavioral constraints are defined by directions of moving target or obstacles relative to a fixed world reference frame and restrictions on path velocity.

A. The dynamics of heading direction

1) Target acquisition: The target acquisition behavior is specified in the vector-field erecting an attractor (asymptotically stable state) at the orientation Ψ_{tar} (target direction, human, relative to a fixed world reference frame, see Fig. 1), and a repeller (unstable state) at the opposite direction, with strength λ_{tar}. Regardless of current heading direction, it is desired that the robot orientates itself towards this direction. Thus, this contribution should exhibit an attractive force over the entire range of heading direction. The mathematical form reads:

$$\frac{d\phi}{dt} = f_{tar}(\phi) = -\lambda_{tar}\sin(\phi - \Psi_{tar}) \tag{1}$$

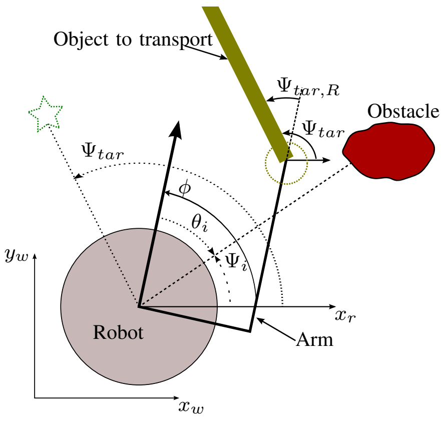


Fig. 1. Target acquisition and obstacle avoidance task constraints. Ψ_i is the direction at which the obstacle lies (the direction to be avoided) from the current position of robot. Ψ_{tar} is the desired (target) direction for the robot and it is the same in robot's center since the reference axis are kept parallel. The direction of x_r axis is kept parallel to a x_w during the robot's movements. This means that if the robot rotates about itself, the x_r axis will be parallel to x_w and so Ψ_i and Ψ_{tar} will be constant. If robot moves with translational velocity, x_r will move with it, but parallel to x_w, and Ψ_i and Ψ_{tar} will change accordingly.

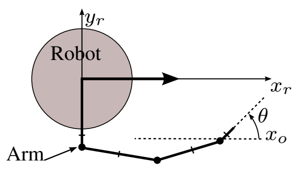


Fig. 2. With different arm configurations, θ is the angle in xy plane between x_o and end-effector direction, where $x_o // x_r$.

The current robot's heading direction ϕ is referenced to a fixed world reference frame, and $\Psi_{tar} = \phi + \Psi_{tar,R}$, thus, we have

$$\phi - \Psi_{tar} \quad = \quad \phi - (\phi + \Psi_{tar,R}) \quad = \quad -\Psi_{tar,R} \tag{2}$$

where $\Psi_{tar,R}$ is acquired directly from the encoder of the free rotational joint on the robot wrist. Since ϕ cancels out, target orientation does not need to be known relative to an external fixed world frame of reference and it is not influenced by calibration errors of the digital compass. This implies that (1) can be rewritten as

$$\frac{d\phi}{dt} = f_{tar}(\phi) = \lambda_{tar}\sin(\Psi_{tar,R}) \tag{3}$$

2) Obstacle avoidance: The obstacle avoidance behavior is expected to steer the robot away from obstacles that lie in its navigation path. Dumbo robot is equipped with a laser range finder, whose range was split in sectors. Each sector gets a fixed direction, θ_i (see Fig. 1), relative to the robot's heading direction and a $\Psi_i = \phi + \theta_i$ relative to the world reference axis. i denotes the number of the sector, ranging from $i = 1 \ldots n$, where n represents the number of sectors. For each sector, the dynamics should erect a repeller at the direction Ψ_i:

$$\begin{aligned} f_{obs,i} &= \lambda_{obs,i}(\phi - \Psi_i)\exp\left[-\frac{(\phi-\Psi_i)^2}{2\sigma_i^2}\right], i = 1 \ldots n \\ &= -\lambda_{obs,i} \times \theta_i \times \exp\left[-\frac{\theta_i^2}{2\sigma_i^2}\right] \quad\quad , i = 1 \ldots n \end{aligned} \tag{4}$$

As in (1) for target acquisition behavior, here, also, only the relative orientation θ_i of each sector i to robot's heading direction ϕ appears in the dynamics of heading direction. Thus the obstacle avoidance behavior is also not affected by calibration errors. In (4), $\lambda_{obs,i}$ is the strength of repulsion at direction Ψ_i and σ_i is the range of such repulsion. The mathematical form for these two parameters can be found in [4]. The final obstacle avoidance dynamics is obtained from the sum of the contribution of each sector $i = 1 \ldots n$:

$$\frac{d\phi}{dt} = F_{obs}\left(\phi\right) = \sum_{i=1}^{n} f_{obs,i}\left(\phi\right) = -\sum_{i=1}^{n} \lambda_{obs,i}\theta_i \exp\left[-\frac{\theta_i^2}{2\sigma_i^2}\right] \tag{5}$$

3) Integrating the two behaviors: The robot's behavior results from the following dynamical system, that integrates the two task constraints, follow the human and avoid collisions:

$$\begin{aligned} \tfrac{d\phi}{dt} = f_{robot}\left(\phi\right) &= -\lambda_{tar}\sin\left(\phi - \Psi_{tar,obs}\right) \\ &= \lambda_{tar}\sin\left(\Psi_{tar,R} + \Psi_{turn}\right) \end{aligned} \tag{6}$$

where $\Psi_{tar,obs}$ is the desired direction for the robot, given by:

$$\Psi_{tar,obs} = \Psi_{tar} + \Psi_{turn} \tag{7}$$

The angular value Ψ_{turn} is a function of obstacles contribution and results from a sigmoid function that rises smoothly from an inferior limit, $\dot{\phi}_i$, to a superior limit, $\dot{\phi}_s$, between a symmetric threshold value, ϕ_t:

$$\Psi_{turn} = \begin{cases} -\phi_t & \text{if} \quad F_{obs}\left(\phi\right) \leq \dot{\phi}_i \\ -\phi_t \cos\left(\pi \frac{F_{obs}(\phi) - \dot{\phi}_i}{\dot{\phi}_s - \dot{\phi}_i}\right) & \text{if} \, \dot{\phi}_i < F_{obs}\left(\phi\right) < \dot{\phi}_s \\ \phi_t & \text{if} \quad F_{obs}\left(\phi\right) \geq \dot{\phi}_s \end{cases} \tag{8}$$

$\dot{\phi}_i$, $\dot{\phi}_s$ and ϕ_t are design parameters.

The sigmoid function and the integration of obstacles contribution in target dynamics allows the control to take into account the target direction even in presence of obstacles, see Fig. 3. If a simpler integration of the two behaviors, such as in [4], would be used, when obstacles are near the robot, very strong repellers would be erected and the target contribution would be easily superseded by obstacles contribution, leading the robot to avoid the obstacles without taking into account the human direction in its behavior, dashed blue line in Fig. 3.

B. The dynamics of path velocity

To completely define the time courses of the robot behavioral variables, a dynamical system for path velocity should be specified. Every dynamic change in the sensed environment (because the robot moves or the environment changes over time) leads to a shift in attractors and repellers. Despite these shifts, the system must remain stable to guarantee the asymptotically stability of the overall control system. We must make sure that the control variables track one of the attractors as they move. In other words, the robot's heading direction should be in or near an attractor at all times [4]. Such task may be accomplished by controlling the path velocity, v, of the mobile platform:

$$\frac{dv}{dt} = g\left(v\right) = -c_{obs}\left(v - v_{obs}\right) - c_{tar}\left(v - v_{tar}\right) \tag{9}$$

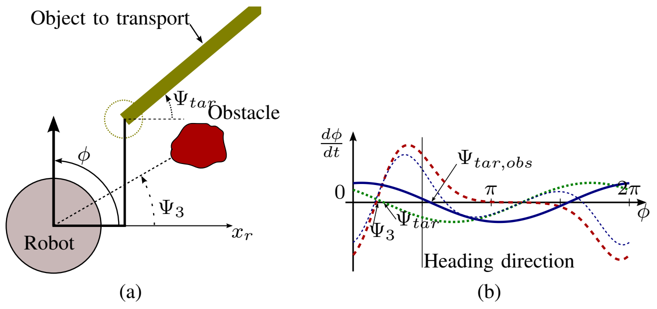


Fig. 3. Integration of Target and Obstacles behaviors. The sum of obstacles contributions, dashed red line, erects two different repulsive force-lets. Target acquisition dynamics contribution is represented by the dotted green line. The resultant non-linear dynamical system, solid blue line, has the same shape as target acquisition, but the attractor is shifted by the obstacles contribution.

The desired path velocity is controlled such that the presence of obstacles influences the velocity contribution of the target. c_i ($i = tar$ or obs) indicates the strength of each contribution [4]. When the potential function ($U\left(\phi\right)$), as explained in [4], is negative, no obstacles were detected, or the robot's heading direction is outside the repulsion zone. The robot must then navigate with a velocity, v_{tar}, proportional to the force/moment sensor readings. A positive value of $U\left(\phi\right)$ implies that the robot's heading direction is in a repulsion zone, and the robot must take into account the presence of obstacles. The path velocity, v_{obs}, is then controlled by the minimum between the velocity to follow the human (proportional to force/moment readings) and a velocity function of the distance to obstacles:

$$v_{obs} = \min\left(d_{min}/T_{2c,obs}, v_{tar}\right) \tag{10}$$

where d_{min} is the distance to the nearest obstacle and $T_{2c,obs}$ is a parameter defining the time to contact with the obstacle. The velocity dynamics presented in (9) guarantees smooth transitions between the velocities.

C. Speech

In order to ease task execution, the robot must communicate its own limitations, which may consist of physical or environment constraints. If the human enters a passage too narrow for a safe navigation of the robot, the previous dynamical architecture will not allow the robot to keep the movement, resulting in a dead end. A set of conditions, to the resultant of the dynamical systems, is applied, in order to address this situation, and others alike. This way, a narrow passage may be detected by, see Fig. 4:

$$narrow\_passage = \text{abs}\left(\alpha\left(\phi\right)\right) > 0 \wedge \text{abs}(\dot{\phi}) < \dot{\phi}_{min} \tag{11}$$

where $\alpha\left(\phi\right)$ is a sigmoidal threshold function, see [4], $\dot{\phi}$ is the current robot angular velocity and $\dot{\phi}_{min}$ is a design parameter. The instant that these conditions are verified, the robot synthesizes *"Wait! This passage is too narrow for me!"*. Similarly, detecting that the human is moving too fast and that the robot can not follow him is signaled by:

$$too\_fast = \text{abs}(v_{force} - v) > \Delta v_{max} \tag{12}$$

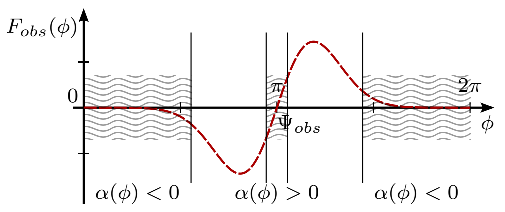


Fig. 4. In a narrow corridor, the resulting obstacles contribution has the shape presented above where, in the shaded areas, the rate of change of heading direction is too low, thus the robot moves slowly.

where $v_{force} = v_{tar}$ is the velocity attractor (when no obstacles are present) of the path velocity dynamics, which is proportional to the force applied by the human to the robot's end-effector. v is the current path velocity and Δv_{max} is a design parameter. When the distance between the desired velocity and actual velocity is higher than Δv_{max} threshold value, the sentence *"Wait! Go slower! I can not move so fast!"* is synthesized by the robot.

IV. Results and Discussion

The complete dynamical architecture was tested on the Dumbo robot in an Entrance Hall scenario. Static and dynamic obstacles are in the robot's navigation path and no predefined trajectory is given. In the implementation, the result of the heading direction dynamical system, $\dot{\phi} = w$ (6), is sent directly to the motors, while the result from the path velocity dynamical system, $\dot{v}$ (9), is integrated numerically using the forward Euler method. Fig. 5 shows a sequence of snapshots obtained from a video that can be downloaded from our MARL web server[1].

From 0s to 14s, the human performs a movement in a straight line leading the robot to follow him. At instant t=14s (Figure 5a), the human goes through a passage that is large enough for him but too narrow for the robot. The sensed obstacles lead to a decrease in the robot's speed, which is characterized by a value of 0.5 in the sigmoidal threshold function (magenta solid line in Figure 5b) at the robot's heading direction (solid vertical line). Thus, the desired robot's path velocity is the minimum between the velocity to follow the human and a velocity function of the distance to the obstacles, see (10). As the human continues his movement, the robot approaches the obstacles and detects, at instant t=24s, that the passage between these obstacles is too narrow for a safe movement, verbally alerting the human. The human acknowledges the robot's difficulty and starts to pursue another pathway. From instant t=24s (Fig. 5c) to t=50s (Fig. 5i), the human changes his navigation path, shifting the attractor along with it. Around instant t=48s, an object was thrown to the robot's path, with the resultant direction to be avoided (repeller) shifting along with the obstacle's movement. When the obstacle is near the current heading direction of the robot, path velocity is now governed by the distance to this obstacle.

[1]http://marl.dei.uminho.pt/Public/Robotica2012_Human-Robot_ObjTransp/

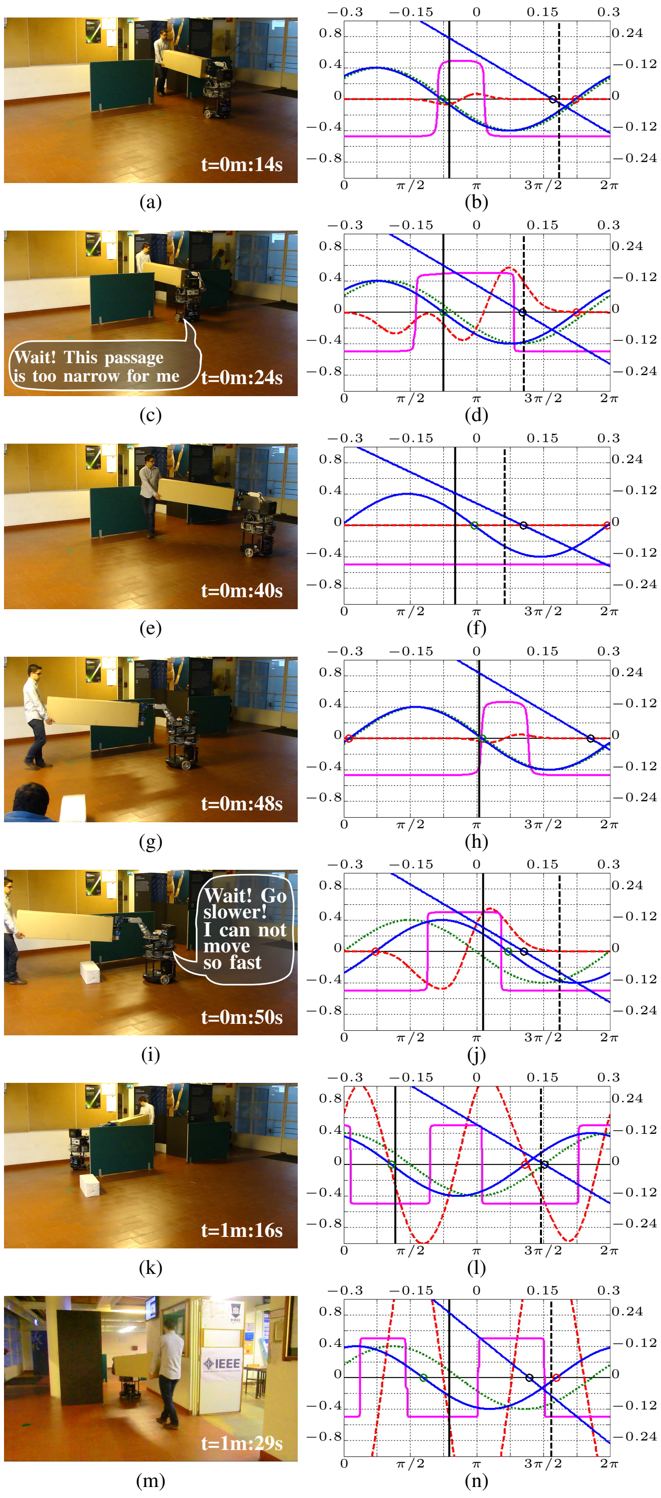


Fig. 5. On the right side are presented both heading direction and path velocity dynamics for the same instant. Left and lower axis are relative to heading direction, $d\phi/dt$ and ϕ, respectively. Right and upper axis are relative to path velocity dynamics, dv/dt and v, respectively. Vertical solid and dashed lines are heading direction and path velocity current values, respectively. Sinusoidal dotted green curve is the target acquisition contribution. Red dashed curve is the obstacles contribution and solid blue sinusoidal curve is the resultant behavior. Solid linear blue line is the path velocity dynamics. Solid magenta curve is the sigmoidal threshold function, see [4]. ϕ units is in radians, while v is m/s. Attractors and reppelers are represented by circles.

Since the human and the mobile manipulator were moving with a relatively high velocity, the robot can not continue to navigate with such velocity, alerting the human partner to the situation. The human acknowledges this instruction, reducing the exerted strength, and allowing the robot to safely avoid the obstacle, see Fig. 5i. As the human continues his movement, the target acquisition dynamics tries to align the robot's direction heading with the human and the obstacles avoidance dynamics tries to steer the robot through the passage, without colliding with the near by walls. As can be seen in instant t=1:16m, two strong repellers are erected at approximately $-\pi/2$ and $\pi/2$ from current heading direction (representing the contribution of both walls). Since the contribution of each wall is nearly the same, a resultant attractor is erected at the direction that keeps the robot equally spaced from both walls. The task ends with the human entering the room, Fig. 5m.

V. Conclusion

In this paper we have presented a dynamical control architecture that endows an autonomous mobile manipulator with the capability to help a human in a joint transportation task in dynamic environments. The robot's overall behavior is smooth and stable, where the information is locally gathered by the robot. The integration of speech synthesis, which allowed the robot to communicate its own difficulties, enhanced the execution of the task.

The issue of controlling simultaneously the movement of the robotic arm and mobile platform will be addressed in the very near future. It must be stressed out the need for the human to continuously exert strength to keep the robot in motion. An important next step will be to endow the robot with speech recognition, which may further enhance human experience and allow the inclusion of an adaptation/learning mechanism, enabling the robot to tune its internal design parameters on behalf of user's feedback.

Acknowledgment

This work was supported through fp6-IST2-EU-project JAST (project no 003747), Portuguese Science and Technology Foundation (FCT) and FEDER project COOPDYN (POSI/SRI/38081/2001). Tiago Malheiro and Toni Machado would also to thank the FCT for theirs Ph.D scholarships, ref. SFRH/BD/81334/2011 and SFRH/38885/2007, respectively.

References

[1] M. Lawitzky, A. Mörtl, and S. Hirche, "Load sharing in human-robot cooperative manipulation," in *RO-MAN, 2010 IEEE*, 2010, pp. 185–191.

[2] E. Bicho, L. Louro, N. Hipolito, S. Monteiro, and W. Erlhagen, "Motion control of a mobile robot transporting a large size object in cooperation with a human: a nonlinear dynamical systems approach," in *Proc. of the IEEE 11th Intl. Conf. on Advanced Robotics*, 2003, pp. 197–203.

[3] T. Takubo, H. Arai, Y. Hayashibara, and K. Tanie, "Human-Robot Cooperative Manipulation Using a Virtual Nonholonomic Constraint," *The International Journal of Robotics Research*, vol. 21, no. 5-6, pp. 541–553, 2002.

[4] E. Bicho, P. Mallet, and G. Schöner, "Target Representation on an Autonomous Vehicle with Low-Level Sensors," *The International Journal of Robotics Research*, vol. 19, no. 5, pp. 424–447, 2000.

[5] E. W. Large, H. I. Christensen, and R. Bajcsy, "Scaling the Dynamic Approach to Path Planning and Control: Competition among Behavioral Constraints," *The International Journal of Robotics Research*, vol. 18, no. 1, pp. 37–58, 1999.

[6] P. Althaus, "Indoor Navigation for Mobile Robots : Control and Representations," Ph.D. dissertation, Royal Institute of Technology, 2003.

[7] R. Soares, E. Bicho, T. Machado, and W. Erlhagen, "Object transportation by multiple mobile robots controlled by attractor dynamics: theory and implementation," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 2007, pp. 937–944.

[8] S. Monteiro and E. Bicho, "Attractor dynamics approach to formation control: theory and application," *Auton. Robots*, vol. 29, no. 3-4, pp. 331–355, 2010.

[9] L.-P. Ellekilde and H. I. Christensen, "Control of mobile manipulator using the dynamical systems approach," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*. Ieee, 2009, pp. 1370–1376.

[10] A. Goswami, M. A. Peshkin, and J. E. Colgate, "Passive robotics: an exploration of mechanical computation," in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, vol. 1, 1990, pp. 279–284.

[11] K. Fukaya, Y. Hirata, Z. Wang, and K. Kosuge, "Design and Control of A Passive Mobile Robot System for Object Transportation," in *Mechatronics and Automation, Proceedings of the 2006 IEEE International Conference on*, 2006, pp. 31–36.

[12] Y. Hirata, K. Suzuki, and K. Kosuge, "Improvement in the performance of passive motion support system with wires based on analysis of brake control," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 4272–4277.

[13] Y. Hirata and K. Kosuge, "Distributed robot helpers handling a single object in cooperation with a human," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 1, 2000, pp. 458–463.

[14] Y. Hirata, T. Takagi, K. Kosuge, H. Asama, H. Kaetsu, and K. Kawabata, "Map-based control of distributed robot helpers for transporting an object in cooperation with a human," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 3, 2001, pp. 3010–3015.

[15] N. Hogan, "Impedance Control: An Approach to Manipulation: Parts {I}, {II}, and {III}," *Journal of Dynamic Systems, Measurement, and Control*, vol. 107, no. 1, pp. 1–24, 1985.

[16] O. M. Al-Jarrah and Y. F. Zheng, "Arm-manipulator coordination for load sharing using reflexive motion control," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 3, 1997, pp. 2326 – 2331.

[17] R. Ikeura and H. Inooka, "Variable impedance control of a robot for cooperation with a human," in *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, vol. 3, 1995, pp. 3097–3102.

[18] R. Ikeura, H. Monden, and H. Inooka, "Cooperative motion control of a robot and a human," in *Robot and Human Communication, 1994. RO-MAN '94 Nagoya, Proceedings., 3rd IEEE International Workshop on*, 1994, pp. 112–117.

[19] Y. Yamamoto, H. Eda, and X. Yun, "Coordinated task execution of a human and a mobile manipulator," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 2, 1996, pp. 1006–1011.

[20] V. Fernandez, C. Balaguer, D. Blanco, and M. A. Salichs, "Active human-mobile manipulator cooperation through intention recognition," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 3, 2001, pp. 2668 – 2673.

[21] Y. Maeda, T. Hara, and T. Arai, "Human-robot cooperative manipulation with motion estimation," *Proceedings 2001 IEEERSJ International Conference on Intelligent Robots and Systems Expanding the Societal Role of Robotics in the the Next Millennium Cat No01CH37180*, vol. 4, pp. 2240–2245, 2001.

[22] P. Fitzpatrick, G. Metta, and L. Natale, "Towards long-lived robot genes," *Robotics and Autonomous Systems*, vol. 56, no. 1, pp. 29–45, 2008.

Challenges in the Application of Autonomous Cognitive Systems within Logistics

Marco Bonini, Alice Kirchheim, Wolfgang Echelmeyer
Reutlingen University
Reutlingen, Germany
esb-logistikfabrik@reutlingen-university.de

Abstract— **In order to enable the application of an autonomous cognitive system to the process of unloading of containers, the complexity of an industrial unloading scenario has to be simplified. In this paper is presented an approach whose first step is the identification of all the factors that make this application challenging. The state of the art in cognitive robotics needs to be exploited and, as it is developed in a software direction, in the second step of the approach all those factors which are concerned with hardware challenges are cut out. The presented result is a simplified scenario which still embodies those challenges to be fulfilled in the design of a cognitive software architecture. Next steps are also planned in order to lead the future research back to the original industrial complexity of unloading scenarios.**

Keywords - Cognitive System; Robotics; Logistics; Autonomous Unloading

I. INTRODUCTION

Complexity of scenarios in the process of unloading represents one of the main reasons for the application of cognitive robotics in the field of logistics. The development of a system capable of unloading all kinds of items from containers would have a huge impact on the service logistic process of unloading.
All definitions commonly agree that logistics is the planning and implementation of material flow and information flow within a supply chain. In logistics the particular branch of “service logistics includes the complete planning, controlling, realization and testing of all institution internal and overlapping flow of goods and personnel” [1]. Among the tasks of service logistics, the automation of unloading is the most challenging due to the undefined position and orientation of goods in carriers, especially considering the weights, dimensions and when surfaces are difficult to be handled [2].

Some single purpose solutions with a low level of cognition, aimed to perform the process of unloading autonomously, already exist as a market ready product. These systems are the result of the synthesis of progress in adjacent fields of research such as sensing, perception, grasping and planning. It is the case of the Parcel Robot, an autonomous system developed to unload parcels from containers [3]. The highest cognitive feature of this system is the decision about which parcel is easiest to grasp for each step of the unload procedure, in order to have an efficient and collision free trajectory for every movement of the kinematic chain of the robot. This feature stays however within the borders of the definition of an *autonomous robot system*. This is indeed a robot able to perform tasks in an unstructured environment; sensory inputs and actuators output allow such system to sense and modify the environment without continuous human guidance. This property, although necessary also for a *cognitive system*, it is not distinctive of it. Three more properties, coming from cognition in humans, have to be fulfilled by a system in order to be cognitive:

- learn from past experiences
- have long-term and short-term memory
- be able to modify and/or suppress reacting behaviours in order to reach future goals [4].

The development of a system with such characteristics would enable the automatic unload of goods which were previously manually unloaded. Efficiency is indeed guaranteed by the planner of the machine, which develops a plan for the sequence of items to be unloaded, aiming to the reduction of the cycle time for the whole unloading process. Such an algorithm has also to take into account the collision avoidance; therefore all the trajectories of the kinematic chain of the planned unloading sequences have to be verified before being executed. As this process of calculation and verification is not instantaneous, the algorithm has to be as simple and efficient as possible in order to reduce the idle time of the system.

As said, recently results coming from the progress in different fields of research have been synthesized and developed aiming towards industrial applications. In the state of the art of cognitive robotics there is however no branch yet which takes into consideration and develops specific challenges due to the application of cognition to logistics. In order to apply the state of the art it is first necessary that a simplification of the real scenarios is taken into consideration; then basic research methods have to be developed to tackle specific challenges that are due to the application in the logistic field.

This paper presents an approach for the simplification of a real industrial unloading scenario. The result is a simplified unloading scenario to which the state of the art in cognitive robotics can be applied; this will also be the starting point for the development of further basic research concerning cognitive robotics within logistics. Choice and positioning of items in the scenario will be explained in order to highlight how major challenges due to the application of cognitive robotics to logistics are embodied in the simplified scenario; guidelines for future steps are also given.

II. INDUSTRIAL UNLOADING SCENARIOS

Complexity that is faced in the unloading process of containers is due to two factors: unstructured environment and the variety of goods whose unload has to be accomplished. Both factors are concerned with the principle of *space-saving*, that lays the foundation of the process of loading containers. Transportation costs are indeed higher than unloading costs: containers are therefore preferred to be loaded in order to reduce the unused space, even though this increases the unloading process costs.

The *unstructured environment* could arise from two opposite phenomena:

- Necessity of exploiting the characteristic of deformability of the items to be loaded in order to save space. So far no autonomous systems have been developed in order to optimise the loading process exploiting deformability of items; this process is therefore accomplished by human operators, which explains the lack of a standard procedure for the loading process of containers. Therefore even two containers loaded in the same buffer and with the same homogeneous type of good might have a different packing pattern.
- Displacement of items due to the presence of empty space: goods could indeed move during the transportation, if they are not stuck in the container. This promotes an unstructured environment.

Variety of goods to be unloaded can be due to:

- a mix of products having in common the delivery to one customer, in order to reduce the number of intermediate handling and cross-docking of items
- a mix of products having in common the shipment source and addressed to more than one customers, in order to reduce the outgoing number of containers.

Due to the unstructured environment and to the variety of products that can be found, eight different possibilities of packing patterns can be found in an industrial unloading scenario. These possibilities arise from the combination of the two characteristics that each of the following factors have:

TABLE I. CHARACTERISTICS OF UNSTRUCTURED ENVIRONMENT IN A CONTAINER

Factors	Characteristics
Kind of goods	Homogeneous / Heterogeneous
Packing pattern	Neat / Loose
Frame situation	Stuck / Slack

In neat packing patterns goods have all the same orientation (even if heterogeneous) and the container looks tidy. On the contrary in a loose packing pattern goods are positioned in different orientations (even if homogeneous) and the container looks messy.

In a stuck frame situation force has to be applied to the goods to free them from the friction of touching surfaces and not only to lift them. In a slack frame situation only the force for lifting the goods has to be applied, as items are not stuck.

These are two examples of opposite situations:

- homogeneous goods neatly and tightly packed
- heterogeneous goods loosely and slackly packed

To visualize how complexity springs from an unstructured environment and the variety of goods, hereafter some figures of possible industrial scenarios are shown in Fig. 1, Fig 2 and Fig. 3.

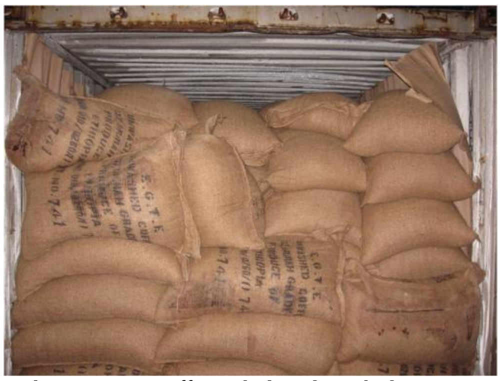


Figure 1: homogeneous coffee sacks loosely packed in a container

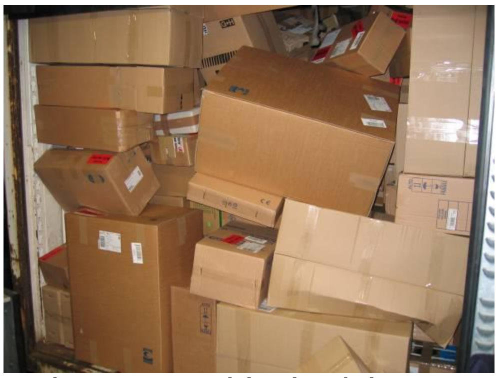


Figure 2: heterogeneous parcels loosely packed in a container

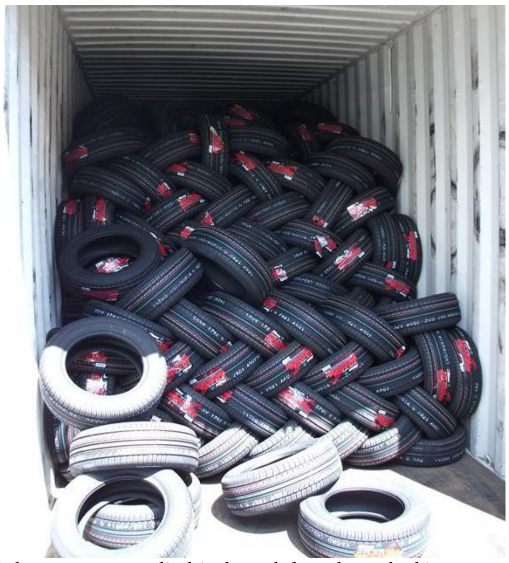


Figure 3: homogeneous cylindrical goods loosely packed in a container

III. Approach to the simplification of industrial complexity

In this section a two steps approach followed for the simplification of a real industrial unloading scenario will be explained. The first step is the identification of those factors that make it challenging to apply cognitive robotics in the unloading process; in the second step those factors whose related challenge concerns hardware more than software development are cut out.

A. Challenging factors and characteristics

Factors will be analysed in two clusters, which will then be taken into account for the presentation of the resulting simplified scenario.

1) Physical factors of the items to be unloaded

TABLE II. Physical Factors of Items

Factors	**Characteristics**			
Dimensions	Small (max dim. < 250 mm)	Medium (max dim. < 600 mm)		Big (max dim. < 1100 mm)
Weight	Light (W < 10 kg)	Medium (W < 35 kg)		Heavy (W < 70 kg)
Shape	Fixed		Variable	
Surface	Smooth	Porous		Damaged

The shape and the surface factors need further explanation and values, in order to identify the cut-off point between their characteristics.

Tolerances for a shape to be considered *fixed* or *variable* have to be found in order to avoid any ambiguity. For example a parcel is considered to have a *fixed* shape, even though the cardboard is flexible and elastically deformable; in the case that the cardboard would result in being plastically deformed, the parcel would be considered still to have a *fixed* shape, but with a *damaged* surface.

A surface has to be considered *smooth* when it is graspable through a suction cup gripper; when this is not possible, other gripping principles have to be applied.

Further details (i.e. values for forces and tolerances) for the cut-off point between the characteristics of factors are not relevant to be given in this paper.

2) Positional factors of items

TABLE III. Positional Factors of Items

Factors	**Case**	**Occlusion**	**Mutual dependency**
Into	A into B	A and B might be occluded	Movement of A might cause a movement of B and vice versa
Stuck (into/ between/ among)	A stuck into B	A and B might be occluded	Movement of A will cause a movement of B and vice versa
Onto (Underneath)	A onto B (B underneath A)	B might be occluded	Movement of B might cause a movement of A
In front of (Behind)	A in front of B (B behind A)	B might be occluded	Movement of B might cause a movement of A
Leant on	A leant on B	A and B might be occluded	Movement of B might cause a movement of A

An item is occluded when the vision system cannot recognise the object; this can be due to the presence of another item between the undetected object and the vision system sensor. As occluded, an item is not present for the system in the scenario; as the item that causes the occlusion is removed, the occluded good could become visible and detectable by the system. Occlusion is therefore a challenge for the planner of the system, because this has to quickly modify in each step the planned sequence of items to be unloaded in order to reach the final goal of optimization of the cycle time for the whole unloading process.

The scenario presented in the following section, arises from these hypothesises concerning the two clusters:

- *Heavy* weight items lead to hardware related challenges
- *Porous* and *damaged* surfaces, as well as *variable* shapes, create both software and hardware (gripping devices) related challenges
- *Occlusion* and *mutual dependency* in movement of the items, due to factors in cluster 2, create the major software related challenges.

B. Exclusion of non-software concerning factors

As the core of the state of the art in cognitive robotics is made by research in the software field, in this step of the approach, characteristics that affect hardware issues are cut out. Among the listed factors the weight is hypothesised to be affecting the hardware and just the low level software architecture (e.g. the acceleration values of the drives of the system that have to consider the inertia of the items). The characteristics *medium* and *heavy* of this factor are therefore cut out for the moment, considering only items which belong to the *light* category characteristic. Goods are however put in the scenario as real scale dimensions; this is done in order to ease sensing and perception tolerance errors, which are not worth to be refined in the first steps of this basic research method.

Challenges concerning the software architecture will be however guaranteed not only by the presence of items with different characteristics of shape, surfaces and dimensions, but also by their positioning, chosen in order to create *occlusion* and *mutual dependency* in movement.

The result is the simplified scenario presented in the next section.

Such a scenario is designed in order to test the capability of the system concerning:

- Handling of deformable items
- Handling surfaces-troubled items
- Planning and verifying an unloading sequence of items
- Optimization of the cycle time.

All of these tasks concern especially the high level software architecture of the system.

IV. Simplified unloading scenario

In this section a simplified unloading scenario is presented. Each good to be included in this scenario are described thoroughly according to the characteristics of the factors in cluster 1. The positioning of the goods in the scenario considers factors in cluster 2.

First the goods at play will be introduced, then an example of their set-up is given. The final part of this section is dedicated to the explanation of how goods satisfying cluster 1, positioned in respect to cluster 2, will guarantee the presence in the scenario of those challenges to be faced in order to design an efficient cognitive software architecture.

The design of a cognitive software architecture is a closed loop process: after the first code is developed, feedback from tests made on the presented scenario are needed in order to improve the software architecture iteratively. In order to reduce the time needed due to this closed loop design process, the number of items has been limited to six. This is indeed hypothesised to reduce the time required for the development, test and improvement of the software architecture. Three of these goods are industrial items, while the other three are not-industrial. Those items which are here classified as industrial are usually loose in containers, positioned in some kind of packing pattern, while those classified as not-industrial are usually packed in other boxes or pallets. Hereafter a description of goods in respect to characteristics in cluster 1 is given.

The industrial items are:

- Parcel, with *medium* dimensions (390x290x270 mm), *light* weight, *fixed* shape and *smooth* surface.
- Jute coffee sack, with *big* dimensions (950x625x225 mm), *light* weight, *variable* shape and *porous* surface.
- Car tire, here drafted as a P 205/55 R16, with *big* dimensions, *light* weight, *fixed* shape and *porous* surface.

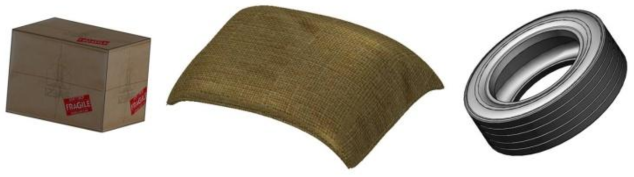


Figure 4: industrial items in the simplified scenario

The not-industrial items are:

- Microsoft Xbox Kinect box, with *medium* dimensions (376x149x122 mm), *light* weight, *fixed* shape and *smooth* surface. This being a market ready product for end users, the box is part of the good and cannot be ruined; it can therefore only be found packed into pallets or other boxes within containers.
- Teddy bear, with *small/medium* dimensions, *light* weight, *variable* shape and *porous* surface. Its exclusion from the industrial goods is due to the same reason as the Kinect box.
- Beer keg, with a capacity of 30 l, *medium* dimensions, *light* weight, *fixed* shape and *smooth* surface. This kind of keg is usually handled through special pallets and cannot be found loosely packed in containers.

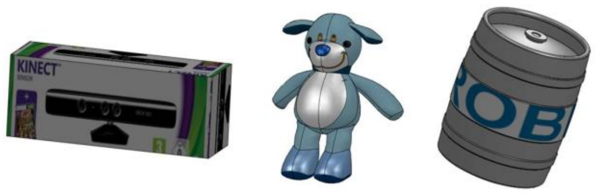


Figure 5: not-industrial items in the simplified scenario

The goods are put into the scenario considering their real scale; this in order to ease sensing and perception tolerance errors, which are not worth to be refined in the first steps of this basic research method. *Heavy* and *medium* categories of weight characteristics have been cut out, but it has to be noted that, besides the teddy bear and the Kinect box, potentially all the other four goods might be in the *medium* or *heavy* weight category characteristics. Hereafter some solutions to scale down the weights are stated case by case:

- The parcel and the beer keg can be empty or filled with some light material
- The coffee sack can also be filled with some light material which has to simulate the behaviour of a full sack, expanded polystyrene for instance

- The weight of the P 205/55 R16 tire can be around 10 kg. If this is considered to be a problem for the hardware available for the tests, this tire could be replaced, for instance, with a motor scooter tire.

The positioning pattern of these goods is fundamental to embody all of the challenges in the scenario; positions are indeed thought of in order to create *occlusion* and *mutual dependency* in movement between the items. Hereafter the procedure for setting up the scenario is given referring to factors in cluster 2.

1. Position the tire in flat position
2. Position the beer keg *into* the tire in an approximately concentric position

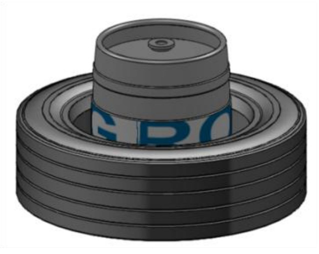


Figure 6: step 2 of the set-up of the simplified scenario

3. Position the Kinect box *onto* the sidewall of the tire and *in front of* the beer keg
4. Position the teddy bear *leant on* the tread of the tire and *behind* the beer keg, in a diametrically opposite position in respect to the Kinect box

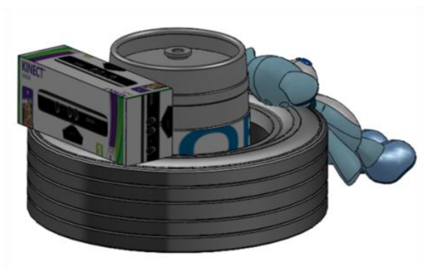


Figure 7: step 4 of the set-up of the simplified scenario

5. Position the parcel *leant on* the tread of the tire, for example on the right side of the Kinect box
6. Position the sack *onto* the top of the beer keg, in a way that it is also *leant on* the Kinect box, *on* the teddy bear and *on* the parcel.

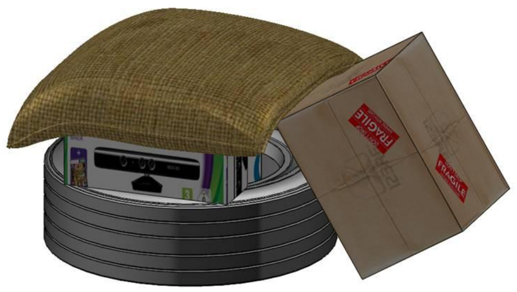


Figure 8: step 6 of the set-up of the simplified scenario

It is important to notice that after the positioning of the sack, both the parcel and the beer keg result to be *stuck between* the sack and the floor, while the Kinect box is *stuck between* the sack and the tire.

For a clearer understanding of the positions of the goods, different views of the set scenario have been reported hereafter in Fig. 9 and Fig. 10.

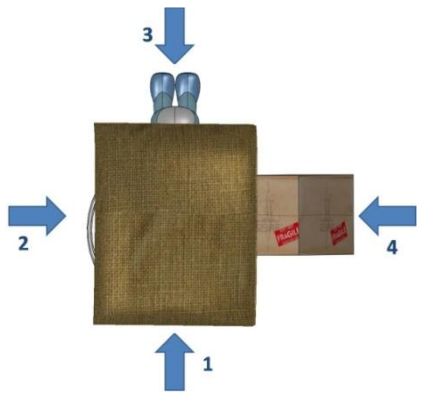


Figure 9: plan view of the set-up of the simplified scenario

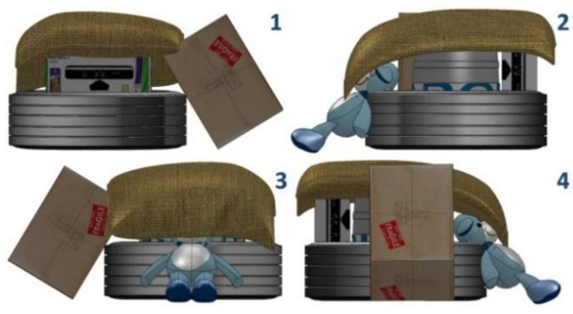


Figure 10: lateral views of the set-up of the simplified scenario

As said the positioning of items aims to create *occlusion* and *mutual dependency* in movement between items. The sack and the parcel are indeed the only two goods that are free from occlusion at first. All the other items are partially (Kinect box, teddy bear, tire) or totally (beer keg) hidden. During the unloading process occlusion can be removed or can arise. The arising of occlusion can be due to the mutual dependency in movement among the items; such a phenomena is easy to understand with the following example. Three hypotheses have to be made to build up the example:

- The tire is detected, by the object recognition system, through the pattern of the tread
- The planner decides to initialize the unloading sequence by grasping the Kinect box
- The sack, which is leant also on the Kinect box, once this is moved, it slippers in front of the tire, totally covering its tread.

This would have the following consequences:

- The parcel might fall and change position
- The tire is not detectable anymore by the system
- The beer keg and the teddy bear might be detectable as occlusion, previously caused by the sack, has been removed.

This example explains how such a scenario can be used for testing the capability of the object recognition system to detect disappearance and appearance of items due to the removed or arising occlusion, nonetheless the ability and rapidity of the planner in generating a new sequence of items whose unloading has to be accomplished.

V. Future steps

The future steps in order to design the software architecture of a cognitive system to be applied in the unloading process of containers are two: (A) the decision about the level of cognition needed and (B) a stepwise upgrade of the scenario, up to a real industrial one.

A. Level of cognition

Once a simplified unloading scenario has been designed, the level of cognition to assign to the system has to be decided. The range of possibilities is wide. The most simple and less cognitive is the development of a system that, as with the Parcel Robot, is able to detect in each step the easiest item to be unloaded and accomplish the unloading process with a collision free trajectory. The most complicated involves intelligent learning, both about grasping points of items to be unloaded and about the best sequence to be followed in order to optimize the cycle time. The level of cognition has to fulfil the compromise between a challenging and an applicable industrial goal.

B. Increasing complexity through a stepwise approach

In order to have a cognitive system able to accomplish autonomously the unloading of a container, both hardware and software have to be upgraded. The simplest way to do it is through a stepwise approach: each step considers one or more characteristics that will make the scenario more realistic and these are then introduced. Such process continues until then every challenging characteristics will be included. Weighs for example have to be scaled-up, which involves the design of a new hardware architecture. Also due to the workspace needed for the unloading of a container, such architecture will probably be the result from the integration of more systems. In case more systems have to be integrated, it is likely also the software architecture will have to be upgraded: each step of the increasing complexity approach involves further development of new parts of the machine, whose design however will not be a zero-based one. Results of tests conducted on one scenario in a step, will tell in which direction the complexity of the following step scenario has to be increased.

VI. Conclusions

An approach for the simplifications of the industrial complexity of unloading scenarios in logistics has been presented in this paper. It enables the application of the state of the art in cognitive robotics to the field of logistics. In the first step of this approach all the factors that make the application of cognition in logistics challenging have been identified. In the second step all those factors which are concerned with hardware challenges have been cut out. The result is a simplified scenario which still embodies those challenges to be fulfilled in the design of a first cognitive software architecture. Such a scenario is focused on the development of algorithms for the handling of deformable and surface-troubled items, as well as on the development of the algorithm used by the planner in order to create, verify and update a collision free sequence of items to be unloaded. Also the optimization of the whole unloading cycle time is considered. Goods in the scenario have been indeed chosen with challenging physical properties and they have been also positioned in such a way that occlusion and mutual dependency in movement are guaranteed. This will put to test the main cognitive characteristic of the system, which is the ability and rapidity of the planner to generate at each step a new and collision free sequence of items to be unloaded.

In further steps the first version of a software architecture will be developed and tested on the presented scenario. Results will be analyzed and will enable a stepwise upgrading process of the scenario, that will lead back to the original industrial complexity of unloading scenarios.

Acknowledgment

Supported by the European Commission through the Seventh Framework Programme for Research and Development under the Grant Agreement number 270350.

[1] H. Baumgarten. (2011, June 10). [Online]. Available: http://logistics.de/downloads/04/84/i_file_45167/Jahrbuch%202002%20-%20Dienstleistung.pdf

[2] Wolfgang Echelmeyer, Alice Kirchheim, Achim L. Lilienthal, Hülya Akbiyik, Marco Bonini, "Performance Indicators for Robotics Systems in Logistics Applications" [online]. Available: http://kaspar.informatik.uni-freiburg.de/~mmartlog/pdfs/papers/echelmeyer_et_al_mmartlog11.pdf

[3] ThyssenKrupp System Engineering. (2011, June 10). [Online]. Available: http://www.thyssenkrupp-systemengineering.com/cms/website.php?id=/587/299/en/productrange/service/roboticlogistics/parcelrobot.htm

[4] Miller, E. K. "The Prefronta Cortex and Cognitive Control", Nature Reviews: Neuroscience, 2000

Vision-based Hand Wheel-chair Control

Paulo Trigueiros
Departamento de Informática
Instituto Politécnico do Porto
Porto, Portugal
paulo@trigueiros.org

Fernando Ribeiro
Departamento de Electrónica Industrial
Universidade do Minho
Guimarães, Portugal
Fernando@dei.uminho.pt

Abstract — **Several studies have shown that people with disabilities benefit substantially from access to a means of independent mobility and assistive technology. Researchers are using technology originally developed for mobile robots to create easier to use wheelchairs. With this kind of technology people with disabilities can gain a degree of independence in performing daily life activities. In this work a computer vision system is presented, able to drive a wheelchair with a minimum number of finger commands. The user hand is detected and segmented with the use of a kinect camera, and fingertips are extracted from depth information, and used as wheelchair commands.**

Keywords - independent mobility, machine vision, wheelchair control, hand segmentation, finger control

I. Introduction

Several studies have shown that people with disabilities benefit substantially from access to a means of independent mobility and assistive technology [1], being independent mobility an important aspect of self-esteem [2].

Assistive devices such as powered wheelchairs improve one's quality of life. While the needs of many individuals with disabilities can be satisfied with traditional manual wheelchairs, some find it difficult to use.

To accommodate this population and even other segments, several researchers have used technologies originally developed for mobile robots to create user friendly easier to use wheelchairs and "smart wheelchairs". Smart wheelchairs typically consist of either a standard power wheelchair to which a computer and a collection of sensors have been added or a mobile robot base to which a seat has been attached [2].

This work presents a simple and effective HCI (human computer interface) giving the user the ability to easily control a robotic wheelchair with a minimum number of finger commands. The main goal consists of giving the user the capability to control it without touching any physical device. For that purpose, a computer vision interface was developed, able to detect fingertips and able to use that information for driving the wheelchair.

To extract the hand and fingertip localization a kinect [3] camera is mounted on the back of the wheelchair pointing down to the user's hand.

Machine vision is a promising sensor technology. With nowadays cameras, smaller than a lot of other sensors, they can be mounted in multiple locations, giving larger sensor coverage. Also, the cost of machine vision hardware has fallen significantly, and the solutions based on computer vision continue to improve.

II. Related Work

Several wheelchair control devices were studied as alternatives to traditional input methods.

Within the analogue control systems the joystick is by far the most common drive control [4] and it can be mounted for either right or left hand use. The joystick usually consists of a metal stick with a hard plastic head [5] that the user use to command the chair, an on/off switch, battery gauge, maximum speed control and sometimes a drive mode switch.

With the wheelchair chin control, the gimble is mounted on a swingaway mount of some sort and positioned slightly below and forward of the chin (Figure 1). Chin controls work much the same as conventional joysticks in that the user simply pushes the gimble the direction they want to move and control their speed with the distance they push the gimble. This system is designed for a user with good head control. [4].

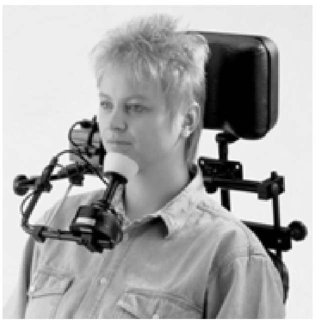


Figure 1 Chin control device, from MEYRA[1]

When set up to be actuated by the head, the gimble is mounted behind the head and attached to a headrest. The user pushes the headrest left to go left, right to go right and back to go forward (Figure 2). One drawback of this system is that the user cannot actually use the headrest, as a headrest, unless power to the chair is turned off. Another drawback of this set up is that the user must activate a switch to be able to move backwards, and activate the switch again to move forward. Normally this is not a serious drawback, but if the user is in a situation where several back and forward movements are required to get through a doorway or enter an elevator etc., it can be quite annoying to have to activate the forward/reverse switch so often.

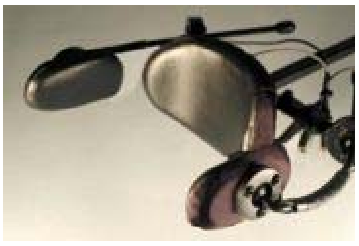


Figure 2 Head array controller, from Adaptive Switch Lab. Inc

As an alternative to these systems, there is the finger wheelchair drive control and the touch pad wheelchair drive control. The first one consists of a small square box about 3"x3" x 1 1/2" with a 2" hole on top. The finger control box can be mounted just about anywhere the user can comfortably reach. To drive the chair with a finger control box, the user places one finger through the hole on the top of the box and moves the finger in the direction they want the power wheelchair to move. This system is basically the same principle as a joystick in that it is a proportional drive but instead of moving a gimble, the user moves a finger (Figure 3).

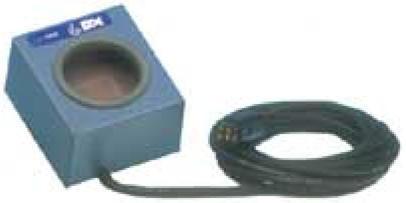


Figure 3 Finger wheelchair control

The touch pad also drives the power wheelchairs with a finger and can be mounted on several places of the wheelchair depending on the ability of the user to access it. Because touch pads are also proportional analogue drives, the user can determine and control the speed of the wheelchair while moving simply by a small movement of the finger.
Maskeliunas et al.[6] developed a HCI that tries to combine a traditional input with speech and video recognition technologies into one multimodal control "package". The authors think that the creation of a multimodal control interface combining various input and output modalities is a reasonable choice to fit the targeted audience with limited capabilities.
Carlson et al. [7], proposed a method that integrates a brain computer interface (BCI) with a vision system that interprets the high-level BCI commands given the experimental environmental context. The proposed method gives the user the ability to effectively drive the wheelchair without any collisions around an office.
More exotic input methods that have been implemented include detection of the wheelchair user's sight path (i.e., where the user is looking) through electrooculographic (EOG) activity [8] or the use of machine vision to calculate the position and orientation of the wheelchair user's head.
Reis et.al. [9] [10] developed a platform for intelligent wheelchairs called IntellWheels composed of a control software, simulator/supervisor and a real prototype of the intelligent wheelchair . The simple multimodal human-robot interface developed allows the connection of several input modules, enabling the wheelchair control through flexible input sequences of distinct types of inputs (voice, facial expressions, head movements, keyboard and, joystick). The system is capable of storing of storing user defined associations, of input's sequences and corresponding output commands.
In the proposed approach, hand segmentation is carried out through the use of a kinect[3] depth image in order to extract the hand blob and detect fingertips position to control the wheelchair. One advantage of the proposed solution compared to similar ones (finger drive), is the ability to control the wheelchair without touching any physical devices.

III. APPROACH DESCRIPTION

The proposed approach uses a kinect camera system mounted on the back of a wheelchair. The kinect is used to gather depth information for hand segmentation and finger detection. The

[1] http://atwiki.assistivetech.net/index.php/Alternative_wheelchair_control

user hand is segmented with the help of two planes that define the minimum and maximum thresholds of the extracted depth array. After hand segmentation the fingertips are extracted using the k-curvature algorithm [11]. The index finger is used to control the forward movement and turning left or right (Figure 4 - a, b, c) and the thumb control the lateral displacement to the left or right (Figure 4 - e, f). Movement to the rear is controlled with two fingers in 'V' (Figure 4 - d). Closed hand is the stop command.

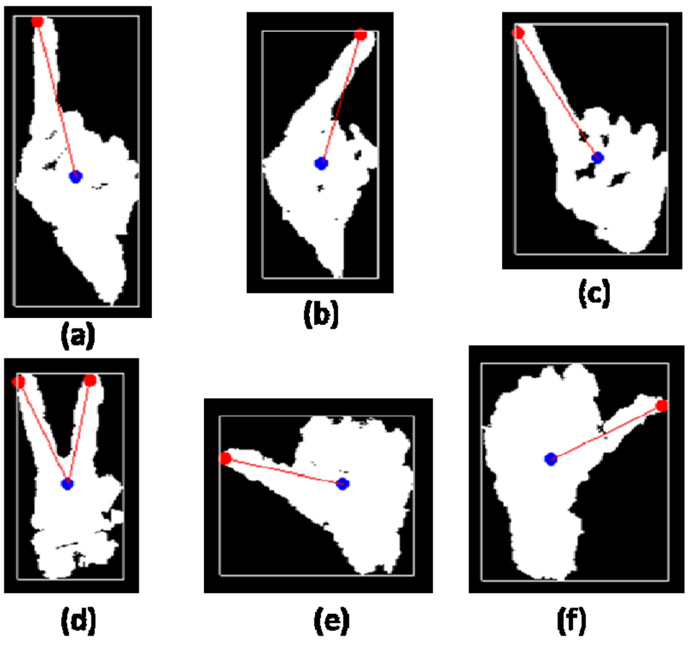


Figure 4. Finger commands used to control the wheelchair. (a) move forward, (b) turn right, (c) turn left, (d) move backwards, (e) move left, (f) move right

A. *Hand Segmentation*

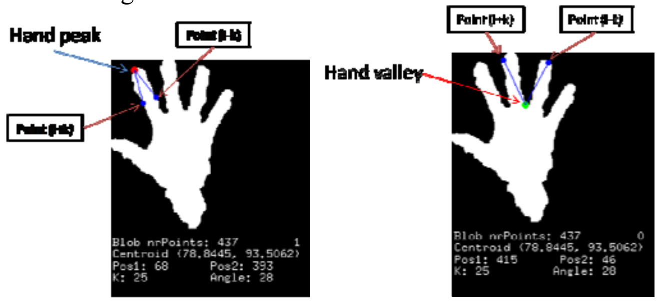
In order to segment the hand region, the nearest point to the camera is calculated on each frame. For each time t, the closest point on the depth image I is calculated according to the formula:

$$distMin = \min \left\{ \begin{matrix} I(x,y) if 0 \le x \le height(I) \\ and 0 \le y \le width(I) \end{matrix} \right\} \quad (1)$$

Using this value, two parallel planes [distMin-15, distMin+15] are defined to extract the hand blob from which the contour is calculated. The hand contour is then used to detect fingertips using the k-curvature algorithm.

B. *K-Curvature*

The k-curvature is an algorithm that attempts to find pixels that represent peaks along the contour perimeters [11] as shown in Figure 5.



```
Blob nrPoints: 437          1
Centroid (78.8445, 93.5062)
Pos1: 68        Pos2: 393
K: 25           Angle: 28
```

```
Blob nrPoints: 437          0
Centroid (78.8445, 93.5062)
Pos1: 415       Pos2: 46
K: 25           Angle: 28
```

Figure 5 Hand peak and valley point detection

At each pixel i in an hand contour C, the k-curvature is calculated and consists on the angle between the vectors A=[C(i), C(i-k)] and B=[C(i), C(i+k)], where k is a constant set equal to 30 in our implementation. The angle can be easily calculated using the dot product between the two vectors as illustrated in Figure 6.

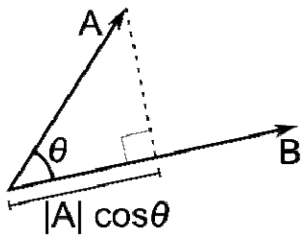


Figure 6 Dot product between vectors A and B

A value of θ=35º is used, such that only points below this angle will be considered further.

In order to classify the points as peaks or valleys, the cross product between the vectors is calculated (Figure 7). If the sign of the z component is positive, the point is labeled as a peak and stored; otherwise the point is a valley and is discarded.

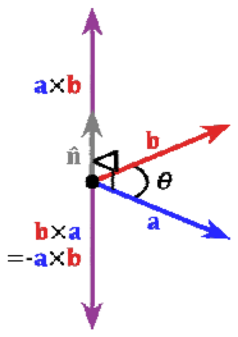


Figure 7 Cross product calculation

Finally, an average of all peak points detected on each finger is calculated, since it was found that a set of peaks were detected in the neighborhood of the strongest locations.

The following is the algorithm used to calculate fingertip locations:

Algorithm 1.

```
// blob = hand blob binary image
fingerK = 30;
for(int i=0; I < (blob.nPts) - fingerK; i++) {
    //calculating angle between k-vectors
    //first the case where we are in the first k points....
    if(I < fingerK)
        v1.set(blob.pts[i].x-blob.pts[blob.nPts+i -fingerK].x,
               blob.pts[i].y - blob.pts[blob.nPts+I - fingerK].y);
    else
        v1.set(blob.pts[i].x - blob.pts[I - fingerK].x,
               blob.pts[i].y - blob.pts[I - fingerK].y);

    v2.set(blob.pts[i].x - blob.pts[i+fingerK].x,
           blob.pts[i].y - blob.pts[i+fingerK].y);

// 3D vectors lying in the XY-plane for cross product calculation
    if(i<fingerK)
        v1_3D.set(blob.pts[i].x-blob.pts[blob.nPts+i-fingerK].x,
                  blob.pts[i].y-lob.pts[blob.nPts+i-fingerK].y,0);
    else
        v1_3D.set(blob.pts[i].x-blob.pts[i-fingerK].x,
                  blob.pts[i].y-blob.pts[i-ingerK].y,0);
    v2_3D.set(blob.pts[i].x - blob.pts[i+fingerK].x,
              blob.pts[i].y - blob.pts[i+fingerK].y,0);
    vXv = v1_3D.cross(v2_3D);
    v1.normalize();
    v2.normalize();

    theta=v1.angle(v2);
    // if theta < 35 then we are at a peak or valey ( /\ or \/ )
    if(fabs(theta) < 35) {
        if(vXv.z> 0)
            fingerFound = true;
    }
}
```

C. *Direction of movement and turning*

The wheelchair moving direction is calculated by the dot product between the control vector, vector between the hand centroid and the fingertip, and a horizontal vector parallel to a line that crosses the image centre as illustrated in Figure 8.

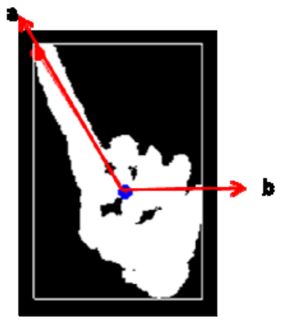


Figure 8 Vectors used in the calculation of finger orientation

The angle θ is calculated by using the dot product between the two vectors according to equation 2.

$$\theta = \arccos\left(\frac{a \bullet b}{\|a\|\|b\|}\right) \qquad (2)$$

IV. EXPERIMENTS AND RESULTS

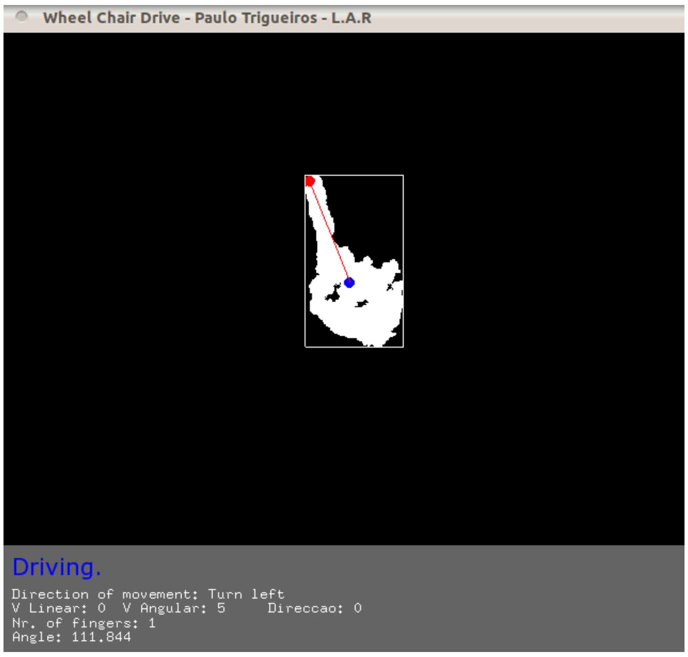
In order to validate the method, a series of experiments were made with an MSL robot from the Minho Team (from University of Minho). The finger commands calculated according to the above-described method are transmitted to the robot computer, allowing the user to control the wheelchair.

Several scenarios have been tried, with and without obstacles to test the easiness of wheelchair driving. The obtained results were very good for the problem in hands, showing that the solution is able to give people with disabilities an easy and inexpensive way to control a wheelchair.

The HCI (Figure 9) was developed using the C++ language, and the Openframeworks toolkit with two addons: the OpenCV[12] addon (ofxOpenCv) and the kinect addon (ofxKinect) under Ubuntu. OpenCV is used for some of the vision algorithms, and the fingertip extraction algorithm used, has been implemented by the same authors as an addon for the openframeworks. ofxKinect is used to control the kinect camera and to extract depth information. The computer used was a conventional notebook, with a 2GHz Core 2 Duo Processor.

The vision system operates at approximately 30 fps, and is able to correctly detect fingertips, which gives the possibility to have stable finger commands.

It takes 4 ms to calculate the near point and extract hand information, and takes about 2 ms to calculate fingertips for command classification.



```
Driving.
Direction of movement: Turn left
V Linear: 0  V Angular: 5     Direccao: 0
Nr. of fingers: 1
Angle: 111.844
```

Figure 9 User interface showing segmented hand and vector information

V. DISCUSSION

The proposed method consists of a new way to control a robotic wheelchair with the use of fingertip information, based on a Kinect sensor system to facilitate the extraction of useful hand features. One major advantage of this method consists on its simplicity, which leads to rapid learning rates, and gives the user the needed independent mobility. Also, the use of inexpensive hardware and open source tools make it a solution that can easily be applied to many other applications where human-computer interface can improve the quality of human life. The solution is not intended to be the best solution, but an alternative to the many solutions that exist in the market at the moment.

VI. CONCLUSIONS

This paper presents a method for controlling a robotic wheelchair using a kinect sensor system. One of the main advantages of the method is its simplicity in the number of commands the user has to use to drive the wheelchair. The user has just a number of finger commands. Another advantage is the ability to control the robot without wearing or touching any kind of device, using only finger movements and a kinect sensor which is mounted on the back of the wheelchair. The hardware used is very accessible, making this kind of solution very attractive. The solution is not universal – every disabled person is different and has different needs.

More tests, with the wheelchair developed in the laboratory of automation and robotics from the University of Minho, will be carried out to verify the robustness of the system and possible improvements.

The availability of open source tools to develop applications for this type of hardware is a very important aspect to take into account. These open source tools are quickly becoming widely used.

The type of solution used is easily adaptable to robotic equipment in general and is very useful in most kinds of situations. The software achieves very good performances, which gives the possibility to drive the wheelchair in a very natural way. The use of depth information is sufficient to extract hand features, and allows the system to operate in environments with varying light intensity.

There is however a drawback with this type of camera that has to do with the minimum distance required to operate it (approximately 0.5 m), which requires a special adapter to install the camera in the wheelchair.

ACKNOWLEDGMENT

The authors wish to thank all members of the Laboratório de Automação e Robótica, at University of Minho, Guimarães. Also special thanks to the ALGORITMI Research Centre for the opportunity to develop this research work.

REFERENCES

[1] Lab, U.L.R., Vision-Based Interaction for Robot Arms and Wheelchairs in Human Environments, UMass Lowell Robotics Lab: Lowell, MA 01854.

[2] Simpson, R.C., Smart wheelchairs: A literature review. Journal of Rehabilitation Research & Development, 2005. 42(4): p. 423-346.

[3] Wikipedia. Kinect. [cited 2011; Available from: http://en.wikipedia.org/wiki/Kinect.

[4] Lange, M.L. Driving with Your Head: head controlled power wheelchair access methods. 2001 May 2001; Available from: http://quickplace.medgroup.com/QuickPlace/nrn/PageLibrary86256D21004D93AC.nsf/h_05C223C63E65046486256F1D00569435/F555413977D54DB386256F17005AFA5F/?OpenDocument.

[5] Mahmood Ashraf, M.C., Towards Natural Interaction Using Alternate Wheelchair Controllers. International Journal on New Computer Architectures and Their Applications, 2011. 1(3): p. 1000-1013.

[6] R. Maskeliunas, R.S., Multimodal Wheelchair Control for the Paralyzed People, in Electronics and Electrical Engineering2011. p. 81-84.

[7] Tom Carlson, G.M., José del R. Millán, Vision-Based Shared Control for a BCI Wheelchair. International Journal of Bioelectromagnetism, 2011. 13(1): p. 20-21.

[8] Yanco, H.A., ed. Wheelesley, a Robotic Wheelchair System: Indoor Navigation and User Interface. Lecture notes in Artificial Intelligence: Assistive Technology and Artificial Intelligence, ed. M. VO, et al.1998, Springer-Verlag. 12.

[9] Reis, L.P., et al., IntellWheels MMI: A Flexible Interface for an Intelligent Wheelchair, in RoboCup 2009: 13th annual RoboCup Int. Symposium, Springer, Editor 2009: Graz, Austria. p. 296-307.

[10] Braga, R.A.M., et al., IntellWheels: A Modular Development Platform for Intelligent Wheelchairs. JRRD - Journal of Rehabilitation Research and Development, 2011. 48(9): p. 1061-1076.

[11] Malik, S., Real-time Hand tracking and Finger Tracking for Interaction, 2003.

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

CoopDynSim: a 3D robotics simulator

Toni Machado, Miguel Sousa, Sérgio Monteiro and Estela Bicho
Department of Industrial Electronics, University of Minho, 4800-058 Guimarães, Portugal
Email: {tmachado, msousa, sergio, estela.bicho}@dei.uminho.pt

***Abstract*—This paper presents *CoopDynSim*, a multi-robot 3D simulator. The main motivations for the development of a new simulation software lie in the need to emulate specific, custom made sensors, combined with the desire to smoothly transfer controller code from simulation to real implementation. The latter is achieved through the use of the same middleware layer already implemented in the real platforms. The high modularity of the solution allows the user to easily add new components or design new platforms. By having independent simulation threads for each robot, distributed control algorithms can easily be tested, abetted by a socket based connection, granting the possibility for an asynchronous, over the network, controller architecture. The ability to run simulations in *real* or *simulated* time, as well as a *play back* option, represent valuable features of the software. The simulator has been used in several projects, with different platforms and distinct control applications, proving it as a heterogeneous and flexible solution. Furthermore, its usage as a teaching tool in a robotics' summer school as well as in an introductory robotics class in our university, upholds its simplicity and user-friendliness.**

I. Introduction

Computer based robotic simulators have recently gained the attention of researchers [1], [2], [3], [4]. The availability of computers with an ever increasing processing power, combined with accurate physics engines and enhanced visual representations, of both robots and virtual worlds, made simulators go from being component or platform specific, often proprietary with restricted access, to a multi-platform and reconfigurable tool, widely used (especially) in academia.

Simulations provide a mean for one to collect data without the dangers of damaging expensive equipment, otherwise encountered when using the real platforms. For instance, the process of testing a control algorithm or the validation of a new sensor or platform are eased by means of simulation, while keeping the costs low, not only in terms of time spent, but also in terms of human resources needed. Furthermore, it becomes possible to perform tests under specific conditions, which may prove difficult to mimic, or expensive, in the real world, abetted by the availability of multiple (simulated) robots, even if only a few real platforms exist.

With a wide array of simulators accessible nowadays, certain features may help differentiate the available solutions from one another [5], [6], [7]. *Graphical* and *physical accuracy*, as the extent to which the robots and virtual world are similar to their real counterparts, *flexibility*, that is, the type of hardware that can be simulated, and *transparency*, implying the possibility for the user to seamlessly migrate from simulation to the real platforms, represent the key characteristics, from our perspective. Furthermore, the cost of the solution and openness of the source code may be important. Moreover, we argue that the simulator should be simple and user friendly, both in terms of the installation process and normal usage. Our work presents a solution, built from the ground up, which meets the above requirements.

CoopDynSim (*Coop*erative *Dyn*amics *Sim*ulator) is built on top of the *Newton Game Dynamics* [8] physics engine, recurring to *OpenGL* [9] to render the environment. Albeit initially designed for the hardware platforms developed in-house, which feature custom made components (difficult to add to other available simulators), it still offers the possibility to add other platforms, designed in third party software, as well as user-defined worlds.

The main strengths of the proposed simulator architecture lie in the modularity and level of abstraction of the robotic components, through the use of a middleware layer. Furthermore, the ability to run in *real* or *simulated* time, as well as a *play back* feature, which allows the user to replay a simulation, represent key characteristics of the solution.

The simulator is being used in several research projects [10], [11] and as a teaching tool for robotics courses in our university, which further help in its validation. *CoopDynSim* is in constant development, and it currently runs on Windows.

The remainder of the paper is organized as follows. Section II presents a brief description of some related work. Section III describes the overall architecture of the simulator, its components and its features. In Section IV, a few use cases are presented. Finally, Sections V and VI conclude the paper and present some guidelines for future work, respectively.

II. Related work

In this section we briefly describe some of the available simulators. We only aim to give an overall review of a few products, both free and commercially available. For more in-depth surveys on the subject, please refer to [5], [6], [7].

A. Freeware

One of the most notorious open-source solutions is the *Player-Stage-Gazebo* project [12]. Player represents a hardware network server, and Stage and Gazebo are the 2D and 3D simulators, respectively. Player is a TCP socket based middleware layer, which guarantees abstraction of the robotic hardware modules. Stage is the two-dimensional simulator with low physical accuracy, providing only basic collision detection and simple models. Nonetheless, it excels in the simulation of large groups of robots, such as swarms. Gazebo, on the other hand, is a three-dimensional simulator devised for simulating a smaller number of platforms. It can make use of the ODE physics engine [13], and multiple sensors and commercial robots are available. The simulator runs only on UNIX based machines and has a challenging installation process, which can represent shortcomings of the solution.

USARSim (Unified System for Automation and Robot Simulation) [14], is a simulation tool based on the Unreal engine. Although the simulator itself is free, the engine has a cost associated with it. USARSim is the official simulator used in the RoboCup's Rescue simulation league. Because of its incorporation with the Unreal engine, a high degree of detail and realist world interactions are provided (Karma is the built-in physics engine). Robot's programming and control can be achieved using UnrealScript, but also through other network based frameworks (integration with Player, SIMware and Pyro is possible). USARSim is cross-platform, but its installation process, here also, can be overwhelming, since one has to install the engine and several external packages, as well as become familiar with a large amount of documentation [7].

Simbad [15] is an open source simulator written in Java and only requires the Java 3D visualization environment to run, thus making it highly cross-platform. It features only basic physics simulation, being mainly designed for researchers in evolutionary robotics and artificial intelligence, since it includes dedicated libraries for artificial neural networks and genetic algorithms.

SimRobot [16], built on top of the ODE physics engine, is mainly used in RoboCup's Standard Platform League. It features a user friendly, drag and drop interface to build the simulated world, as well as the possibility for the user to easily create its own platform, with generic bodies and sensors. One characteristic that distinguishes this solution from others, resides in the built-in code with the simulator's executable, rather than a client/server approach, which the authors argue that allows the user to easily pause and continue the simulation, easing the debugging process.

B. Commercially available

Within commercial robotics simulators, *Microsoft Robotics Developer Studio* [17] is a popular solution. It is based on the high fidelity physics engine PhysX , and features high quality visualization, with a large collection of robots available. The main programming language used is C#, along with the Visual Programming Language (VPL) developed by Microsoft, which allows users to easily create a control application, without the need of being familiar with programming.

Webots [18], developed by Cybertronics, is a multi-platform simulation software that features a large number of commercially available platforms, as well as the possibility for the user to create its own, using any of the existing sensors and actuators. Relying on the ODE physics engine, it is capable of simulating wheeled, legged and flying robots. Programming can be done through C, C++ and Java (TCP connection for external interface is also featured), and applications can be cross compiled for the real hardware platforms.

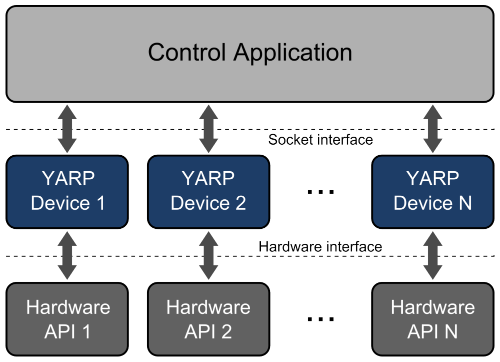


Fig. 1. Schematic of the Hardware Abstraction Layer used in all of our robots.

robotSim [19] is another simulation software solution, from Cogmation Robotics, which offers a very realistic environment, along with customizable physics for each robots, which allow the user to tweak how it interacts with the environment, in order to get more realistic simulations. Multiple robots can be simulated, and their (individual) control may be achieved through the provided C++ API or through the network. *robotBuilder* is a concomitant tool from Cogmation Robotics, allowing the user to create a custom platform, with any of the available sensors.

III. Architecture

CoopDynSim is a 3D robotics simulator, developed in C++, capable of emulating multiple robots or teams of robots, obstacles and targets. Newton Game Dynamics [8] is the chosen physics engine, with Open Graphics Library [9] being used to render the scene.

The simulator was developed taking into account the robots existing in our laboratory, i.e. the simulated robots have the same characteristics and interface as the real ones and follow the client-server topology, where each robot is composed of several hardware modules that act as servers, and the control application has clients that connect to each of these modules. This modularized approach makes the addition or removal of hardware (i.e. sensors, actuators) an easy task.

The middleware in use is based on YARP [20] and provides a wrapper with a socket based interface for each of the hardware modules, as illustrated in Fig. 1. Since this abstraction layer is employed both in simulation and in real implementation, the same control application can be used, thus eliminating the, most of the times, hard and time consuming process of migrating from simulation to the real platforms. Nevertheless, a few control parameters may need to be adjusted, due to the fact that the real platforms have unknown perturbations that can not be accounted for in simulation.

To interface with each of the modules, a generic protocol was developed, which is implemented in all of the hardware modules in our laboratory (from robotic manipulators, to vision systems and motor drivers, etc). The message format can be seen in Fig. 2.

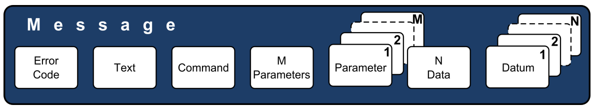


Fig. 2. Communication protocol.

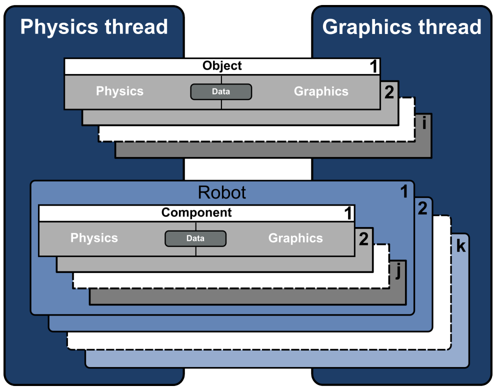


Fig. 3. Thread implementation of the virtual environment.

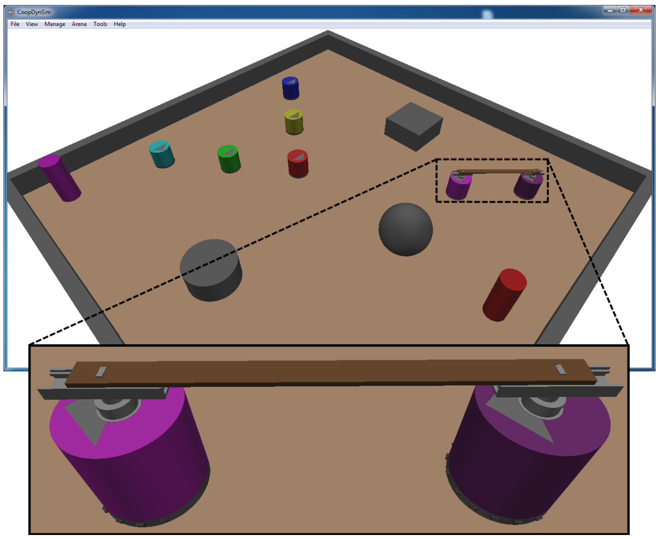


Fig. 4. Virtual environment with a team of 2 robots transporting a bar and a team of 5 robots in a formation.

The message's fields are as follows: *Error Code* is an integer that reports the success of the message; *Text* is a set of characters; *Command* is an integer that contains the command's id; *M Parameters* represents the number of parameters that the current message has, with *Parameter 1, 2, ..., M* accounting for the parameters sent; in the same way, *N Data* and *Datum 1, 2, ..., N* are the number of floating point values, and the values themselves, respectively. *M Parameters* and *N Data* are used to unpack the message on its destination. By following this message protocol, the user can control the robots using any language that supports socket based connections.

A. Design

Concerning the main implementation scheme, *CoopDynSim* runs three main threads, one dedicated to the physics update, another dedicated to render the scene and the final one responsible for the interface window. Each of these threads runs independently from the others, with different update rates. For instance, the visualization thread does not require a high rate (we use 10 fps by default), whereas the physics one needs a greater update rate (around 100 fps in our case). Furthermore, each robot inserted in the simulator spawns a dedicated thread. Responsible for updating the values of the actuators (with the last command received) and sensors (with the last update from the physics), this thread can run independently from the physics thread, implying *real time* simulation, that is, the time elapsed in simulation is the same as the real time, and the simulation is independent from the control application(s) connecting to the robot(s). Conversely, if one wishes to speed up a simulation or increase the number of platforms in use, a *simulated time* option is available, where each iteration step from the part of the physics will only occur after each robot has received its control command (thus implying a synchronization mechanism), and the time to be simulated by the engine is a fixed value, independent from the elapsed time, which can be defined by the user (it can not be lower than 10 milliseconds, in order to guarantee physics stability).

As for the objects in the virtual world, each is composed of physical and graphical properties, Fig. 3. Physical properties are represented by the shape (simple shapes or composed ones), the mass, the mass distribution, friction coefficients, etc. All these properties are predefined, unless if the object is inserted using a configuration file (see Fig. 7), in which the user can specify its mass. Graphical properties are defined by the 3D shape to represent, i.e. the object's vertices and colors information. When an object is inserted in the virtual environment, the physics' thread "acts on it" and the graphics' thread updates its location. Hence, each object has a shared block of memory, accessed by both, that contains information about its location. The robotic platforms are nothing more than a set of attached objects, arranged in the best possible fashion, in order to accurately emulate the real robots.

The virtual robots have the same characteristics as the real ones, i.e. dimensions, sensors and actuators. Actually, two main types of robots were implemented at first. Composed of a cylindrical chassis, eleven distance sensors, two differential motorized wheels, two caster wheels, one type has, in addition, a dedicated support needed for cooperative transportation tasks. To emulate the distance sensors, a ray trace algorithm, provided by the physics engine, is used in each of the sectors. As for the locomotion, the motorized wheels are emulated by two cylinders attached to the main chassis by a hinge joint, with the caster wheels needed to balance the platform. The vision system is not being replicated, with the module simply returning the target (colored marker or another robot) information. Fig. 4 shows a team of two robots transporting a bar and its target (magenta cylinder), a team of five robots in a inverted V shape formation and its target (red cylinder), and 3 distinct obstacles (box, cylinder and sphere).

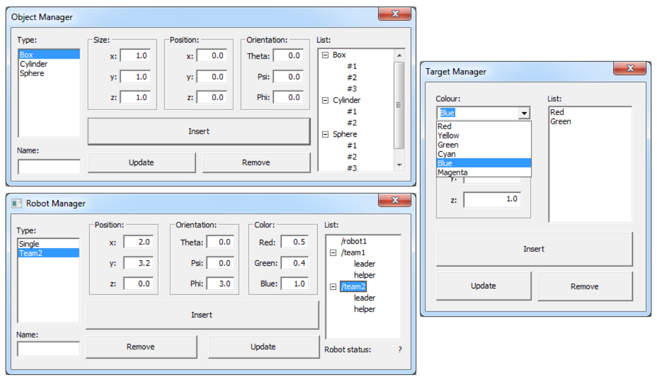


Fig. 5. Objects', Robots' and Targets' Managers.

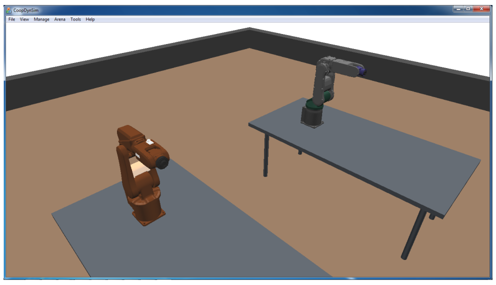


Fig. 6. Virtual environment with two robotics arms: *ABB^{TM}IRB 120* (left side) and *Motoman^{TM}MH5* (right side) [21].

```
[FLOOR]
size        lenght width height
[WALL_1]
type        box_cylinder_or_sphere
size        lenght_wall_1 width_wall_1 height_wall_1
position    x_wall_1 y_wall_1 z_wall_1
orientation phi_wall_1 psi_wall_1 theta_wall_1
mass        m_wall_1
...
[WALL_n]
type        box_cylinder_or_sphere
size        lenght_wall_n width_wall_n height_wall_n
position    x_wall_n y_wall_n z_wall_n
orientation phi_wall_n psi_wall_n theta_wall_n
mass        m_wall_n
```

Fig. 7. World file template.

B. *User interaction*

After describing the main components of the simulator and addressing its inner workings, the question of how the user can interact with the software arises. Starting with an empty world (with only the floor), navigation through the virtual environment is achieved via the mouse buttons: left button for rotation, right button for translation and mouse wheel for zooming in and out. To insert elements in the world, two levels of abstraction are available to the user.

1) Basic elements: Accessible through the *Manager* menu, these represent the three basic constituents of the virtual environment: *objects*, *robots* and *targets*.

a) Objects: With the *Object Manager* (Fig. 5 top left) the user can easily insert simple objects, such as a *Box*, a *Cylinder* and/or a *Sphere*, specifying the size, position and orientation within the virtual world. These can also be updated, at any time, or removed, through the same menu. Only three basic objects are available, nevertheless, the implementation of more complex ones is a trivial task, given the way the code is organized.

b) Robots: Single or teams of robots (the latter used for transportation tasks) can be inserted through the *Robot Manager* menu (Fig. 5 bottom left). In the same way, position and orientation can be modified, with the addition of a customizable color (useful to differentiate the robots from one another, in multi-robot scenarios) and name. This name identifies the virtual platform on the network, and each needs to have a different one. For instance, "/robot1/motors" and "/robot2/motors" represent the tags, on the YARP network, for the motors' module of robots 1 and 2, respectively.

Not only mobile platforms can be emulated though, with three robotic arms being added to the simulator at the time of writing (concretely, *amtec^{TM}lwa 7dof*, *ABB^{TM}IRB 120* and *Motoman^{TM}MH5*), which helps to prove the flexibility of the solution, when it comes to the type of platforms it can simulate (Fig. 6).

c) Targets: Representing a special type of objects, targets (*Target Manager*, Fig. 5, right) are colored landmarks which specify desired destinations for the robots to reach. Each mobile platform has a *target* module that returns the distance and angular displacement to these markers, thus (roughly) simulating the vision system (we use colored boxes, with the real platforms, to represent the desired location to reach in the real world).

2) Compound elements: In order to decrease the effort necessary to setup an experiment, more complex elements are available, concretely, *worlds* and *scenarios*.

a) Worlds: Composed of a floor and *N* objects, these *arenas* help the user to setup a custom environment, and easily load it into the simulator. Several default ones are already available via the *Arena* menu. In order to create a new arena, a plane text file with the *.world* extension is used. Fig. 7 shows the structure of such file, where the user can specify the *FLOOR* dimensions (*length*, *width* and *height*) and each of the object's properties (*type*, *size*, *position*, *orientation* and *mass*). These user defined arenas can be loaded using the *Load from file...* item in the *Arena* menu.

b) Scenarios: In order to quickly setup an experiment, in addition to a custom world, a complete scenario can also be loaded by the user (plane text file with *.scenario* extension file). The same file based approach is used here (its structure can be seen in Fig. 8), where the world, targets and robots (type and properties) can also be defined. This option can be accessed through the *Load Scenario...* item in the *File* menu.

C. *Play back mode*

A useful feature implemented in the simulator concerns the play back option. If the user chooses to, in each iteration step, the software will save the robots' positions within the virtual world to a specific file, as well as the scenario used.

```
[WORLD]
file        path_to_the_world_file
[TARGET]
color_1     x_pos_tar_1 y_pos_tar_1 z_pos_tar_1
...
color_n     x_pos_tar_n y_pos_tar_n z_pos_tar_n
[ROBOT_1]
type        single_or_team2
name        network_name_of_the_robot_1
position    x_robot_1 y_robot_1 z_robot_1
orientation phi_robot_1 psi_robot_1 theta_robot_1
color       R_1 G_1 B_1
...
[ROBOT_N]
type        single_or_team2
name        network_name_of_the_robot_n
position    x_robot_n y_robot_n z_robot_n
orientation phi_robot_n psi_robot_n theta_robot_n
color       R_n G_n B_n
```

Fig. 8. Scenario file template.

Afterwards, to recap the simulation, a load menu is prompted to the user, and the positions from the play back file are stored in memory. In order to go to a specific time in the past simulation, a global module based on YARP is used, which receives said desired time via the network (much like the modules of the robotic platforms).

Analyzing the dynamics of the robots (along with log files from the control application), whilst having a visual feedback of the positions of the robots in the world, makes the process of debugging a control approach a much easier task, especially in a multi-robot scenario.

IV. Use cases

CoopDynSim started being developed to be used with object transportation tasks by multi-robot teams [22]. Fig. 9 illustrates a simulation with two teams of two robots transporting a long bar from an arena's corner to the opposite one. Here, each team features a *leader*, whose main responsibility is reaching the final destination, and a *helper*, who helps the leader carry the load. By making use of the simulated support (custom made sensor) and replicating a complex joint transportation task, this scenario is particularly important when it comes to endorse the flexibility of the simulator.

The software was also used in a multi-robot formation research [11], in which teams of autonomous mobile robots navigate in a desired configuration, whilst avoiding obstacles that may lie in the robots' path, by breaking formation, returning to their position after such obstacles are surpassed (Fig. 10 depicts such scenario).

A different project in development, using the software, is aimed at using robotic arms to aid in brain surgery. By adding different arms to the simulator, the study of which is more appropriate to the task in hand becomes easier, as well as the testing of different control algorithms, for such a delicate application.

Concerning its applicability as a teaching tool, it was used for the first time in the *Hands-on Summer School: Neural Dynamics Approaches to Cognitive Robotics 2011* [10]. The participants quickly became familiar with the software, and a few transferred the code that themselves wrote (in the control application) for the simulation, to the real platforms, with only a few parameters' adjustment needed to obtain the same results as in simulation. This simplicity and complete transparency of the whole approach made the simulator receive a great feedback from the participants.

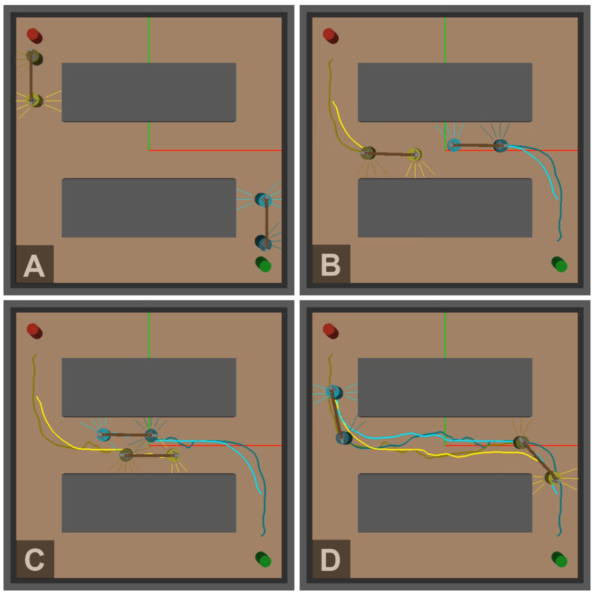


Fig. 9. Snapshots taken from a simulation of a team of two robots transporting long loads [23].

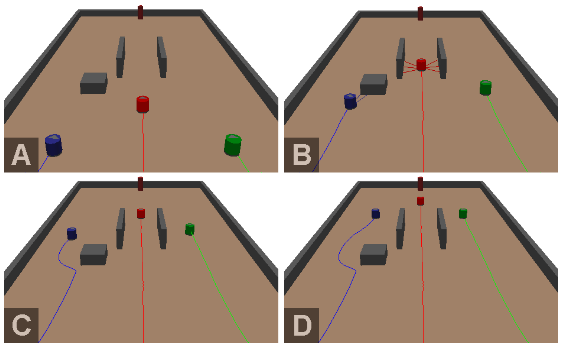


Fig. 10. Snapshots taken from a simulation of a team of three robots navigating in a triangle formation [11].

After such successful use of *CoopDynSim*, it was adopted as a teaching tool for the *Automation, Control and Robotics* in the Industrial Electronics Engineering Master Degree, at the University of Minho in Portugal. Here, the students are given a single install package for the simulator and a MATLAB application to implement the control. The software helps the students to better understand the theoretical concepts taught in the course.

V. Conclusions

We have presented *CoopDynSim*, a 3D robotics simulator, based on Newton Game Dynamics, Open Graphics Library and YARP.

By having complete transparency, when it comes to the controller code, as one of the design criteria, the simulator greatly diminishes both the effort and the time spent migrating from simulation to the real implementation. With this dedicated solution, but easily expandable to include other robot models, when the control application is transferred to our real platforms just fine tunning control parameters may be needed (real robots are accurately replicated in the simulator, but obviously some unexpected real perturbations can not be taken into account). If robot models provided by the most other simulators were used, the switch to real platforms would exhibit an exaggerated effort which increases with the number of robots (multi-robot control) due to the sensory information and motor actuation are distinct from the one of the real platforms.

Furthermore, the possibility to easily add custom made sensors and platforms (albeit having to directly modify the source code), succors the flexibility of the solution, widening the possible applications in various research projects.

The *simulated time* and *real time* options further increase the software's flexibility, since the former can be used for a simulation with a high number of platforms (for instance, swarms) and the latter is more suitable to use in scenarios with only a few robots. Also, the *play back* feature gives the user a possibility to recall an entire simulation, which represents an useful feature of the software.

The simulator's user friendliness and simplicity were validated by its usage as a teaching tool in our University.

CoopDynSim is free and can be downloaded from our MARL web server [1].

VI. Future work

CoopDynSim is still in an early development stage. The main requirements for the project have been fulfilled, nevertheless, many aspects can be improved and some features can be added.

In order to make the simulator more visually appealing, some textures could be added to the objects on the world. Furthermore, the possibility to easily add new objects and/or robots without making changes to the source code, as well as a, more intuitive, drag and drop user interface are key features found in many available simulators that are lacking in our solution.

The simulated vision system is another point that needs improvement. Instead of just directly returning the angle and distance to a target, a virtual image of the robot's field of vision (virtual camera) will give the possibility to use the same image processing application that is in use in the real robots.

Moreover, the software should be made platform independent (i.e. Linux, Mac OS, etc), liberating the user from having to use a specific system (currently it only runs on Windows OS).

Acknowledgments

We would like to thank all the people that work in our laboratory, MARL (Mobile and Anthropomorphic Robotics Laboratory) in University of Minho. All of them contributed in several ways for the success of this work. A special thanks to Carlos Faria, for the addition of the three robotic arms to the simulator. Toni Machado would also like to thank the Portuguese Science and Technology Foundation for providing his Ph.D. scholarship (ref. SFRH/BD/38885/2007).

[1] http://marl.dei.uminho.pt/Public/CoopDynSim.zip

References

[1] "Workshop on robot simulators: available software, scientific applications and future (along with IROS)," Nice, France, 2008.
[2] "Workshop on Space Robotics Simulation (along with IROS)," 2011.
[3] "International Conference on Simulation, Modeling and Programming," Darmstadt, Germany, 2010.
[4] "International Conference on Modelling and Simulation," Phuket Island, Thailand, 2011.
[5] J. Craighead, R. Murphy, J. Burke, and B. Goldiez, "A Survey of Commercial & Open Source Unmanned Vehicle Simulators," in *2007 IEEE International Conference on Robotics and Automation*, no. April, 2007, pp. 852–857.
[6] P. Castillo-Pizarro, T. V. Arredondo, and M. Torres-Torriti, "Introductory Survey to Open-Source Mobile Robot Simulation Software," in *2010 Latin American Robotics Symposium and Intelligent Robotics Meeting*. Ieee, Oct. 2010, pp. 150–155.
[7] A. Harris and J. M. Conrad, "Survey of Popular Robotics Simulators , Frameworks , and Toolkits," in *2011 Proceedings of IEEE Southastcon*, 2011, pp. 243–249.
[8] "Newton Game Dynamics." [Online]. Available: http://newtondynamics.com
[9] "Open Graphics Library." [Online]. Available: http://www.opengl.org
[10] "Summer School Neuronal Dynamics Approaches to Cognitive Robotics," 2011. [Online]. Available: http://cognitive-robotics-school.dei.uminho.pt
[11] M. Sousa, "Multi-robot cognitive formations," M.Sc. dissertation, University of Minho, 2011.
[12] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems," in *International Conference on Advanced Robotics (ICAR 2003)*, no. Icar, 2003, pp. 317–323.
[13] "Open Dynamics Engine." [Online]. Available: http://www.ode.org
[14] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, "USARSim: a robot simulator for research and education," in *2007 IEEE International Conference on Robotics and Automation*, no. April, 2007, pp. 1400–1405.
[15] L. Hugues and N. Bredeche, "Simbad: An Autonomous Robot Simulation Package for Education and Research," *Lecture Notes in Computer Science*, vol. 4095, pp. 831–842, 2006.
[16] T. Laue, K. Spiess, and R. Thomas, "SimRobot - A General Physical Robot Simulator and Its Application in RoboCup," *Lecture Notes in Computer Science*, vol. 4020, pp. 173–183, 2006.
[17] J. Jackson, "Microsoft Robotics Studio: A Technical Introduction," *IEEE Robotics & Automation Magazine*, no. December, pp. 82–87, 2007.
[18] O. Michel, "Webots: Professional Mobile Robot Simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, pp. 39–42, 2004.
[19] "Cogmation Robotics: robotSim documentation." [Online]. Available: http://www.cogmation.com/pdf/robotsim_doc.pdf
[20] P. Fitzpatrick, G. Metta, and L. Natale, "Towards long-lived robot genes," *Robotics and Autonomous Systems*, vol. 56, no. 1, pp. 29–45, 2008.
[21] C. Faria, "Robotic implantation of intracerebral electrodes for deep brain stimulation," M.Sc. dissertation under development, University of Minho, 2012.
[22] R. Soares, E. Bicho, T. Machado, and W. Erlhagen, "Object transportation by multiple mobile robots controlled by attractor dynamics: theory and implementation," in *Proceedings on the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. San Diego, CA, USA: IEEE, Oct. 2007, pp. 937–944.
[23] T. Machado, T. Malheiro, S. Monteiro, and E. Bicho, "A dynamical systems approach to flexible transportation by teams of two robots," *Manuscript under preparation*, 2012.

A bottom-following problem approach using an altimeter

José Melo, Aníbal Matos
INESC TEC (formerly INESC Porto) and Faculty of Engineering
University of Porto
Rua Dr. Roberto Frias, s/n
4200-465 Porto, Portugal
{jose.melo, anibal}@fe.up.pt

***Abstract*—Information on the profile of the bottom of the sea or river is of unquestionable relevance and importance. The applications for this are traditionally related with bathymetry, where the objective of mapping the bottom of the river or sea is achieved by using advanced ultrasound equipment. However, more recently, this information has found use in bottom-following applications, where the purpose is obtaining images or footage from the river or sea floor.**

This paper addresses the problem of bottom following by an Autonomous Underwater Vehicle in an environment which is not previously known. To accomplish this, the MARES AUV vehicle was equipped with an down-facing altimeter to continuously measure ranges to the bottom of the seabed.

The essential of this work consisted on fully integrating the altimeter with the navigation and control layers of the on-board software of the vehicle. First the altimeter settings were fine-tuned to minimize the number of spurious range measurements. After that, a Kalman Filter was implemented to smooth the noisy data and prevent possible wrong range measures to be propagated to the control loop. Finally, the whole experimental setup was validated, and the MARES AUV was able to execute several bottom-following missions in an open-water environment.

I. Introduction

With the development of Autonomous Underwater Vehicles (AUV), inspecting the bottom of seas, rivers, lakes, but also underwater infrastructures is becoming increasingly more interesting, easy and affordable. AUVs are now smaller, cheaper and incorporate large sets of sensors, making it extremely convenient to gather different kinds of information, in a very easy way. Underwater missions involving visual inspection of the seabed are now more and more common. These applications are for example bathymetry, monitoring of industrial structures, sonar or video imaging, and so on [1]. However, The bottom of rivers or seas, usually offers adverse and unpredictable conditions. The scarce light available and the turbidity of the water next to the seabed dramatically affect tasks related with the visual inspection. The ability to follow the bottom as closely as possible is of obvious relevance.

In perfectly known environments, navigating close to the bottom becomes a trivial navigation problem without much complexity. Nonetheless, in most cases it is impossible to know in advance the profile of the bottom. Having the capability to safely navigate close to the bottom without neglecting the safety of the vehicles and equipments involved can be extremely valuable. In this paper we address the problem of bottom following, using an altimeter, in shallow-water environments which are not previously known, by u

The work here presented consisted on fully integrating the altimeter with the navigation and control layers of the on-board software of the vehicle. Firstly the altimeter settings were fine-tuned to minimize the number of spurious range measurements. After that, a Kalman Filter was implemented to smooth the noisy data and prevent possible wrong range measures to be propagated. The output of the Kalman Filter will then be used to generate the proper references to the control loop, guaranteeing that the vehicle navigates at a safe distance to the bottom. Finally, the whole experimental setup was validated, with the MARES AUV being able to execute several bottom-following missions in an open-water environment. Alternative approaches to the bottom following problem can be found in [2], [3], [4].

The remaining of the article is organized in the following way: the experimental setup is described in section II, and after that, and in section III we introduce the Kalman Filter algorithm to be used. Section IV describes the between of the range measures with the navigation layer of the software some results, both simulated and and experimentally taken during operations in the Douro river are presented in section V. Finally, in the last session, some conclusions are presented as well as orientations for future work.

The remaining of the article is organized in the following way: the experimental setup is described in section II, and after that, and in section III we introduce the Kalman Filter algorithm to be used. Section IV describes the integration of the range measures in the navigation layer of the software and in the following section some simulated and also experimental results. The latter ones were experimentally taken during operations in the Douro, performed in August 2012. Finally, in the last session, some conclusions are presented as well as orientations for future work

II. Experimental Setup

Bottom-following was initially described as "maintaining a fixed altitude above an arbitrary surface whose characteristics may or may not be known" [5]. In this paper, we address the problem of bottom following with the MARES (Modular Autonomous Robot for Environment Sampling) AUV.

The MARES AUV, in Figure 1, is a highly modular torpedo-shaped 1.5 meters long AUV, designed to be able to carry a wide variety of payload sensors, in different vehicle configurations. Weighting 32kg, and propelled by 4 motors, the vehicle achieves a very high degree of manoeuvrability with an almost decoupled control of the horizontal and vertical motion of the vehicle [6]. However, the major defining characteristics of MARES is the ability to hover in the water column, and perform trajectories with arbitrary small horizontal and vertical speeds, making it a most adequate vehicle for the intended bottom-following operations. The MARES is equipped with a set of sensors for navigating in dead-reckoning mode, like a pressure sensor and an inertial measurement unit (IMU), that consists on three accelerometers, three gyroscopes and three magnetometers, all aligned with the x, y and z axis of the vehicle. Additionally, operations are also supported with an acoustic Long Baseline network. This network, that consists on a set of two acoustic beacons or buoys, makes it possible to overcome the drifting inherent to dead-reckoning techniques.

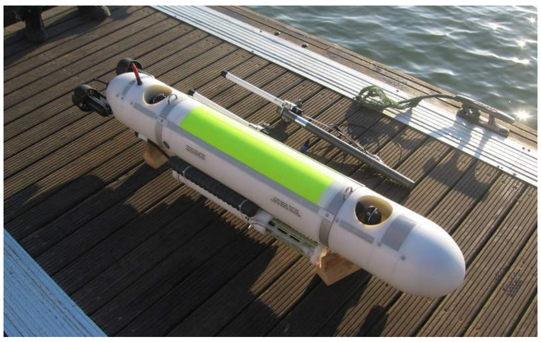


Fig. 1. AUV MARES

Nowadays it is widely known that the most reliable way of assessing the distance to the bottom in underwater environments is by using sonar techniques, mostly due to the unique characteristics of sound propagation in the water. To be able to assess the distance of the MARES to the bottom of the seabed, the vehicle was equipped with an altimeter in a downward-faced configuration, to continuously measure the vehicle distance to the bottom. The Imagenex Model 862, in Figure 2 was the sensor used. This is a completely self-contained altimeter that operates at frequencies of 330Khz and ranges of up to 50 meters, adequate enough to shallow-water environments, like rivers or near-shore sea. This altimeter has a 10^o conical beam, and the footprint on the bottom will in general be an ellipsoid. For slant terrains, the altimeter will always provide the range corresponding to the first reflection.

The altimeters rely on sonar techniques and, as such, on the propagation of acoustic waves, to measure ranges. In a rather simplistic way, what the altimeter does is to emit a sound wave at a given frequency, and during a given time period, and then wait to detect the reflection of that same wave. As the sound waves travel, they are attenuated by the medium where they travel, in this case the water. The same happens when a sound wave hits an object, with the reflected wave being attenuated when comparing to the original wave. By considering the velocity of the sound in water fairly constant, then the distance covered by the sound is proportional to the time of travel, and computing these distances is straightforward by timing both emission and reception instants.

Fig. 2. The Imagenex 864 altimeter

Sound waves propagate extraordinarily well in the water, even faster than in the air, with speeds of approximately 1500m/s. Although the speed of sound in water can be considered fairly constant, there are some environmental factors that make it vary from site to site. Environmental parameters like temperature, salinity or even depth are known factors that can directly affect the propagation and attenuation of sound; also, the attenuation that a sound wave is subject to when hitting an object and being reflected will vary significantly with the properties of the object. In the case of the bottom of the sea or river, if the seabed is sandy, the attenuation with the reflection will be much higher than if the seabed would be rocky.

Given that the altimeter in use can be configured with different parameters for range, gain and pulse length, it must be calibrated accordingly. These parameters are in fact of crucial importance, and failing to do so can negatively influence a mission, as exemplified in Figure 3. In a first approach, the altimeter was thoroughly tested in a small tank, in the lab. The tests allowed to understand the influence of the different parameters to be calibrated in the observed ranges. Despite that, and as expected, the optimal parameters that were empirically found in the tank, were rather different from the ones used in missions performed in the river. In the same way, it is likely that these parameters would need some tuning for missions performed under different environmental conditions, e.g. in the sea.

On both of the graphs displayed in Figure 3, an example of the profile of the bottom of the river can be seen, taken during a calibration mission in Douro river, close to Porto, in June 2011. The altimeter was attached to an autonomous surface vehicle - the ZARCO ASV - and set to continuously ping the bottom. The figure presents the range measurements from the altimeter throughout the time, while the ASV was performing random trajectories at the surface. On the left side of Figure 3, though it is possible to perceive the profile of the river, the set of data is extremely noisy, with a lot of false points detected in the band of 2m to 4m; moreover, due to the apparent randomness of the noisy points, this data sets

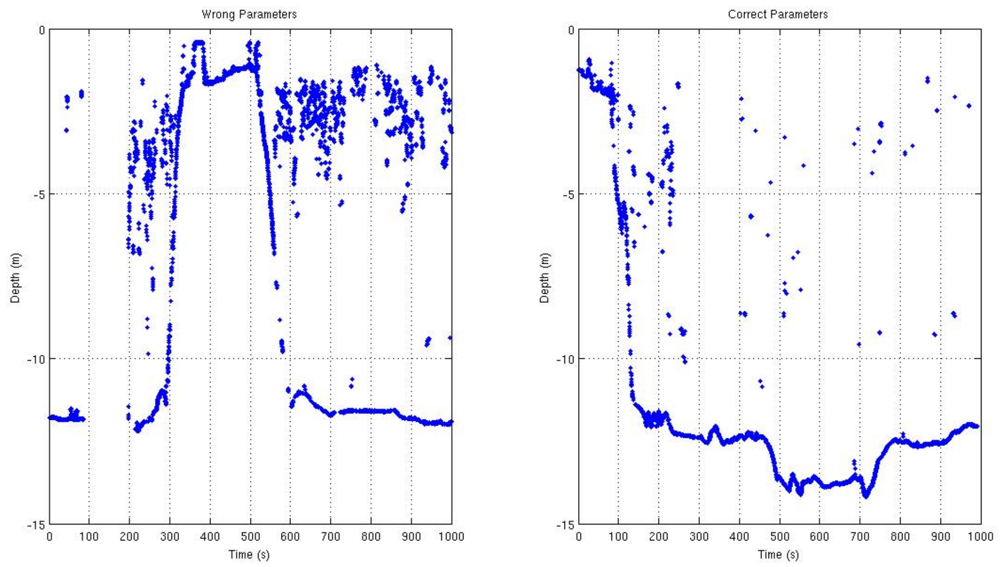


Fig. 3. Output of the altimeter: on the left, a situation when excessive gain and pulse length is depicted, with a lot of noisy measurements; on the right, a situation where the parameters have been correctly set

configures a situation where an efficient filtering of the noisy data would be extremely hard to achieve. This is a typical situation where the parameters of the altimeters were badly tuned: the gain was too high and the noise in the band of 2m to 4m is probably a cause of multiple reflections of the sound, both on the bottom of the seabed and at the surface. The image on the right side, on the other hand, shows the profile of the bottom of the river, taken at the same spot, but with the parameters correctly set. Even though it is possible to clearly see some outliers and some low-amplitude noise, there is an undoubtedly improvement to the previous situation.

III. Filtering

The output of the altimeter, when its configuration parameters are properly set, presents measures that are consistent throughout time. Despite that, and as expected, these measures still present some noise, most of the times in the same order of magnitude of the quantum of the sensor, which is 2cm. Moreover, this effect is more noticeable when the sensor is sending acoustic waves while moving horizontally, for example, when mounted on a vehicle which is moving with appreciable speeds.

The ranges measured by altimeter are supposed to generate proper depth references to be fed to the control of the vehicle and that necessarily need to present a relatively smooth behaviour. The need for filtering the raw altimeter measure naturally arises: on one hand outliers and spurious measurements need to be eliminated, and on the other hand, this stream of measures needs to be smoothed out. On top of that, it must be ensured that the delay introduced by the filtering process does not influence the control of the vehicle. Even though the vehicle dynamics are slow, delays higher the 0.5 seconds are already considered significant. An example of the raw output of the altimeter can be seen in Figure 4, where the presence of outliers is clear.

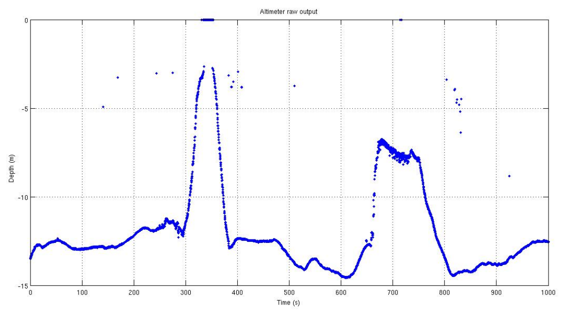


Fig. 4. Raw output of the altimeter; outliers are clearly identified

Given the filtering requirements, a choice for a one-dimension Kalman Filter came naturally, as it provides not only efficient smoothing, but also has the possibility to discard outliers by simply evaluating the covariance of the innovation. The state model for our systems is therefore uni-dimensional with its state being the depth, as given by the altimeter. The state model of the system was chosen to be a first order moving average:

$$z_{k+1} = z_k + u_k \tag{1}$$

Equation 1 tries to express the fact that the depth, z, should vary only by influence of the motion of the vehicle on the vertical plane. In that sense, $u_k = u \sin \psi + v \sin \psi$, ψ is the pitch angle of the vehicle and u and k are, respectively, the surge and heave movements of the vehicle.

The Kalman Filter algorithm is divided in to 2 different phases: the "time update", where the current state is projected ahead in time, according to the system model, and the "measurement update", where the projected estimate of the state is adjusted by an actual measurement. In our filter, new depth measures are available every 250 milliseconds, and the filter algorithm evolves according to the equations on (2);

$$\begin{aligned} S_{k+1} &= HP_kH^T + r \\ K_{k+1} &= P_kH^TS_k^{-1} \\ X_{k+1} &= X_k + K_k(z_k - HX_k) \\ P_{k+1} &= (I - K_kH)P_k \end{aligned} \qquad (2)$$

On the other hand, in between every two consecutive measurements, the filter will then evolve according to equations (3). For the filter in question, $[A] = 1$ and $[B] = 1$; also, as we can obtain our state, the depth, directly by our measures, z_k, then also $[H] = 1$. Due to the lack of information regarding the stochastic characterization of the altimeter, the measurement noise r was assumed to be constant and equal to 0.1, which corresponds to half the sensor quantum; the process noise q, on its hand, was adjusted to improve the performance of the filter in terms of rejection of the outliers and delay.

$$\begin{aligned} X_{k+1} &= AX_k + Bu_k \\ P_{k+1} &= AP_kA^T + q \end{aligned} \qquad (3)$$

A very important step of the Kalman Filter is the validation of the new measures, which can be performed by evaluating the covariance of the innovation, S_k. In fact, it is possible to define a parameter, γ, that will define whether a new measure, z_k, is valid and should be incorporated or if, on the other hand, should be discarded. Experimentally, it was found that setting γ to 0.5 offered the best results in terms of filtering noisy/spurious measurements. The obtained results are depicted in Figure 5.

$$\|z_k - Hx_k\| \, S_k^{-1} \leq \gamma \qquad (4)$$

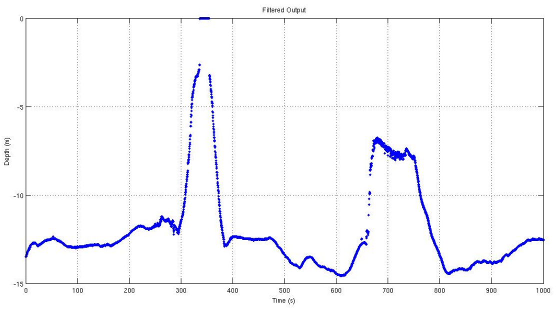


Fig. 5. Filtered output of the altimeter; outliers were removed

IV. Bottom Following and Control Strategy

As the ultimate goal of the present paper, we pretend to make the vehicle navigate at a given distance from the bottom. Typical applications, such as video acquisition for bottom inspection, requires the vehicle to be very close to the bottom in order to guarantee satisfactory results. Such problem is not trivial and becomes risky as the distance from the bottom decreases. The bottom-following missions foreseen and covered with this article are relatively simple. The MARES AUV controllers allow for a decoupled horizontal and vertical motions and our implemented bottom following technique takes advantage of that, by following a simple but effective approach. The Kalman filter outputs smoothed ranges to the bottom and consistent with the altimeter. The necessary references to the control of the vehicle will be generated by adding to these distances the depth of the vehicle Z_{PRS}, as given by the pressure sensor

$$Z_{REF} = Z_{PRS} + Z_{ALT} - D_f \qquad (5)$$

In (5), D_f is the parameter that sets the distance to the bottom that the vehicle should maintain, and in this particular case it was set to 1.5. Besides collision with the bottom, we also wait to prevent situations of possible trap or loss of the vehicle. Therefore, Z_{REF} is bounded so that the vehicle is not allowed to submerge more that what is considered a safe depth. This value will obviously vary according to the environment where the missions are being executed. In a similar fashion, it is also important that the references don't vary in a very rough way. In that sense, $\dot{Z}_{REF}$ is also bounded.

V. Results

The results here presented are the outcome of a set of field trials with the MARES AUV, conducted during August 2011 in a dam in Douro river, located at the eastside Porto, in the north of Portugal. Due to some previous missions performed in the same place, the morphology of the bottom of the river is known with some detail thus making it an appropriate setting for the tests. The MARES AUV was equipped with the aforementioned altimeter, and the appropriate C++ routines were developed and integrated with the onboard navigation and control software of the AUV.

One of the challenges inherent to this work is to assess the accuracy of the ranges measured by the altimeter, since a wrong configuration of its parameters might lead to incorrect measurements. Even though this was tested in a small tank in the lab facilities, the parameters under use in the river were radically different from those ones. Reasons for this are the difference in the water parameters, like salinity and temperature, and also the characteristics of the bottom. A pressure sensor integrates the standard equipment carried, being straightforward to extrapolate the actual depth of the AUV relative to the surface. By comparing the measurements of depth given by this sensor with the expected motion of the vehicle and the ranges given by the altimeter, it is possible to check if both measures are consistent, thus allowing to verify if the altimeters is properly configured.

A couple of standard bottom-following missions were initiated at different times and different locations of the river, trying to depict different case-scenarios of operation. In all of them the purpose was to follow the bottom at a distance of 1,5 meters, the distance thought to be the appropriate for the envisioned applications. In Figures 7 to 9 the ranges to the bottom, measured by the altimeter, and the depth of the AUV, measured by the pressure sensor, were plotted along the time, allowing to verify the behaviour of the AUV for several bottom-following manoeuvres. Figure 7 depicts a situation when the AUV was initially at the surface level, and the waters were 8m deep. It is clear from the plot that at around 840s a bottom-following manoeuvre was initiated, and the AUV started to descend steadily, keeping at the same time a constant horizontal speed. At around second 890, the depth is of approximately 9 meters, and the range to the bottom of about 1.5 meters, consistent with what was expected. A similar situation is depicted in Figure 8, but at greater depths. The trajectory performed by the AUV on the vertical plane presents a small a subtle, but noticeable, overshoot and subsequent oscillation when trying to follow a given reference in depth. Similar oscillations can be seen on both figures 7 and 8. This effect can be observed on both range measurements to the bottom and depth measurements, and they are a good indicator on the AUV ability to follow smooth variations of the bottom profile.

More interesting, perhaps, are the results shown on Figure 9: it can be seen that initially the range to the bottom was of approximately 2 meters, while the AUV was in the surface. As the bottom-following manoeuvre was initiated the depth of the AUV keeps increasing steadily, while the range to the bottom slightly decreases to about 1.5 meters, with small variations throughout the duration of the mission. This is actually a very impressive result, as it clearly demonstrates the ability to follow arbitrarily changing bottoms. This, of course, as long as variations of the slope of the terrain are not too rough, and remain compatible with the maximum actuations that the vehicle can withstand. As opposed to the previous figures, where the profile of the bottom was more or less stable, here the AUV follows a descending profile correctly. Also in this figure, it is clear that smooth changes in the bottom profile are correctly followed by correspondent changes in the AUV trajectory, for example around second 3100.

VI. Conclusion

This paper describes an effective bottom-following method for AUVs using an altimeter. Moreover, and by equipping the MARES AUV with such sensor, it was possible to experimentally verify that this solution achieves good results as the vehicle performs trajectories that closely resemble the profile of the bottom of Douro river. The first challenge was to configure the altimeter in a proper way and tuning its parameters to adequate levels. The next step was the use of an adequate filter to smooth the ranges measured by the altimeter, removing at the same time obvious outliers. In that sense, the ability of the Kalman Filters to reject measures by evaluating the covariance of the innovation, revealed itself to be fundamental. Finally, the whole algorithm was integrated with the onboard control software of the MARES AUV, and tests were conducted, that allowed to tune all the parameters for optimal results.

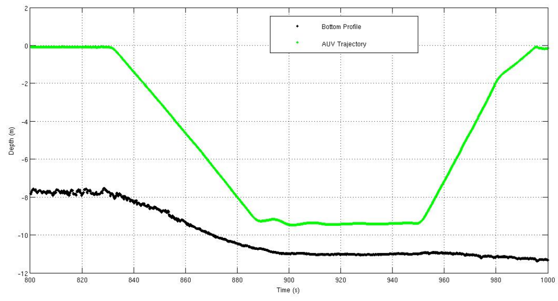


Fig. 6. Bottom of the river and the correspondent trajectory performed by the vehicle

All the applications that require to closely follow the bottom of the sea or rivers are likely to find use in the work here presented, for example the inspection of underwater structures. In addition, traditional bathymetry tasks could also be performed in this way, as exemplified in figure 6. There, the trajectory of the vehicle can be seen together with the profile of bottom of the river, obtained by combining depth data of the pressure sensors and ranges to the bottom. Possibilities for future work are quite encouraging. On one hand, a more complex Kalman Filter, containing a model for the sea bottom could be used. Using this information would allow to infer about the upper bounds on maximum depth, velocity and pitch that the vehicle should pose, to properly map the seabed. Moreover, depending on the confidence on the bottom depth measure and on the rugosity, the vehicle's pitch could be adjusted to anticipate possible unexpected obstacles or sudden slope variations. However, practical limitations on actuation would naturally bound the pitch angle.

Acknowledgment

The first author was supported by the Portuguese Foundation for Science and Technology through the Ph.D. scholarship SFRH / BD / 70727 / 2010.

References

[1] V. Creuze, B. Jouvencel, and P. Baccou, "Seabed following for small autonomous underwater vehicles," in *OCEANS, 2001. MTS/IEEE Conference and Exhibition*, vol. 1, 2001, pp. 369 –374 vol.1.
[2] C. Silvestre, R. Cunha, N. Paulino, and A. Pascoal, "A bottom-following preview controller for autonomous underwater vehicles," *Control Systems Technology, IEEE Transactions on*, vol. 17, no. 2, pp. 257 –266, march 2009.
[3] J. Gao, D. Xu, N. Zhao, and W. Yan, "A potential field method for bottom navigation of autonomous underwater vehicles," in *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*, june 2008, pp. 7466 –7470.
[4] A. J. Healey, "Obstacle avoidance while bottom following for the remus autonomous underwater vehicle," *Security*, vol. 1, p. 1, 2004.
[5] M. Caccia, G. Bruzzone, and G. Veruggio, "Active sonar-based bottom-following for unmanned underwater vehicles," *Control Engineering Practice*, vol. 7, no. 4, pp. 459 – 468, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0967066198001683
[6] J. Melo and A. Matos, "Guidance and control of an asv in auv tracking operations," in *OCEANS 2008*, sept. 2008, pp. 1 –7.

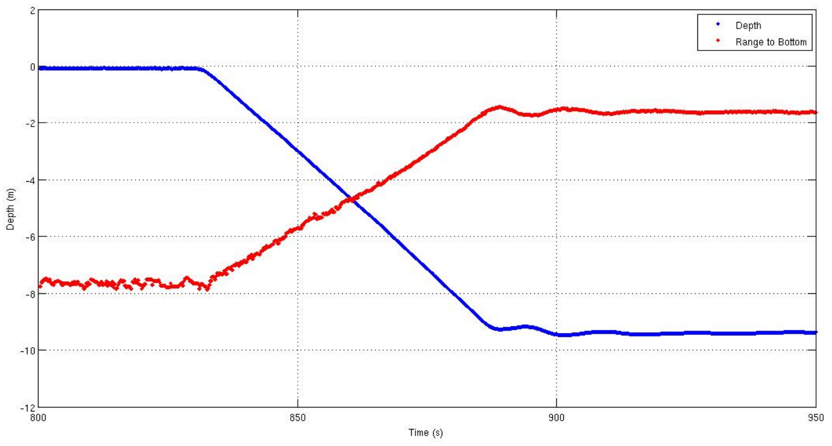


Fig. 7. Example 1 for depth and ranges to the bottom acquired over time

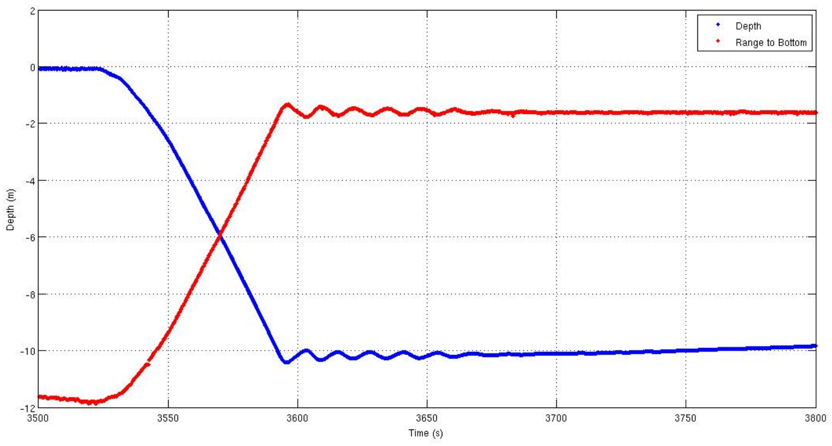


Fig. 8. Example 2 for depth and ranges to the bottom acquired over time

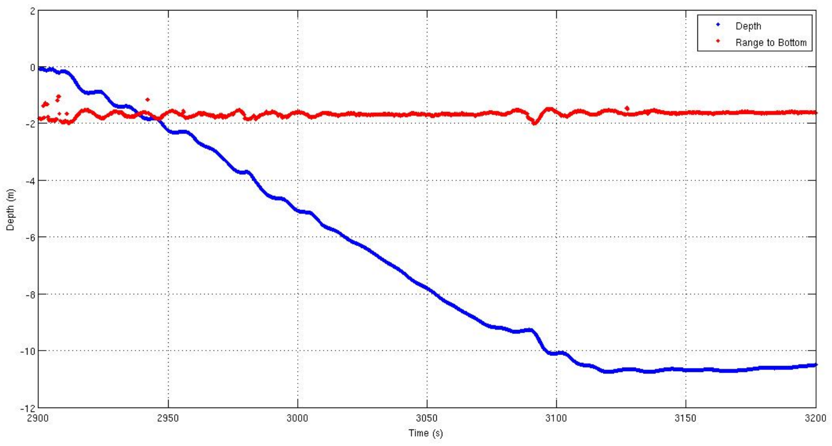


Fig. 9. Example 3 for depth and ranges to the bottom acquired over time

Modeling and control of TriMARES AUV

Bruno M. Ferreira, Aníbal C. Matos, Nuno A. Cruz
INESC TEC (formerly known as INESC Porto),
Faculty of Engineering, University of Porto,
Rua Dr. Roberto Frias, 378
4200-465 Porto, Portugal
Email: bm.ferreira@fe.up.pt

***Abstract*—In robotics, models frequently play a central role both in analysis and development. In this paper, we derive the model of TriMARES, a new inspection AUV with five degrees of freedom (DOF). We describe the complete six DOF model, derive and present the corresponding parameters and coefficients. This model was used to develop the controllers that currently govern the motion of TriMARES. We present the main ideas behind such control laws and show experimental results obtained from validation tests under typical conditions.**

I. Introduction

Dynamic, robust and high performance control commonly requires the use of mathematical models. Those models capture the dynamics and the kinematics of the moving platform and can be used for both localization and control purposes. Approximated linear models often fail to accurately represent the dynamics of an object over a wide region of operation, in the state space domain. This becomes particularly consequential for the motion of marine vehicles as a result of the dominating nonlinear terms.

The present paper shows the derivation of the model of the TriMARES AUV [1], a five degrees of freedom (DOFs) robot for dam inspection, developed by INESC TEC. The modularity feature, which was exploited in its predecessor, the MARES AUV [2], [3], was one of the key concepts that led to the present shape. Inspection tasks generally require high maneuverability and robust stability. The TriMARES' body and the thruster configuration are a result of careful contemplation of typical maneuvers that can be performed by the AUV such as station keeping, hovering, lateral or frontal scan, just to cite a few.

II. Modeling

Some considerations about the body shape can considerably simplify the model derivation. In this section, we will first start by highlighting TriMARES geometric characteristics and subsequently determine the model parameters.

A. Symmetries

For the design of TriMARES body shape, we have identified the following major requirements:

- Efficiency
- Maneuverabilty
- Stability
- Space availability for integration of user's new sensor

As a result of the first item, the vehicle is based on three streamlined bodies to minimize the viscous damping along the natural (longitudinal) axis. The disposal of the thrusters is strongly related to maneuverability and stability considerations. Body assymetries are particularly unwanted since they induce couplings between different DOFs as a consequence of hydrodynamic cross-terms. Therefore, the vehicle was built so that it is symmetric along the vertical xz-plane in order to avoid cross-effects on yaw or sway dynamics and thus avoiding undesired thrust compensation to cancel out such effects.

The vehicle symmetry is particularly usefull for model simplification, as we will see next.

B. General model

In this subsection, we present the general model of TriMARES, based on the standard formulation [4].

Let us consider the motion of a mobile robot in the tridimensional space. Define $\{I\}$ as the inertial referential frame and $\{B\}$ as the body fixed referential frame with origin coincident with the center of gravity and the x and y-axes being coincident with the surge and sway axes. The robot's absolute linear position in $\{I\}$ is denoted by the vector $\eta_l = [x_I, y_I, z_I]^T \in \mathbb{R}^3$, while its angular position is denoted by $\eta_a = [\phi, \theta, \psi]^T \in \mathbb{R}^3$. The relative linear and angular velocity vectors of the robot, expressed in the $\{B\}$ frame, are

B. Ferreira is supported by the Portuguese Foundation for Science and Technology (FCT) through the PhD grant SFRH/BD/60522/2009.

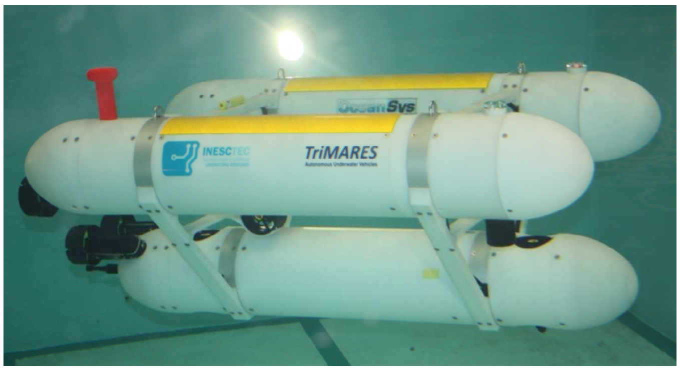


Fig. 1. TriMARES

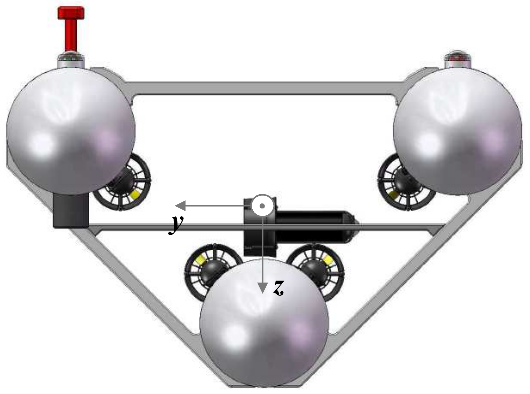


Fig. 2. Bow view of TriMARES

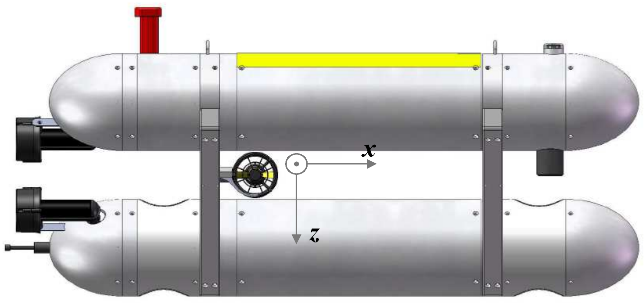


Fig. 3. Starboard view of TriMARES

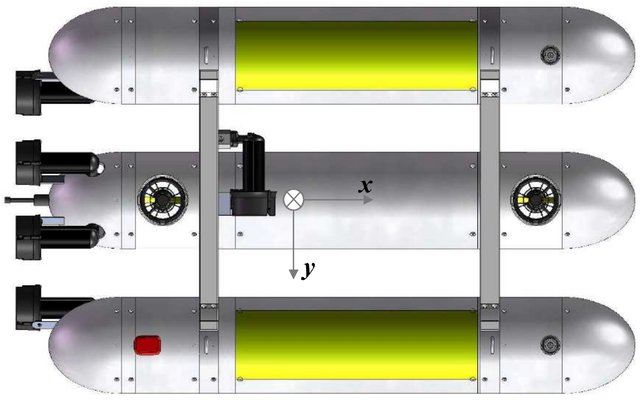


Fig. 4. Top view of TriMARES

given by $\nu_l = [u, v, w]^T \in \mathbb{R}^3$ and $\nu_a = [p, q, r]^T \in \mathbb{R}^3$, respectively. During operation, the robot is assumed to be subject to the effects of drifts that are represented by $\nu_c = [v_x, v_y, v_z]^T \in \mathbb{R}^3$, expressed in the $\{I\}$ frame. Introducing the orthogonal rotation matrix $J_l(\eta_a)$ (i.e., $J_l^{-1}(\eta_a) = J_l^T(\eta_a)$) from $\{B\}$ to $\{I\}$ parametrized by η_a, the velocity vectors in both referential frames $\{I\}$ and $\{B\}$ are related through the following expression (see [4]):

$$\dot{\eta}_l = J_l(\eta_a)\nu_l + \nu_c. \tag{1}$$

Similarly, the angular velocity relation is given through the transformation matrix $J_a(\eta_a)$ (see [4] for further details):

$$\dot{\eta}_a = J_a(\eta_a)\nu_a. \tag{2}$$

Mobile robots are governed by a second order differential equation which incorporates the effects of damping, added mass, restoring and applied forces and moments. The relation between these forces and the rigid body dynamics is given by the following six degrees of freedom (DOF) equation:

$$M\dot{\nu} = -C(\nu)\nu - D(\nu)\nu - g(\eta) + \tau, \tag{3}$$

where $\eta = [\eta_l^T \ \eta_a^T]^T$ and $\nu = [\nu_l^T \ \nu_a^T]^T$ are the concatenation vectors of the absolute position and of the relative velocity, respectively, τ is the vector of actuation forces and moments, $M, C(\cdot), D(\cdot) \in \mathbb{R}^{6\times6}$ and $g(\cdot) \in \mathbb{R}^6$. M is a positive definite matrix, i.e. $x^T M x > 0 \forall x \in \mathbb{R}^6$. The matrix M is due to rigid-body and added mass forces and moments, $C(\cdot)$ is originated by the Coriolis and centriptal effects, $D(\cdot)$ contains the terms related to viscous damping and $g(\cdot)$ is the vector of restoring forces and moments originated by the weight and the bouyancy forces.

Some considerations have to made at this point, in order to simplify the model. The following assumptions will stand for the rest of the present paper:

1) The cross relation effects are mainly induced by the three main composing bodies (tail, cynlinder and nose) and connecting *bars*
2) The composing bodies are sufficiently far away from each other and the displaced fluid does not influence the dynamics of the remaining ones
3) The influence of the protuberances on the bodies is small compared to the remaining forces and moments and can be neglected (note that we are not including the connecting bars)
4) The origin of the body-fixed frame coincides with the center of gravity of the vehicle

The item 2) lets us assume that there is no influence on heave dynamics in pure-surge motion and vice-versa. Therefore the combined inertia, M_{RB}, and added mass, M_A, matrix $M = M_{RB} + M_A$ is given by (4).

From assumption 4) and from (4), the derivation of the matrix of centripetal and Coriolis terms $C(\cdot)$, resulting from the sum of the terms from rigid-body from inertia matrix and from added mass $C = C_{RB} + C_A$, is straightforward from [4].

Regarding the viscous damping matrix, we assume that linear components of damping are neglegible for the velocity under consideration. Similarly, the terms higher than second order are also considered to be small enough to be neglected. Therefore, the resulting nonlinear viscous damping matrix is expressed in (6).

The vector of restoring forces and moments is given in [4],

$$M = \begin{bmatrix} m - X_{\dot{u}} & 0 & 0 & 0 & -X_{\dot{q}} & 0 \\ 0 & m - Y_{\dot{v}} & 0 & -Y_{\dot{p}} & 0 & -Y_{\dot{r}} \\ 0 & 0 & m - Z_{\dot{w}} & 0 & -Z_{\dot{q}} & 0 \\ 0 & -K_{\dot{v}} & 0 & I_{xx} - K_{\dot{p}} & 0 & 0 \\ -M_{\dot{u}} & 0 & -M_{\dot{z}} & 0 & I_{yy} - M_{\dot{q}} & 0 \\ 0 & -N_{\dot{v}} & 0 & 0 & 0 & I_{zz} - N_{\dot{r}} \end{bmatrix} \tag{4}$$

$$C(\nu) = \begin{bmatrix} 0 & -mr & mq & 0 & -Z_{\dot{w}}w - Z_{\dot{q}}q & +Y_{\dot{v}}v + Y_{\dot{p}}p + Y_{\dot{r}}r \\ mr & 0 & -mp & +Z_{\dot{w}}w + Z_{\dot{q}}q & 0 & -X_{\dot{u}}u - X_{\dot{q}}q \\ -mq & mp & 0 & -Y_{\dot{v}}v - Y_{\dot{p}}p - Y_{\dot{r}}r & +X_{\dot{u}}u + X_{\dot{q}}q & 0 \\ 0 & -Z_{\dot{w}}w - Z_{\dot{q}}q & +Y_{\dot{v}}v + Y_{\dot{p}}p + Y_{\dot{r}}r & 0 & I_{zz}r - Y_{\dot{r}}v - N_{\dot{r}}r & -I_{yy}q + X_{\dot{q}}u + Z_{\dot{q}}w + M_{\dot{q}}q \\ +Z_{\dot{w}}w + Z_{\dot{q}}q & 0 & -X_{\dot{u}}u - X_{\dot{q}}q & -I_{zz}r + Y_{\dot{r}}v + N_{\dot{r}}r & 0 & I_{xx}p - Y_{\dot{p}}v - K_{\dot{p}}p \\ -Y_{\dot{v}}v - Y_{\dot{p}}p - Y_{\dot{r}}r & +X_{\dot{u}}u + X_{\dot{q}}q & 0 & I_{yy}q - X_{\dot{q}}u - Z_{\dot{q}}w - M_{\dot{q}}q & -I_{xx}p + Y_{\dot{p}}v + K_{\dot{p}}p & 0 \end{bmatrix} \tag{5}$$

$$D(\nu) = -\begin{bmatrix} X_{u|u|}|u| & 0 & 0 & 0 & X_{q|q|}|q| & 0 \\ 0 & Y_{v|v|}|v| & 0 & Y_{p|p|}|p| & 0 & Y_{r|r|}|r| \\ 0 & 0 & Z_{w|w|}|w| & 0 & Z_{q|q|}|q| & 0 \\ 0 & K_{v|v|}|v| & 0 & K_{p|p|}|p| & 0 & 0 \\ M_{u|u|}|u| & 0 & M_{w|w|}|w| & 0 & M_{q|q|}|q| & 0 \\ 0 & N_{v|v|}|v| & 0 & 0 & 0 & N_{r|r|}|r| \end{bmatrix} \tag{6}$$

by

$$g(\eta) = g(\eta_2) = \begin{bmatrix} (W-B)\sin\theta \\ -(W-B)\cos\theta\sin\phi \\ -(W-B)\cos(\theta)\cos\phi \\ y_B B\cos\theta\cos\phi - z_B B\cos\theta\sin\phi \\ -z_B B\sin\theta - x_B B\cos\theta\cos\phi \\ x_B B\cos\theta\sin\phi + y_B B\sin\theta \end{bmatrix}, \tag{7}$$

where W and B are the weight and the buoyancy forces, respectively. The vector $g(\cdot)$ can be considerably simplified if we consider that the roll angle is stable and equals zero ($\phi \approx 0$) and that some of the components of the position of the center of buoyancy are null (see table I).

Finally, the vector of forces and moments originated by thrust is given by

$$\tau = P\tau_a \tag{8}$$

where $\tau_a \in \mathbb{R}^{n_a}$ is the vector of thruster forces and $P \in \mathbb{R}^{6\times n_a}$ is given in (9). Note that we defined d_i as the vector that maps the direction of the force exerted by the thruster i and r_i the corresponding position both in the body frame. For the present case, $n_a = 7$ is the number of thrusters.

Some of the parameters are presented in the table I.

III. Derivation of the coefficients

In this section we derive the coefficients of the previous sections which relate the accelerations and the velocities with the forces and moments acting on the body. Much of the derivation, in particular for viscous damping terms, is based on semi-empirical formulas (see [5]). Several books provide experimental and empirical formulas for determination of the coefficients. See, for example, [6], [7], [8].

TABLE I
PARAMETERS

Parameter	Value	Units
m	$1.10 \cdot 10^{2}$	kg
W	$7.84 \cdot 10^{2}$	N
B	$7.88 \cdot 10^{2}$	N
x_B, y_B	0	m
z_B	$-3.50 \cdot 10^{-2}$	
z_1, z_2	$-4.69 \cdot 10^{-2}$	m
z_3, z_4	$9.97 \cdot 10^{-2}$	m
$y_1, -y_2$	$-2.29 \cdot 10^{-1}$	m
$y_3, -y_4$	$-7.67 \cdot 10^{-2}$	m
x_5	$-1.59 \cdot 10^{-1}$	m
x_6	$-3.31 \cdot 10^{-1}$	m
x_7	$4.38 \cdot 10^{-1}$	m

A. Added mass terms

The axial added mass coefficient $X_{\dot{u}}$ for an ellipsoid is given by the formula in [9]. Most of the equation for the derivation are empirical and imply the approximation of the body by an ellipsoid. For modeling purpose, and for this coefficient only, we assume that the three main bodies contribute with the major added mass force. The protuberances such as motors and antenna, as well as the connecting bars, are not considered. Therefore, we determine the axial added mass coefficient as

$$X_{\dot{u}} \approx \sum_{i=1}^{3} -\frac{\alpha_0}{2-\alpha_0} m/3, \tag{10}$$

where $\alpha_0 = \frac{2(1-e^2)}{e^3}\left(\frac{1}{2}ln\frac{1+e}{1-e} - e\right)$ and $e = 1 - (b/a)^2$, where $a = 1.21$m is the length of the main bodies and $b = 0.2$m their diameter.

Before starting deriving the remaining added mass coefficients, let us define the cross section area $a(x)$ in the

$$P = \begin{bmatrix} d_1 & \dots & d_{n_a} \\ r_1 \times d_1 & \dots & r_{n_a} \times d_{n_a} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ z_1 & z_2 & z_3 & z_4 & 0 & -x_6 & -x_7 \\ -y_1 & -y_2 & -y_3 & -y_4 & x_5 & 0 & 0 \end{bmatrix} \tag{9}$$

TABLE II
ADDED MASS COEFFICIENTS

Coefficient	Value	Units
$X_{\dot{u}}$	$-5.87 \cdot 10^0$	kg
$Y_{\dot{v}}$, $Z_{\dot{w}}$	$-1.11 \cdot 10^2$	kg
$K_{\dot{p}}$	$-1.90 \cdot 10^2$	kg·m^2/rad
$M_{\dot{q}}$, $N_{\dot{r}}$	$-1.26 \cdot 10^1$	kg·m^2/rad
$Y_{\dot{p}}$, $K_{\dot{v}}$	$-1.26 \cdot 10^1$	kg·m/rad, kg·m
$Y_{\dot{r}}$, $-Z_{\dot{q}}$, $-M_{\dot{w}}$, $N_{\dot{v}}$	$-5.00 \cdot 10^0$	kg·m/rad, kg·m
$X_{\dot{q}}$, $M_{\dot{u}}$	$1.35 \cdot 10^{-1}$	kg·m/rad, kg·m
$Y_{\dot{p}}$, $K_{\dot{v}}$	$-5.59 \cdot 10^1$	kg·m/rad, kg·m

plane defined by the set of points so that x is constant. The function $a(x)$ can be seen as the summation of the areas of an infinitesimal slice of the vehicle at x. Furthermore, we define $l(y,z)$ to be the length of the vehicle at point (y,z), i.e., $l(x,z) = \int_{x_0(y,z)}^{x_f(y,z)} 1dx$. Hence the added mass coefficient are given as follow:

$$Y_{\dot{v}} = -\rho \int_L a(x)dx \tag{11}$$
$$Z_{\dot{w}} = Y_{\dot{v}} \tag{12}$$
$$K_{\dot{p}} = -\rho \int_H \int_{y_0(z)}^{y_f(z)} \|(y,z)\|_2^2 \, l(y,z)dydz \tag{13}$$
$$M_{\dot{q}} = -\rho \int_L x^2 a(x)dx \tag{14}$$
$$N_{\dot{r}} = M_{\dot{q}}, \tag{15}$$

where ρ is the fluid density, L is the length of the vehicle, H is the height and $y_0(z)$ and $y_f(z)$ represent the limits of integration which are obviously functions of z.

The cross-related terms are derived through the following expressions:

$$Y_{\dot{p}} = -\rho \int_H \int_{y_0(z)}^{y_f(z)} \|(y,z)\|_2 \text{sign}(z) \, l(y,z)dydz \tag{16}$$
$$K_{\dot{v}} = Y_{\dot{p}} \tag{17}$$
$$Y_{\dot{r}} = -\rho \int_L x \, a(x)dx \tag{18}$$
$$N_{\dot{v}} = -M_{\dot{w}} = -Z_{\dot{q}} = Y_{\dot{r}} \tag{19}$$

As for $X_{\dot{u}}$, the derivation of $X_{\dot{q}}$ relies on approximations of the bodies by ellipsoids [4]. A rough approximation of the remaining coefficients is given by

$$X_{\dot{q}} = M_{\dot{u}} \approx \sum_{i=1}^{3} z_i \frac{X_{\dot{u}}}{3}, \tag{20}$$

where z_i denotes the position of the body $i, i = 1,2,3$, in the z-axis.

The table II lists the values of the added mass coefficient for the particular case of TriMARES.

B. Viscous damping coefficients

The viscous damping force is a function of the projected area. We define $l_{yz}(z)$ to be the integral of the drag coefficient over the length of the yz-projected (see fig. 2) area at position z, i.e., $l_{yz}(z) = \int_{y_0(z)}^{y_f(z)} C_D(y,z)dy$. Analogously, the integral of the drag coefficient over the height of the xz-projected (see fig. 3) area at position x is given by $l_{xz}(x) = \int_{z_0(x)}^{z_f(x)} h(x,z)C_D(x,z)dz$, where $h(x,z) : [x_0, x_f] \times [z_0(x), z_f(x)] \to \{1,2\}$. This function results from the fact that the projected bars and upper bodies contribute two times for the viscous damping. Indeed, assuming they are sufficiently far from each others, their contributions are equal and must be added. This is implicitly done by means of $h(z)$. Similarly, $l'_{xz}(z) = \int_{x_0(z)}^{x_f(z)} h(x,z)C_D(x,z)dx$ is the integral of the drag coefficient over the length of the xz-projected area at position z. We also define $l_{xy}(x) = \int_{y_0(x)}^{y_f(x)} C_D(x,y)dx$ to be the integral of the drag coefficient over the length of the xy-projected (see fig. 4) area at position x.

Actually, the drag coefficient C_D is a function of the shape and its value depends on the Reynolds number, on the length and on the diameter of the vehicle. In this paper, we will consider that it depends only on the shape. For the operating velocities under consideration, we consider $C_D = 0.2$ for ellipsoidal shapes moving in the same direction as their major axes, $C_D = 0.8$ for cylindrical shapes, and $C_D = 1.07$ for cubic shapes.

Hence, the axial viscous damping coefficients are given by

$$X_{u|u|} = -\frac{1}{2}\rho \int_H l_{yz}(z)dz \tag{21}$$
$$Y_{v|v|} = -\frac{1}{2}\rho \int_L l_{xz}(x)dx \tag{22}$$
$$Z_{w|w|} = -\frac{1}{2}\rho \int_L l_{xy}(x)dx \tag{23}$$
$$K_{p|p|} = -\frac{1}{2}\rho \int_H |z|^3 l'_{xz}(z)dz \tag{24}$$
$$M_{q|q|} = -\frac{1}{2}\rho \int_L |x|^3 l_{xy}(x)dx \tag{25}$$
$$N_{r|r|} = -\frac{1}{2}\rho \int_L |x|^3 l_{xz}(x)dx. \tag{26}$$

The equations for the cross relation coefficients are ex-

TABLE III
VISCOUS DAMPING COEFFICIENTS

Coefficient	Value	Units
$X_{u\|u\|}$	$-3.24 \cdot 10^1$	kg/m
$Y_{v\|v\|}$	$-3.03 \cdot 10^2$	kg/m
$Z_{w\|w\|}$	$-3.26 \cdot 10^2$	kg/m
$K_{p\|p\|}$	$-1.21 \cdot 10^0$	kg·m^2/rad^2
$M_{q\|q\|}$	$-1.75 \cdot 10^1$	kg·m^2/rad^2
$N_{r\|r\|}$	$-1.64 \cdot 10^1$	kg·m^2/rad^2
$X_{q\|q\|}$	$9.52 \cdot 10^{-2}$	kg·m/rad
$Y_{p\|p\|}$	$1.92 \cdot 10^{-1}$	kg·m/rad
$Y_{r\|r\|}$	$-5.867 \cdot 10^0$	kg·m/rad
$Z_{q\|q\|}$	$7.20 \cdot 10^0$	kg·m/rad
$K_{v\|v\|}$	$-6.32 \cdot 10^0$	kg
$M_{u\|u\|}$	$7.27 \cdot 10^0$	kg
$M_{w\|w\|}$	$1.31 \cdot 10^1$	kg
$N_{v\|v\|}$	$-1.08 \cdot 10^1$	kg

pressed as follow:

$$X_{q|q|} = -\frac{1}{2}\rho \int_H z|z|\, l_{yz}(z)dz \tag{27}$$

$$Y_{p|p|} = \frac{1}{2}\rho \int_H z|z|\, l'_{xz}(z)dz \tag{28}$$

$$Y_{r|r|} = -\frac{1}{2}\rho \int_L x|x|\, l_{xz}(x)dx \tag{29}$$

$$Z_{q|q|} = \frac{1}{2}\rho \int_L x|x|\, l_{xy}(x)dx \tag{30}$$

$$K_{v|v|} = \frac{1}{2}\rho \int_H z\, l'_{xz}(z)dz \tag{31}$$

$$M_{u|u|} = -\frac{1}{2}\rho \int_H z\, l_{yz}(z)dz \tag{32}$$

$$M_{w|w|} = \frac{1}{2}\rho \int_L x\, l_{xy}(x)dx \tag{33}$$

$$N_{v|v|} = -\frac{1}{2}\rho \int_L x\, l_{xz}(x)dx. \tag{34}$$

The values of the viscous damping coefficients are presented in table III.

IV. CENTER OF GRAVITY

In order to determine the center of gravity (CG), an experiment was carried out. The key idea was to suspend the vehicle from a single point. After having stabilized, the vector of gravity in the body frame has the same direction as the vector that joins the hanging point to the CG. Thus, performing the same experiment for several hanging points, the intersection of the lines that pass through the hanging points and have the same direction as the vector of gravity for the corresponding experiment will determine the CG. The method is illustrated in 5.

We take the origin referential frame $\{B_0\}$ so that its origin coincides with the intersection of the vertical plane of symmetry of the lower cylinder (coincident with the xz-plane in fig. 2-4) with the vertical plane tangent to bow (noses) and the horizontal plane tangent to the top of the vehicle. The axes x_0 and z_0 are taken so that they point in the forward and downward directions, respectively.

The corresponding data is given in table IV

TABLE IV
GRAVITY VERSOR EXPRESSED IN THE BODY FRAME FOR SEVERAL HANGING POINTS

Hanging point	Gravity vector (expressed in $\{B_0\}$)
$[-2.92, -3.00, -0.15]^T \cdot 10^{-1}$	$[-6.67, 5.71, 4.79]^T \cdot 10^{-1}$
$[-2.92, 3.00, -0.15]^T \cdot 10^{-1}$	$[-6.82, -5.86, 4.39]^T \cdot 10^{-1}$
$[-8.88, -3.00, -0.15]^T \cdot 10^{-1}$	$[5.31, 6.50, 5.43]^T \cdot 10^{-1}$
$[-8.88, 3.00, -0.15]^T \cdot 10^{-1}$	$[5.38, -6.73, 5.08]^T \cdot 10^{-1}$

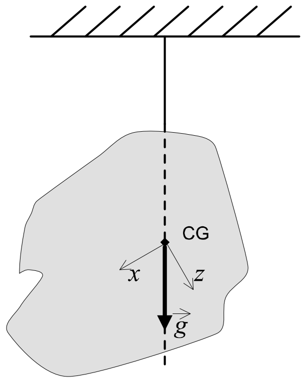


Fig. 5. Method for determination of the CG

The center of gravity, expressed in the frame $\{B_0\}$, is easily obtained by application of a regression technique and is given by $CG_0 = [-6.43, 0.07, 2.23]^T \cdot 10^{-1}$m.

V. CONTROL

Having focused on flexibility and versatility of TriMARES motion, the control algorithm was developed so that the several controllable DOFs can be handled in a decoupled manner. Such characteristic allows the end-user to perform any type of composed motion without having to be concerned with the low-level control nor the intrinsic complex dynamics. To illustrate the idea, suppose that we are interested in performing an elaborated trajectory in the horizontal plane which may involve position and velocity – typical trajectory tracking, for instance. However, the depth must remain constant over the operation and the pitch must be different from zero (e.g., nose pointing towards the seafloor). This task is made possible by setting the depth and pitch references constant, while the rest of the DOF references remain available and ultimately set by an external entity (this type of motion is particularly interesting for ROV mode operation).

As it could be preferable defining position or velocity references given in different referential frames, we consider two possibilities for each of the DOFs: inertial, earth-fixed referential frame and body-fixed referential. The control laws were derived in order to meet essential requeriments:

- Ability to move according to position references given in the earth-fixed frame
- Possibility of setting velocity references in the earth-fixed frame
- Capability of accepting velocity reference in the body-fixed frame.

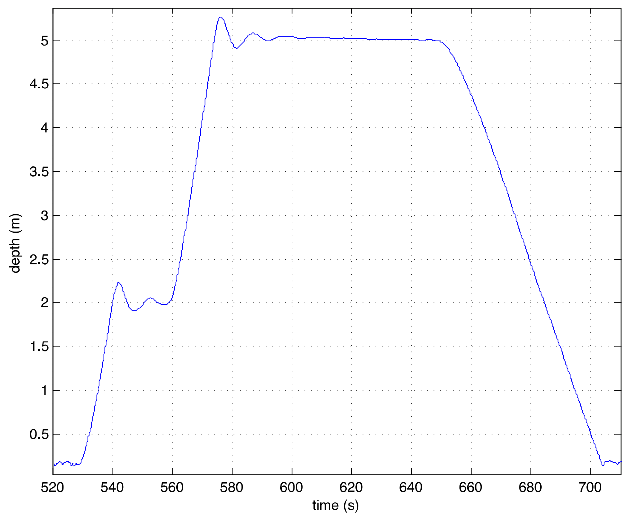


Fig. 6. Depth evolution (log20110520151408)

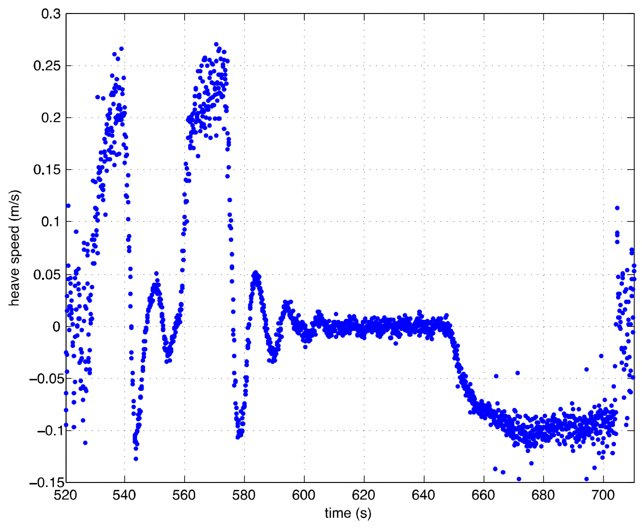


Fig. 7. Heave speed (log20110520151408)

The task of deriving control laws that satisfy such requirements is not trivial and becomes even more complex when the motion along the different axes is intended to be decoupled. The complexity mainly arises from the hydrodynamic coupling effects.

Our approach makes use of the nonlinear control theory [10], implementing the backstepping method to achieve robust and accurate control of TriMARES. This nonlinear control tool has already proven to be effective in our previous works on other vehicles. In particular, a similar approach was implemented in the MARES AUV (predecessor of TriMARES) in [3]. For the sake of brevity, we do not present the derivation of the control law in this paper. For further details, the reader is referred to [3], considering that the approach is similar.

The resulting control law is composed by a combination of feedforward and compensation terms. The former are obtained through the model derived in the previous sections. The latter are typically functions of the errors and gains which were posteriorly tuned to ensure adequate performance while taking into account the practical bounds on thrust actuation.

VI. Experiments

Experimental tests were conducted with TriMARES in the Douro river during spring 2011. The main objectives of the missions were validating the model and the controllers. Several tests were successfully carried out. The model has shown to be accurate and has fed the controllers with dead-reckoned estimates of forces. The accuracy was assessed through direct comparison of estimated and actual behaviors. We found, however, that the axial drag $X_{u|u|}$ is slightly below its actual value. This arises from the choice of C_D which is based on empirical formulas in the literature (see [7], [5]). Nevertheless, this coefficient can be very easily determined via more extensive experiments. The data that we have collected so far indicate that $C_D = 0.3$ would be a better estimate for the drag coefficient of the main bodies.

The fig. 6 shows the evolution of the depth for a mission in which the vehicle is set to hover at a reference depth of 2 meters during the first 30 seconds and is posteriorly switched to 5 meters for the subsequent 50 seconds. Thereafter, the controllers are switched off and the vehicle naturally surface due to the positive buoyancy. The fig. 7 shows the heave speed, w, during the maneuver. In both figures, the data is not filtered and the presence of noise on the depth sensor and on the inertial measurement unit (IMU) cause the observed noise.

VII. Conclusions

We have presented the complete six DOF model of TriMARES. The current controllers use the model to generate suitable commands for the set of seven thrusters on the vehicle. The combined operation of dead-reckoning of the dynamics, using the model, and the control laws has shown very satisfactory performances and has proven that the model is sufficiently precise to generate feedforward commands accordingly and guarantee the stability of TriMARES.

References

[1] N. A. Cruz, A. C. Matos, R. M. Almeida, B. M. Ferreira, and N. Abreu, “Trimares - a hybrid auv/rov for dam inspection,” in *OCEANS 2011*, sept. 2011, pp. 1 –7.
[2] N. A. Cruz and A. C. Matos, “The mares auv, a modular autonomous robot for environment sampling,” in *OCEANS 2008*, 2008, pp. 1–6.
[3] B. Ferreira, A. Matos, N. Cruz, and M. Pinto, “Modeling and Control of the MARES Autonomous Underwater Vehicle,” *MARINE TECHNOLOGY SOCIETY JOURNAL*, vol. 44, no. 2, SI, pp. 19–36, MAR-APR 2010.
[4] T. I. Fossen, *Guidance and control of ocean vehicles*. Wiley, 1994.
[5] T. Prestero, “Verification of a six-degree of freedom simulation model for the remus autonomous underwater vehicle,” Master’s thesis, Massachussets Institute of Technology, 2001.
[6] S. Hoerner, *Fluid-dynamic drag: practical information on aerodynamic drag and hydrodynamic resistance*. Hoerner Fluid Dynamics, 1965.
[7] F. White, *Fluid mechanics*. McGraw-Hill, 2003.
[8] O. Faltinsen, *Hydrodynamics of high-speed marine vehicles*. Cambridge University Press, 2005.
[9] F. H. Imlay, “The complete expressions for added mass of a rigid body moving in an ideal fluid,” David Taylor Model Basin, Tech. Rep., 1961.
[10] H. K. Khalil, *Nonlinear systems*. Prentice Hall, 2002.

Graph Grammars for Active Perception

Luis J. Manso
University of Extremadura,
Cáceres, Extremadura.
lmanso@unex.es

Pablo Bustos
University of Extremadura,
Cáceres, Extremadura.
pbustos@unex.es

Pilar Bachiller
University of Extremadura,
Cáceres, Extremadura.
pilarb@unex.es

Marco A. Gutierrez
University of Extremadura,
Cáceres, Extremadura.
marcog@unex.es

Abstract—**The complexity of the applications in which robots are being used does not stop growing. Different solutions such as sophisticated control architectures have been proposed in order to deal with complexity in robot control. These solutions make robotic systems more robust, scalable and easier to distribute, understand and monitor. However, it is still not clear how to cope with the complexity of the interaction dynamics that underlie the perception of the environment. With this issue in mind this paper presents the concept of *cognitive graph grammar* and two algorithms that make use of it. Cognitive graph grammars are a grammar-based theoretical framework designed to support cognitive perception and, especially, the active nature of perception. They provide a means to describe how graph-based models can be generated and the behaviors to execute depending on the perceptual context. This is done in such a way that the information provided using this formalism can be used for different perceptive purposes at the same time, such as to link action and perception or to diminish perceptive errors. The paper also describes an experiment in which a cognitive graph grammar is used in an autonomous robot in order to efficiently model an environment made of rectangular rooms with obstacles.**

I. Introduction

It would be desirable to have robots able to interact with non-trivial entities (e.g., compound objects, people). In order to perform these tasks robots have to perceive and model their environment to some extent. Unlike in the earliest experiments of robotics, the floor is not necessarily restricted to textureless shadow-free surfaces anymore, and objects are not necessarily simple perfectly shaped boxes. Nevertheless, the environments in which robots operate are not random. In order to build actually intelligent robots, a priori knowledge about the environment must be properly used.

The complexity that roboticists have to face when developing autonomous robotic systems has been successfully handled from different points of view. Control architectures such as those in [1], [2] suggest how to organize control and information flows. Technologies such as [3], [4] handle implementation issues from a software engineering point of view using component-oriented programming. However, none of these approaches can be directly used as a tool to support perception or to ease the binding of perception and action. The control logic of active perception algorithms still tends to be formed by hard-coded if-then-else constructs that map robot proprioception and its world model to specific perceptual states and actions, which is error-prone. Moreover, they rarely take into account context information, which is useful to produce robust and coherent environment interpretations. Thus, despite the use of the previously mentioned technologies makes robotic systems better designed and easier to manage, the complexity of the control logic associated with the perception of the environment is hardly reduced. This paper presents *cognitive graph grammars*, a theoretical framework that helps roboticists building context-aware active perception systems.

When a robot builds symbolic models of its surroundings it generally does so by recording the perceived environment elements and their relationships. Since this can be seen as the generation of a graph where nodes represent the modeled symbols and edges represent the relationships between them, it is interesting to formally describe how the robot might do that. Precisely, graph grammars describe the rules governing the formation of graphs with a specific structure. Graph grammars generalize the concept of string grammars so that productions can also be applied to graphs. In fact, strings can be seen as undirected graphs such that all nodes –characters–, except those at sentence endings, have an edge linking them to each of their adjacent characters. Thus, graph grammars extend string grammars in order to support input data with arbitrarily complex connection patterns.

This paper introduces the concept of cognitive graph grammar (CGG), a graph grammar-based formalism designed to support symbolic active perception. They provide a means to give raise to different active-perceptual mechanisms by describing how graph-based models can be generated. Building on this formalism, additional algorithms are provided so that descriptions based on CGG can be used for different purposes, such as linking action and perception or improving perception robustness. The paper also provides a proof-of-concept experiment in which a CGG is used to model an environment made of rectangular rooms with obstacles.

The remaining of the paper is organized as follows. Section II reviews previous work on graph grammars and active perception. The core of the paper is found in section III. It provides an introduction to the most widely used graph grammar formalisms, describes the limitations that make necessary a new one, and details the concept of cognitive graph grammars both from a formal and practical point of view. It also describes the different benefits of using CGGs and the perceptual phenomena that can arise when using them. Section IV describes an example of a CGG used in order to perform topological mapping along with the experimental results obtained using it. Finally, section V presents the conclusions and future work.

II. Previous Work

Besides active perception, grammars have been used in robotics and artificial vision for a wide range of applications. An algorithm for graph verification (i.e., given a graph and a graph grammar, check if the graph can be generated using the grammar) is proposed in [6]. In this work, only the vertices of the graphs can be labeled. Graph grammars are used in [7] to achieve self-configuring adaptable software architectures. In this work graphs are of fixed order. A similar approach for coordinating multi-robot systems where robots are represented by graph vertices is proposed in [8]. The graph, which is shared by all the robots of the system, is also of fixed order. Coordination is achieved by modifying the linking pattern and the label (i.e., role) of the robots.

A series of works by the same group is presented in [9], [10] and [11]. In [9] and [11], an attributed graph grammar is designed in order to parse rectangle layouts from images of man-made scenes. The authors consider rectangles as terminal symbols and layouts as production rules. Bottom-up and top-down mechanisms are used in order to improve rectangle detection and parsing. Since different possible models can explain input images, the algorithm chooses the one maximizing the posterior probability or minimizing a descriptor length. A similar approach is used for segmenting and recognizing generic scenes in [10]. A similar approach to the one described in [11] is used in [12] in order to represent and recognize objects. In this case, both the set of primitives and production rules are wider, but the foundations are the same. These approaches have two main differences from what is proposed in this paper: a) they are based on string grammars, reducing what can be solved using their approach; b) they use static input data, which is a very hard restriction for robotics (action is not taken into account).

Graphs are used for task planning in [13]. It also covers how plans can be dynamically modified as sub-tasks are accomplished or conditions change. Grammar rules are proposed in order to modify the current plan.

Spatial Random Tree Grammars are proposed in [14] for image parsing. They are context-free grammars with at most two children in which rules are labeled with information for determining the spatial distribution of their nodes (i.e., vertically or horizontally distributed). While in string grammars this is not necessary (i.e., productions are always horizontally distributed), it guarantees the unambiguous interpretation of parse trees from graph grammars. In this work, a probability distribution is also associated to production rules so the probability of a specific parse can be estimated.

III. Cognitive Graph Grammars

Unlike graph grammars, string grammars do not generally provide enough expressive power to be used in order to build environmental representations. On the other hand, string grammars unambiguously describe how the production rules can be applied. This is because in these grammars two symbols are connected if and only if they are in contiguous positions within the sentence. When a string symbol pattern p_1 is replaced by p_2 it is automatically linked to –and only to– the pattern on the left and right sides of p_1, in the same order. Depending on the formalism used, this assumption does not hold for graph grammars since the connection pattern is not fixed or restricted to a specific order. Additionally, none of the previous graph grammar formalisms were conceived with robots or active perception in mind. Different graph grammar formalisms have been proposed aiming to remove any ambiguity in the connection patterns of the nodes involved [5]. However, they do so by assuming a specific behavior: some of them assume that dangling edges should be automatically removed; others assume that dangling edges prevent productions from being applied. In order to avoid ambiguity without limiting what can be expressed by graph grammars they must also allow to specify negative subpatterns (i.e., parts of the patterns that should not exist in order to apply the rule). Since no formalism provides mechanisms to remove these restrictions, it was found necessary to create a more flexible formalism.

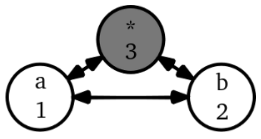


Fig. 1. Example of how the 'avoid' decoration and the wildcard symbol can be used to dismiss any pattern not specifically considered on the right-hand side of grammar rules.

A. CGG formalism

The cognitive graph grammar formalism follows the same principles as single-pushout [5] but, in order to overcome the previously mentioned limitations three modifications are proposed: **a)** the elements of the left hand side of the rules can be decorated with an *avoid* attribute in order to dismiss those matches containing elements decorated in such way (expressed by filling them with gray color); **b)** productions may be accompanied with first-order logic sentences specifying *conditions* and the *operations* performed by the rule if the desired behavior differs from the one of single-pushout; **c)** CGGs have the '$*$' NT symbol, which matches any other symbol. Figure 1 and table I(e) are examples of the use of the avoid attribute. From a formal point of view, cognitive graph grammars are defined as a seven tuple $(N, T, P, B, A_V, A_E, PB)$ such that:

- N and T are the mutually exclusive non-terminal and terminal alphabets, respectively. In particular, N must contain the start and wildcard symbols (S and $*$).
- P is the set of production rules.
- B is the set of possible perceptive behaviors.
- A_V and A_E are the attribute alphabet for vertices and edges, respectively.
- PB is a function mapping P to B.

This formalism allows to unambiguously specify a graph grammar without assuming any specific behavior, as well as to relate behaviors to grammar rules.

B. CGGs properties

As seen in section II, grammars can be used as a tool to support perception. This section describes how to achieve different grammar-based perception-oriented techniques using CGGs. To the knowledge of the authors, all published grammar-based perception techniques can be classified in one of the following types: a) bottom-up parsing, b) model verification, c) context-aware perception restrictions, and d) covert perception. In addition to these, CGGs can also be used to associate perception and action. The remaining of the section describes how to achieve each of these phenomena using CGGs.

Bottom-up parsing is the most common application of grammars: given a sample from the input space, it is parsed in order to recognize its structure. For some applications it might only be necessary to perform **model verification**. It can be seen as a bottom-up parsing where the parse result is ignored, only providing whether or not there was any result.

It is also desirable that robot perception would dynamically adapt to the scenario in which robots are located and their conditions. Here we distinguish between a) passively adapting to the environment by restricting what might be perceived, **context-aware restrictions**; and b) high perceptual layers actively providing top-down information to the bottom perceptual layers in order to influence its output, **covert perception** [16].

Additionally, we propose using CGGs to select the appropriate perception strategy or behavior according to the context. The subset of rules that can be potentially triggered next can be computed by analyzing the grammar rules and the current model. Thus, by associating behaviors to rules, the compatible behavior set is computed as the set of behaviors associated to the rules that can be potentially triggered. It is worth noting that all previous work regarding the use of grammars for perception were applied to static images. The remaining of the section elaborates how these perceptual techniques can be implemented using CGGs.

1) Bottom-up parsing: Regular parsing algorithms are designed to work with static and complete input data. Robot perception generally entails movements that allow robots to sense different parts of the environment. Since these movements change the input data, the standard approaches can not be used to parse it. The process of bottom-up parsing in CGGs is performed by running the rules that are compliant with the current model as long as the terminal symbols they introduce are actually being perceived. When multiple possible rules can be triggered at the same time, the approach presented in [14] can be used in order to provide the most probable parse.

2) Context-aware restrictions on perception: A classic example of this kind of phenomena in humans can be found in [15]: the same visual input can be perceived as different objects depending on the context. Graph grammars express how graphs (symbolic models in our case) can be built. By doing so they also describe how they can not be built, thus providing the power to support context-aware perception. By restricting which world elements can be perceived at each moment according to the limitations of the grammar, the number of false positives (i.e., misrecognized world elements) can be reduced. In order to illustrate this, a simple two-rule grammar example is provided in equation 1. The formalism is not used in this case because this specific property can be provided by all kind of grammars, not just CGGs.

$$\begin{aligned} S &\Longrightarrow arm \cdot A \\ A &\Longrightarrow forearm \end{aligned} \tag{1}$$

If a robot using this grammar is certain that it perceived the arm of a person (so its current model is $'arm \cdot A'$) it can unquestionably discard any other arms. A pseudo-code implementation is shown in algorithm 1. For every potentially applicable rule, it computes the set of non-terminal symbols appearing only on the RHS (not on the LHS) and returns the union of those sets. This set contains the world elements that can be perceived given the grammar and the current model.

Algorithm 1 Contex-aware restrictions algorithm:

```
Require: h: Input graph
Require: G: CGG such that $G = (N, T, P, B, A_V, A_E, PB)$
 1: $U \leftarrow \emptyset$
 2: forall $p = (lhs, rhs) \in P$ do
 3:   if $applicable(p, h)$ then
 4:     forall $s \in (rhs.V - lhs.V)$ do
 5:       if $terminalSymbol(s)$ then
 6:         $U \leftarrow U \cup s$
 7:       end if
 8:     end forall
 9:   end if
10: end forall
11: return $U$
```

3) Covert perception: The information provided by grammars can also be used to influence bottom-up perception, not just to filter its output. Thus, grammars are also a valid framework to enable covert perception [16]. A priori knowledge of the world can be used to influence bottom-up perception by enforcing or inhibiting the detection of specific parts of the environment. Thus, grammars can not only reduce false positives in object detection but also reduce false negatives. The grammar rule described in table I(b), is a good example of this. It is further explained in section IV. When parts of the object to detect are occluded, bottom-up object detectors tend to decrease their effectiveness dramatically. The partial pattern detected can be used to compute the probability of a false negative. This can be expressed using the Bayes theorem as in equation 2. In the equation, F stands for the event "a not detected entity should be forced into the model", and C stands for the event of the robot having a specific context (potentially partial input).

$$p(F|C) = \frac{p(C|F)p(F)}{P(C)} \tag{2}$$

This is one of the most interesting applications of graph grammars. Examples of this type of technique can be found in [9], [10], [11] or [12].

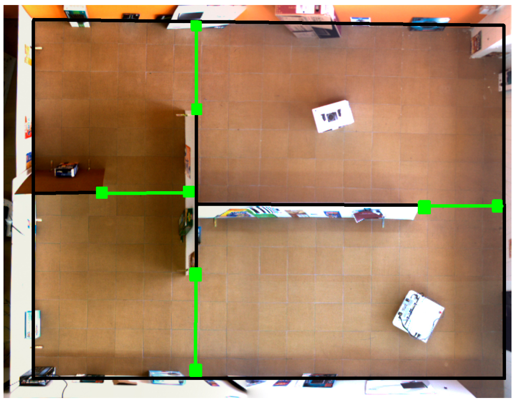


Fig. 2. Overhead view of the environment used for the experiments. It is composed of four rectangular rooms connected in a loop. The robot is the squared object on the right-bottom part of the image. Objects are placed in the environment in order to prove the robustness of the approach.

4) Action selection: Generally, given the grammar and the current graph, only a subset of rules can be potentially applied. Since CGG rules have associated behaviors, computing the subset of potentially applicable rules is equivalent to compute the candidate set of perceptual actions. It can be used to enable robots to decide what to do next. Depending on the grammar and the context, the subset can be formed by several or a single action. Algorithm 2 details how to compute the action set.

Algorithm 2 Action selection algorithm:

```
Require: h: Input graph
Require: G: CGG such that $G = (N, T, P, B, A_V, A_E, PB)$
1: $U \leftarrow \emptyset$
2: forall $p \in P$ do
3:    if $applicable(p, h)$ then
4:       $U \leftarrow U \cup PB(P)$
5:    end if
6: end forall
7: return $U$
```

The novelty of the underlying idea of CGGs in this respect is not the well-known idea of associating behaviors to robot states, but to do it by defining the perceptive state of robots as their set of potentially applicable rules. Moreover, algorithm 2 can be easily extended to only take into account those rules which are of interest depending on the robot goal.

IV. Experiment

In order to illustrate the usage of CGGs, this section provides a real example of a grammar used in a robot that models its environment. In particular, the objective of this grammar is to enable a robot to model a simple world made of rectangular connected rooms in which obstacles can be found. Section IV-B describes the benefits obtained from the use of CGGs. Section IV-C provides experimental results obtained using this grammar.

A. Grammar definition

1) Grammar alphabets: The first step is to define the entities that the world model will be composed of. This is an arbitrary decision: it is up to the roboticist to decide which symbols to use. It will depend on the environment or object to model, and the desired level of detail. For this experiment it was decided to use symbols for rooms, doors and obstacles.

- **r** Used for rooms.
- **d** Used for doors. Doors will link two different rooms.
- **o** For obstacles. Obstacles will be located within rooms.

It is also necessary to define the attributes that the symbols will have. In particular, rooms, doors and obstacles have their position (x and y) and their dimensions ($width$ and $length$). Moreover, rooms also have an $active$ attribute that is true for the room in which the robot is located and false otherwise. As can be seen in table I, attributes can be used in order to enable or disable the application of the different rules. The resulting alphabets are shown in equation number 3.

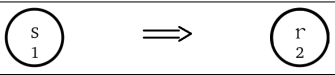
$$\begin{aligned} N &= \{S, *\} \\ T &= \{r, d, o\} \\ A_V &= \{width, length, x, y, active\} \\ A_E &= \emptyset \end{aligned} \tag{3}$$

2) Grammar rules: Once the entities to model are known, the next step is to write the rules that will guide the perception process. In most cases this is a cyclic task (i.e., the symbols may depend on the rules and vice versa), and sometimes it will be necessary to introduce new non-terminal symbols.

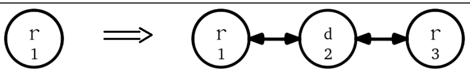
Rule 0 specifies that the start symbol can be transformed into a room. This is the only rule with the start symbol (S) in its left hand side, so the start symbol *can only* be transformed into a room. This implies that, when the perception process starts, the first task robots have to perform is to model the room in which they are located. Rule 1 describes how the discovery of new rooms transforms the model. The rule creates a new room symbol and makes it adjacent to the room in which the robot is located. It is triggered when the robot perceives or goes through a door when there is only one room in the model or when there are no close rooms to perform loop-closing with. Rule number 2 is similar to the rule number 1, but applicable to objects instead of rooms. It links obstacles to rooms, so it is used when the robot perceives new obstacles. Rule 3 is used for loop closing, when the robot realizes that two rooms previously modeled as disconnected are actually adjacent. In this case, a new door is included in the model without including a new room. Rule 4 is also used for loop closing. It describes how the event of finding out that two rooms that were thought to be different are actually the same would affect the graph. This happens when the robot closes a loop without recognizing that the new room it entered is already known. Both rooms collapse and all the ingoing and outgoing links are redirected from r_1 and r_2 to the new room r_3. The formal descriptions of the rules are shown in table I. The resulting P and B sets (containing rules and behaviors)

TABLE I
EXAMPLE RULE 0

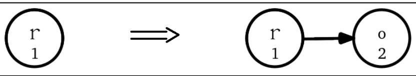
(a) Rule 0

	
Conditions:	
Operations:	$[r_2.active \leftarrow true]$
Behavior:	"explore room".

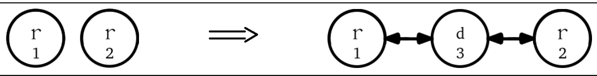
(b) Rule 1

	
Conditions:	$[r_1.active = true]$
Operations:	$[r_1.active \leftarrow false]$; $[r_3.active \leftarrow true]$
Behavior:	"explore room".

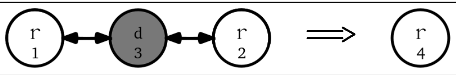
(c) Rule 2

	
Conditions:	$[r_1.active = true]$
Operations:	
Behavior:	"find new room".

(d) Rule3

	
Conditions:	$[r_1.active = true]$
Operations:	
Behavior:	"find new room".

(e) Rule4

	
Conditions:	$[r_1.active = true]$
Operations:	$[r_1.active \leftarrow false]$; $[r_4.active \leftarrow true]$; $[\forall a \in (A_{r_1} \cup A_{r_2})\ A_{r_4} \leftarrow A_{r_4} \cup a]$
Behavior:	"explore room".

are shown in equations 4 and 5. The PB mapping associating a behavior to each of the rules, is shown in equation 6.

$$P = \{Rule0, Rule1, Rule2, Rule3, Rule4\} \tag{4}$$

$$B = \{"explore\ room", "find\ new\ room"\} \tag{5}$$

$$PB = \begin{cases} Rule0 & \Longrightarrow & "explore\ room" \\ Rule1 & \Longrightarrow & "explore\ room" \\ Rule2 & \Longrightarrow & "find\ new\ room" \\ Rule3 & \Longrightarrow & "find\ new\ room" \\ Rule4 & \Longrightarrow & "explore\ room" \end{cases} \tag{6}$$

B. Implications of the use of CGGs

In order to implement the described grammar in a robot it is necessary to have a detector for each of the elements the model will be composed of (e.g., rooms, obstacles). The role of the bottom-up part is to detect atomic world elements and provide them to higher perceptive levels. Top-down might influence or force the detection of parts of the environment. In the case of the previously described grammar, this happens when the robot finds a door on a single room model or there are no rooms to close the loop with: since, in those cases, rule 1 is the only compatible rule introducing a door in the model, the only interpretation is that at the other side of the door there is a new room not previously seen. The grammar-based restrictions imposed by the available productions make the robot ignore impossible or useless signals. For example, the grammar disables the robot to close the loop with an adjacent room in the way that rule 4 does. Another example is that, when the system is started, obstacles, doors and other adjacent rooms are automatically ignored until the robot models the room in which it is located. This situation would be harder to specify and understand using regular programming languages, and would increase the probability of errors.

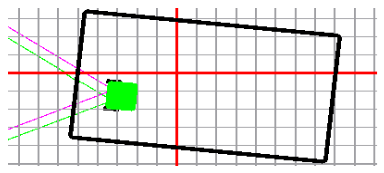


Fig. 3. World model after the robot entered and modeled the initial room.

CGGs provide a means to select the most appropriate robot behavior. In the situation of the last example, the world model would be composed of a single node with the start symbol (S) on it. In this scenario, the only applicable rule is the one that substitutes S with a room node (rule number 0). Thus, the only acceptable behavior would be "explore room". Once the room is detected, the robot will not be able to trigger rule 0 anymore. CGGs enable robots to compute the set of potential actions compliant with the model and the rules. However, since the set might contain several choices, CGGs can not always decide what the robot should do. It is possible that the robot selects a non-achievable behavior, for example, if it tries to find a new room when there are no more rooms. Moreover, in more complex scenarios there may be different potentially applicable rules (potentially interesting actions). These two situations make necessary an additional planning or homeostatic algorithm in order to reach a final decision.

C. Experimental Results

The grammar described in section IV-A has been implemented in an autonomous robot. Figure 2 shows an overhead view of the environment. The experiment starts with a world model containing only the start symbol. According to the behavior selector, the robot adopts the "explore room" behavior, expecting to run rule 0. Figure 3 shows the metric reconstruction of the map after the behavior models the room. Once the robot accomplishes the task, the robot could potentially apply rules 1 and 2. Thus, it interleaves their associated behaviors in order to discover the next room. By repeating this process (computing the candidate behaviors and interleaving them) the robot gets to the point in which all rooms and obstacles are modeled. This is shown in figure 4. Figure 5 shows the final graph model. As it can be depicted

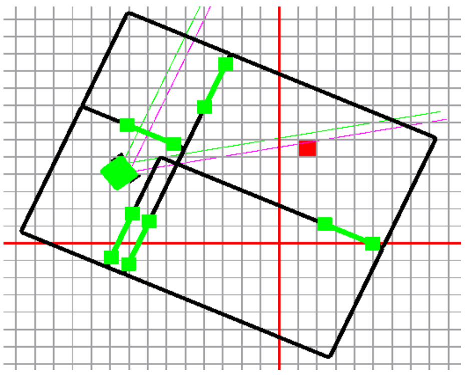


Fig. 4. Metric reconstruction from the topological model after the robot visited all the rooms.

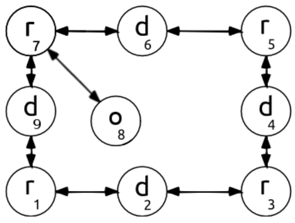


Fig. 5. Structure of the graph after the robot visited all the rooms.

in figure 4, the metric reconstruction shows overlapping errors caused by odometry and sensor uncertainty. These errors are canceled by performing a stochastic gradient descent search in a parameter space defined by the size of the rooms and the positions of the doors (see [17]). The minimization is weighted by the uncertainty of the models, as in standard non-linear graph optimization. This improves the overall result and allows the robot to perform robust loop-closings. The final metric reconstruction is shown in figure 6.

V. Conclusions

This paper presented the concept of cognitive graph grammars, the first grammar-based approach designed for active perception. CGGs provide a means for linking perception and action in order to gather the necessary information robots might need in a robust and context-aware way. At the same time they have interesting top-down properties and perceptive restrictions that help avoiding perception errors (both false negatives and positives). Section IV-C presented a proof of concept experiment, proving CGGs to be a useful and promising approach. We are currently experimenting with more complex grammars and studying methods to automatically infer formal knowledge from grammars. Efforts towards creating a tool in order to automatically translate CGG specifications into regular programming languages using a Domain Specific Language for CGGs are also being made.

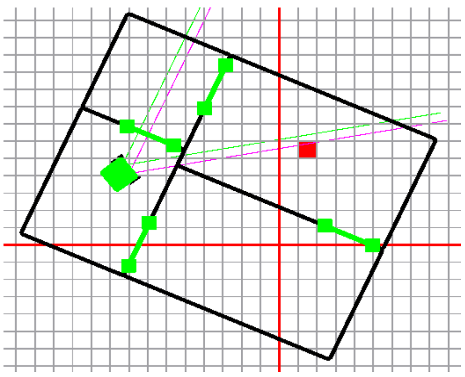


Fig. 6. Final metric reconstruction, after the optimization process.

Acknowledgment

This work has been supported by project IPT430000-2010-2 funded by Spanish Ministry of Science and Innovation, and by project IB10062 of the Extremaduran Government.

References

[1] E. Gat, “On three-layer architectures”, in *Artificial Intelligence and Mobile Robots*, p. 195–210, 1998.

[2] M.N. Nicolescu and M.J. Matarić, “A hierarchical architecture for behavior-based robots”, in *Proc. of Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems*, pp. 227–233, 2002.

[3] L.J. Manso, P. Bachiller, P. Bustos, P. Nunez, R. Cintas, and L. Calderita, “RoboComp: a tool-based robotics framework”, in *Proc. of Conf. on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR)*, pp. 251–262, 2010.

[4] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, “ROS: an open-source robot operating system”, in *Proc. of ICRA Workshop on Open Source Software*, 2009.

[5] R. Heckel, “Graph transformation in a nutshell”, *Electronic Notes in Theoretical Computer Science*, vol. 148, no. 1, pp. 187–198, 2006.

[6] J. Bauer and R. Wilhelm, “Abstract Interpretation of Graph Grammars”, in *Simulation and Verification of Dynamic Systems*, 2006.

[7] I. Bousassida, C. Chassor, and M. Jmaiel, “Graph grammar-based transformation for context-aware architectures supporting group communication”, *Nouvelles Technologies de l’Information*, vol. L, no. 19, pp. 29–42, 2010.

[8] B. Smith, A. Howard, J.M. McNew, J. Wang, and M. Egerstedt, “Multi-robot deployment and coordination with embedded graph grammars”, *Journal of Autonomous Robots*, vol. 26, no. 1, pp. 77–98, 2009.

[9] S.C. Zhu, R. Zhang, and Z. Tu, “Integrating bottom-up/top-down for object recognition by data driven markov chain monte carlo”, in *Proc. of Conf. in Computer Vision and Pattern Recognition*, vol. 1, pp. 738–745, 2000.

[10] Z. Tu, X. Chen, A.L. Yuille, and S.C. Zhu, “Image parsing: Unifying segmentation, detection and recognition”, *Journal of Computer Vision*, vol. 63, no. 2, pp. 113–140, 2005.

[11] F. Han and S.C. Zhu, “Bottom-up/top-down image parsing with attribute grammar”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 1, pp. 59-73, 2009.

[12] L. Lin, T. Wu, J. Porway, and Z. Xu, “A stochastic graph grammar for compositional object representation and recognition”, *Journal of Pattern Recognition*, vol. 42, no. 7, pp. 1297–1307, 2009.

[13] J.M. Hasemann, “A robot control architecture based on graph grammars and fuzzy logic”, in *Proc. of Conf. on Intelligent Robots and Systems*, vol. 3, pp. 2123–2130, 1994.

[14] J.M. Siskind, J.J. Sherman, I. Pollak, M.P. Harper, and C.A. Bouman, “Spatial random tree grammars for modeling hierarchal structure in images with regions of arbitrary shape”, *Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1504–1519, 2007.

[15] A. Torralba, K.P. Murphy, and W.T. Freeman, “Using the forest to see the trees: exploiting context for visual object detection and localization”, *Communications of the ACM*, vol. 53, no. 3, pp. 107–114, 2010.

[16] H. Svensson, A. Morse, and T. Ziemke, “Neural pathways of embodied simulation”, in *Anticipatory Behavior in Adaptive Learning Systems*, pp. 95–114, 2009.

[17] P. Bachiller, P. Bustos, and L.J. Manso, “Attentional Behaviors for Environment Modeling by a Mobile Robot”, in *Stereo Vision*, pp. 17–40, InTech, 2011.

Experimental Validation of a PCA-Based Localization System for Mobile Robots in Unstructured Environments

F. Carreira, C. Christo, D. Valério, M. Ramalho, C. Cardeira, J. M. F. Calado, and P. Oliveira

Abstract— **In this paper a new PCA-based positioning sensor and localization system for mobile robots to operate in unstructured environments (e.g. industry, services, domestic, . . .) is proposed and experimentally validated. The positioning system resorts to principal component analysis (PCA) of images acquired by a video camera installed onboard, looking upwards to the ceiling. This solution has the advantage that the need of selecting and extracting features is avoided. The principal components of the acquired images are compared with previously registered images, present in a reduced onboard image database and the position measured is fused with odometry data. The optimal estimates of position and slippage are provided by a Kalman filter, with global stable error dynamics. The experimental validation reported in this work focus on the results of a set of exhaustive experiments carried out in a real environment, where the robot travels along straight lines. A small position error estimate was always observed, for arbitrarily long experiments, and slippage was estimated accurately in real time.**

I. INTRODUCTION

The problem of localization has been a great challenge to the scientific community in the area of mobile robotics; see [6], [3] and the references therein. As happens with persons or animals, for a robot to navigate from a point to another it is of great importance its ability to look at the environment and rapidly answer the following questions: where am I? and what am I facing?

SLAM (Simultaneous Localization And Mapping) is a process by which a mobile robot can build a map of an environment and at the same time to use this map to estimate its localization. In SLAM, both the trajectory of the platform and the localization of all landmarks are estimated online without the need for any a priori knowledge of localization [6], [19]. However, substantial issues remain to be solved in practice. One of the issues that remain open is that of solutions relying on landmarks or on any other features that the robot may sense in the environment, and will subsequently be used for robot localization. In practice, given one environment, there is no guarantee that the same features will be present in the environment on subsequent visits of the robot to the same localization (loop closure problem). For instance, fast corners [24] are a very efficient way to detect features in an image but the number of corners actually found may depend on many tuning parameters and different corners may appear in different images taken from the same localization at different times. Random Sample Consensus (RANSAC) is considered the state of the art technique to keep track of features while disregarding outliers but in practice all these strategies rely on some structure of the environment [2], [16], [7].

This paper follows an alternative approach resorting to Principal Component Analysis (PCA) that actually does not depends on any predefined structure of the environment. Of course, there should always be something to distinguish data acquired in one location to data acquired in another location but no previous assumptions on the predefined structure of the environment needs to be considered. The PCA data analysis corresponds to the computation of the data orthogonal components that will make each dataset different. Hence, the localization is defined based on the PCA of the large amount of data taken from the unstructured environment. Experimental results in 1D are shown, proving the efficacy of the approach.

A. Current Practices

The use of vision systems for robot localization is very common [22], [21] due to the ability to obtain information about the environment. Many vision systems compute the robot pose (position and attitude) from features of the environment, either from the entire image [11], extracting lines [15], simply getting points of interest [12], [10], or extracting scale-invariant features [17]. The computational complexity of such algorithms to obtain features is not negligible: thus the implementation in real-time systems still demands the search for other approaches of reduced complexity.

Very successful implementations of visual odometry are presented in [21], where a robot was able to localize itself outdoors based on a minimum number of singular points that have to be present in the environment. Although many robots use cameras to look around itself to get its global pose in the environment [23], [10], [14], others use a single camera looking upward [12], [8], [25]. The use of vision from the ceiling has the advantage that images can be considered without scaling, i.e. a 2D image problem results and will be pursued in this work.

B. PCA-based localization and optimal estimation

Since feature based techniques are computationally heavy, some researchers have been working to find methods to make this process more efficient. To achieve reduced complexity algorithms, the use of PCA in mobile robots for self-localization has been explored [14], [18], [1]. However, all these approaches use front or omnidirectional cameras, causing the algorithms to address problems of occlusion or

C. Christo, D. Valério, M. Ramalho and C. Cardeira are with IDMEC / Instituto Superior Técnico, Technical University of Lisbon, Av. Rovisco Pais 1, 1049-001 Lisboa Portugal, cchristo@dem.ist.utl.pt, {duarte.valerio,mramalho,carlos.cardeira}@ist.utl.pt

F. Carreira and J. Calado are with IDMEC / Instituto Superior Técnico, Technical University of Lisbon and with Instituto Superior de Engenharia de Lisboa / IPL, R. Conselheiro Emídio Navarro 1, 1959-007 Lisboa, {fcarreira,jcalado}@dem.isel.ipl.pt

P. Oliveira is with ISR / Instituto Superior Técnico and IDMEC / Instituto Superior Técnico, Technical University of Lisbon, address as above, p.oliveira@dem.ist.utl.pt

comparison with images in different planes. In [20], PCA was used for terrain reference navigation of underwater vehicles. The PCA-based localization system that we present is this work corresponds to a experimental validation of the one proposed in [20], using a Dubins Car equipped with a video camera looking upwards to the ceiling.

Beyond the problems of image processing for self-localization, another challenge is to deal with the fusion of the PCA-based position with the odometry data that is given by the robot kinematics. Mobile robot kinematics (e.g. Dubins car) are in general non linear. This fact prevents the direct use of a Kalman Filter, which is a linear optimal estimator. To tackle this problem, many localization systems use the Extended Kalman Filter (EKF) with well characterized optimality and stability limitations. Even though it can give a reasonable performance, the EKF may diverge in consequence of wrong linearisation or sensor noise.

For the purpose of this paper, the Dubins Car model is restricted to one-dimensional movement, thus avoiding the non-linear model issues mentioned above. Moreover, the filter also estimates the slippage that is eventually present in the reality. Many researchers tend to neglect slippage: our approach addresses the problem explicitly. As slippage is inevitable, we append a state to our model to express the slippage explicitly. The filter estimates both slippage and robot localization. Furthermore, the optimal estimate is achieved, under the assumption that disturbance noise can be modelled by Gaussian distributions, with global stable error dynamics can be obtained (see [20], where however no experimental results are given). Further work will be carried out in the near future to deal with 2D operation of the Dubins car resorting to the recent results that can be found in [4].

C. Advantages and drawbacks

The proposed PCA-based position sensor and localization estimation has the following advantages:

- The robot is able to self-locate in an indoor environment, only with onboard sensors (no external sensors or landmarks are required);
- The algorithm is fast, thus it consumes very few computational resources;
- The database of images stored onboard the mobile robot is of reduced size, when compared with the total number of images considered;
- The memory to allocate for the database storage is flexible and related with the required positioning error accuracy;
- No hypothesis is made about specific features in the environment: thus this system can operate in an unstructured environment where the only requirement is that images must be different in each location;
- Under Gaussian assumption for the disturbances, the localization system estimates in real time the position and slippage with global stable error dynamics.

Some of the limitations for the proposed approach include:

- The robots should work in buildings with ceilings where rich information can be found (e.g. building-related systems such as HVAC, electrical and security systems, etc.);
- The ceilings should be static: the system cannot be used outdoors as the sky is far from static and changes randomly;
- The system is formulated in a digital discretised version as well as the PCA approach pursued.

A general limitation of all vision-based systems is their sensitivity relative to lighting conditions.

This paper is organized as follows: in section II, the principal component analysis technique is introduced in detail. In section III, the mobile robot kinematics model is presented and section IV a set of experimental results are reported to validate and assess the performance of the proposed PCA-based positioning sensor and localization system, resorting to a Kalman filter. Conclusions and future work are presented in section V.

II. PRINCIPAL COMPONENT ANALYSIS

In this section the fundamentals of the positioning system proposed in this work will be introduced. The proposed methodology resorts to optimal signal processing techniques, namely PCA, based on the Karhunen-Loève (KL) transform to obtain a nonlinear positioning sensor. Considering all linear transformations, PCA allows for the optimal approximation to a stochastic signal in the least squares sense. Furthermore, it is a well known signal expansion technique with uncorrelated coefficients for dimensionality reduction. These features make the KL transform interesting for many signal processing applications such as data compression, image and voice processing, data mining, exploratory data analysis, pattern recognition and time series prediction. For a thorough introduction to this topic and a number of state of the art applications see [13].

Consider a set of M stochastic signals $\mathbf{x}_i \in \mathbb{R}^N$, $i = 1, \ldots, M$, each corresponding to the stacked version of an image acquired with the video camera installed onboard the mobile robot and represented as a column vector with mean $\mathbf{m}_x = \frac{1}{M}\sum_{i=1}^{M}\mathbf{x}_i$. The purpose of the KL transform is to find an orthogonal basis to decompose a stochastic signal $\mathbf{x}$, from the same original space, to be computed as $\mathbf{x} = \mathbf{U}\mathbf{v} + \mathbf{m}_x$, where vector $\mathbf{v} \in \mathbb{R}^N$ is the projection of $\mathbf{x}$ in the basis, i.e. $\mathbf{v} = \mathbf{U}^T(\mathbf{x} - \mathbf{m}_x)$. Matrix $\mathbf{U} = [\mathbf{u}_1\,\mathbf{u}_2 \ldots \mathbf{u}_N]$ should be composed by the N orthogonal column vectors of the basis, verifying the eigenvalue problem

$$\mathbf{R}_{xx}\mathbf{u}_j = \lambda_j \mathbf{u}_j,\ \ j = 1, ..., N, \qquad (1)$$

where $\mathbf{R}_{xx}$ is the covariance matrix, computed from the set of M experiments using

$$\mathbf{R}_{xx} = \frac{1}{M-1}\sum_{i=1}^{M}(\mathbf{x}_i - \mathbf{m}_x)(\mathbf{x}_i - \mathbf{m}_x)^T. \qquad (2)$$

Assuming that the eigenvalues are ordered, i.e. $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$, the choice of the first $n \ll N$ principal components leads to an approximation to the stochastic signals given by the ratio on the covariances associated with the components, i.e. $\sum_n \lambda_n / \sum_N \lambda_N$. In many applications, where stochastic multidimensional signals are the key to overcome the problem at hand, this approximation can constitute a large dimensional reduction and thus a computational complexity reduction.

The advantages of PCA are threefold: i) it is an optimal (in terms of mean squared error) linear scheme for compressing a set of high dimensional vectors into a set of lower dimensional vectors; ii) the model parameters can be computed directly from the data (by diagonalising the ensemble covariance); and iii) given the model parameters, projection into and from the bases are computationally inexpensive operations, $\sim \mathcal{O}(nN)$. These advantages suit our problem especially, as the computation power, energy and data storage onboard should be kept as reduced as possible to augment the operation interval and reduce the cost of the systems onboard.

Assume that scenario in the area of indoor mobile robotics (e.g. industrial automation or robotic office applications), where a navigation system to be installed on one or more mobile robots must be developed and operated. In this scenario it is considered that there is data available allowing to develop a positioning system that recognizes the actual position of the robot in real time. The steps to implement a PCA-based positioning sensor using this visual data will be outlined next.

Prior to the deployment of the robots, the visual data of the area under consideration should be partitioned in *mosaics* with fixed dimensions N_x by N_y. After reorganizing this two-dimensional data in vector form, e.g. stacking the columns, a set of M stochastic signals $\mathbf{x}_i \in \mathbb{R}^N$, $N = N_x N_y$ results. The number of signals M to be considered depends on the mission scenario and on mosaic overlapping. The KL transform can be computed, using (1)–(2); the eigenvalues must be ordered; and the number n of the principal components to be used should be selected, according with the required level of approximation.

The following data should be recorded for later use:

1) the data ensemble mean $\mathbf{m}_x$;
2) the matrix transformation with n eigenvectors
$$\mathbf{U}_n = [\mathbf{u}_1 \ \dots \ \mathbf{u}_n]; \tag{3}$$
3) the projection on the selected basis of all the mosaics, computed using
$$\mathbf{v}_i = \mathbf{U}_n^T(\mathbf{x}_i - \mathbf{m}_x), \ i = 1, \dots, M; \tag{4}$$
4) the coordinates of the center of the mosaics
$$(x_i, y_i), \ \ i = 1, \dots, M. \tag{5}$$

During the mission, at the time instants $t_k = Lk$ (where L is a positive integer), the acquired images will constitute the input signal $\mathbf{x}$ to the PCA positioning system. The following tasks should be performed:

a) compute the projection of the signal $\mathbf{x}$ into the basis, using
$$\mathbf{v} = \mathbf{U}_n^T(\mathbf{x} - \mathbf{m}_x); \tag{6}$$
b) given an estimate of the current horizontal coordinates of the robot position $\hat{x}$ and $\hat{y}$, provided by the navigation system, search on a given neighborhood δ the mosaic that verifies
$$\forall_i \|[\hat{x}\ \hat{y}]^T - [x_i\ y_i]^T\|_2 < \delta, \ \ r_{\text{PCA}} = \min_i \ \ \|\mathbf{v} - \mathbf{v}_i\|_2; \tag{7}$$
c) given the mosaic i which is closest to the present input, its center coordinates (x_i, y_i) will be selected as the x_m and y_m measurements.

The relation $\mathbf{f}$ between r_{PCA} and the positioning sensor error covariance $\mathbf{R}$ (observation noise) to be used in the $\mathcal{H}_2$ estimation problem
$$\mathbf{R} = \mathbf{f}\ r_{\text{PCA}} \tag{8}$$
will be chosen according to the chosen environment. Note that the image-based PCA positioning system described above can be straightforwardly extended to incorporate data from other sensors installed onboard mobile robots such as magnetometers and range information from time-of-flight cameras or structured-light 3D scanners (e.g. Microsoft Kinect).

III. MODEL

The experimental validation of the proposed positioning system was performed resorting to a low cost mobile robotic platform [5], with the configuration of a Dubins car. This platform has a PC laptop that controls the motors through a closed loop motor controller connected by a USB and has a webcam pointing upwards to the ceiling (see figure 1). The low replication cost for these platforms will be instrumental during the future tasks envisioned relying on cooperation and multi-agent systems (mentioned among future work in section V).

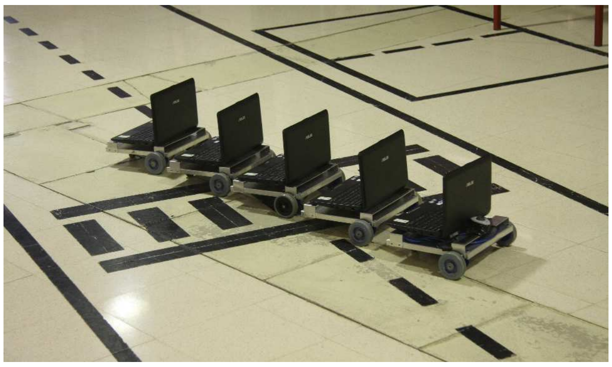


Fig. 1. Mobile robot platforms used for experimental validation

The mobile robot kinematic model that describes the movement in a straight line (1D) is
$$\dot{x} = u + b + \mu_1 \tag{9}$$
$$\dot{b} = 0 + \mu_2 \tag{10}$$
considering the following assumptions:

- the slippage velocity is constant or slowly varying (i.e. $\dot{b} = 0$);
- the noise in the actuation (motors are in closed loop) and the slippage velocity are assumed as zero-mean uncorrelated white Gaussian noise, $\mu_i \sim N(0, \sigma_i^2)$.

Expressing the model dynamics in a state-space system with $\mathbf{x} = [x\ b]^T$,
$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \tag{11}$$
$$y = [1\ 0]\,\mathbf{x} + \gamma \tag{12}$$

The output of this system y is the positioning sensor measurement described in the previous section. Since the position estimator is processed in a digital processor, the discrete model is obtained assuming that the vehicle velocity u is constant (zero order hold assumption) between two consecutive processing times, resulting

$$\mathbf{x}(k+1) = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} T \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} T & T^2/2 \\ 0 & T \end{bmatrix} \mu(k) \tag{13}$$

$$y(k) = [1\ 0]\,\mathbf{x}(k) + \gamma(k) \tag{14}$$

The design of a linear time-invariant Kalman filter for the underlying model described above is by now classic and the reader is referred to [9].

IV. EXPERIMENTAL RESULTS

The mobile robot self-localization methodology proposed in this work is tested for the aforementioned mobile robot travelling along a 3 m length straight line. Ceiling images are captured with a constant distance and referenced, allowing for the creation of the PCA eigenspace (the image database referred in the previous sections of the paper) to capture the principal components of the environment. To create the eigenspace, gray scale images with 320 by240 pixels are subsampled (1 : 25) and transformed into vectors, $\mathbf{x}_i \in \mathbb{R}^N$, $i = 1, \ldots, M$, where M stands for the number of images and N stands for the number of pixels of each image. (Notice that since this is a 1D experiment only one coordinate is necessary, along the direction of movement.)

The covariances to be used in the Kalman Filter design were considered as constant and were obtained considering $\mathbf{Q} = \mathbf{Q}(k)$ and $\mathbf{R} = \mathbf{R}(k)$ as the covariance error in the actuation and the pose estimator, respectively. The value of $\mathbf{Q} = 4.1 \times 10^{-6}$ m^2 was obtained measuring the covariance error of the robot motion along one predefined path. The value of $\mathbf{R} = 6.8 \times 10^{-3}$ m^2 was obtained measuring the covariance error of the pose estimator (position given by the PCA positioning sensor) when the robot moves along one path with images in the eigenspace. This process and sensor noises lead to a Kalman filter gain $\mathbf{K} = [0.0429\ 0.0188]^T$.

To study the PCA positioning sensor performance, 31 ceiling images (with a distance of 0.1 m) were captured with the mobile robot travelling with a constant velocity of 0.125 m/s along the straight line, as mentioned above. The images have been subsampled with a step of 5 pixels in width and height to reduce the amount of processing data (1 : 25). Analysing the eigenvalues and selecting components that explain the variability of the images in an excess of 80%, results on an eigenspace (image database) of 4 eigenvectors.

A. *Monte Carlo Performance Tests*

To assess the mobile robot self-localization methodology proposed, a Monte Carlo test composed of 10 experiments as described above has been repeated. Images were captured at 20 Hz and the PCA-based positioning sensor was acquired; figure 2 gives the localization results obtained in one of those experiments. The results show that the PCA algorithm provides a good approximation to the real robot localization. However, some discontinuities in the acquired robot position are observed. Anyway, the deviations observed in instants 6 s, 13 s and 22 s are due to disturbances. It is important to remark that the results from the Kalman filter smooth out the position errors present in the PCA-based positioning sensor. The estimated errors for 5 experiments are depicted in figure 3.

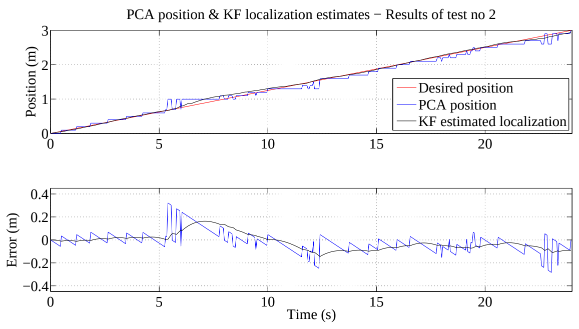


Fig. 2. Results of PCA-based positioning sensor and localization estimates from Kalman filter

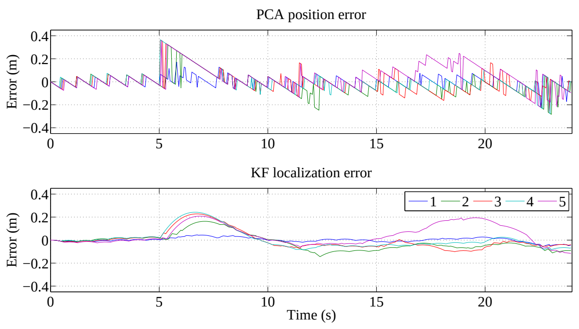


Fig. 3. Localization errors of tests along a straight line

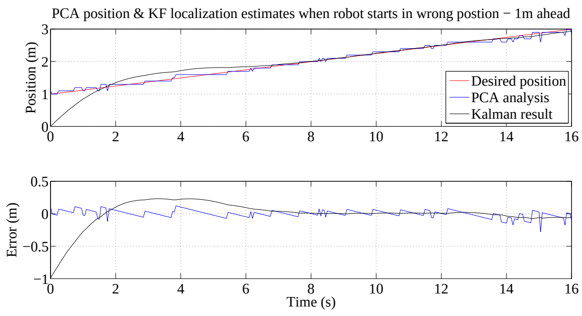


Fig. 4. Results of positioning system when the robot starts 1 m ahead of the usual position

B. *Stability Validation*

A second test was performed to assess the positioning system global stability when the initial position coordinates do not match the robot real initial position. Thus it is possible to check that the estimator is able to correct the initial position error, as predicted by the stability properties of the Kalman filter. In this case, the robot was placed 1 m ahead of the usual initial position. An Extended Kalman filter could easily diverge under such experimental conditions. The eigenspace was again created with a distance between acquire images of 0.1 m (same 31 images as in the previous set of tests) and the results show that the positioning system needs less than 1.5 s to provide an accurate estimate of the mobile robot localization. Considering that the robot moves at a constant velocity of 1.5 m/s, the positioning system is able to identify the mobile robot real position at the same time that the second image is captured to the eigenspace (figure 4).

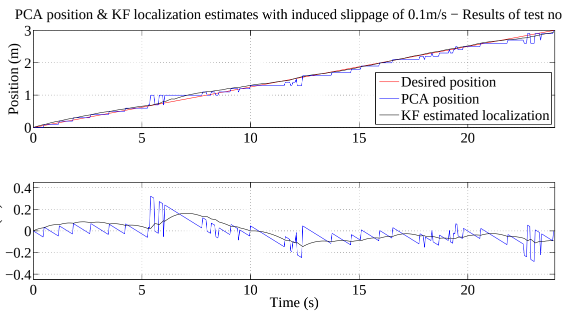


Fig. 5. Results of the positioning system when the robot moves with a slip velocity of 0.1 m/s

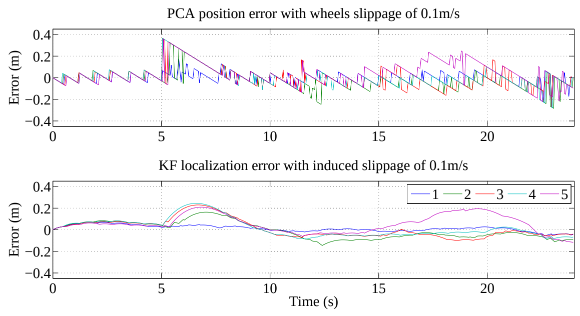


Fig. 6. Error of positioning system when the robot moves with a slip velocity of 0.1 m/s

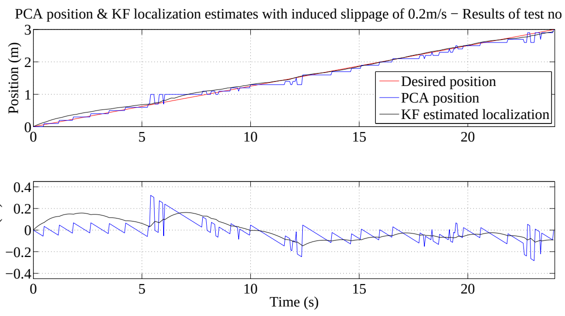


Fig. 7. Results of the positioning system when the robot moves with a slip velocity of 0.2 m/s

C. Real-time Slippage Estimation

As a further assessment of the localization system performance, a set of tests have been conducted considering that the mobile robot experiences a constant, artificially imposed, wheel slippage. Two tests are reported considering that the mobile robot travels with a slippage in the wheels, that leads to a constant velocity below 0.1 m/s and 0.2 m/s, respectively in figures 5 and 7, relative to the commanded velocity. The estimation errors are depicted respectively in figures 6 and 8. Results show that the localization system is able to accurately estimate the mobile robot real position in all situations. The Kalman filter estimates present initial higher errors for higher values of slippage (above 0.2 m/s). After a transient of about 5 s (see figure 9), the localization system is able to estimate and correct the wheels slippage in real-time and the results obtained in the remaining of the experiments have similar performance as the ones obtained in the experiments without slippage.

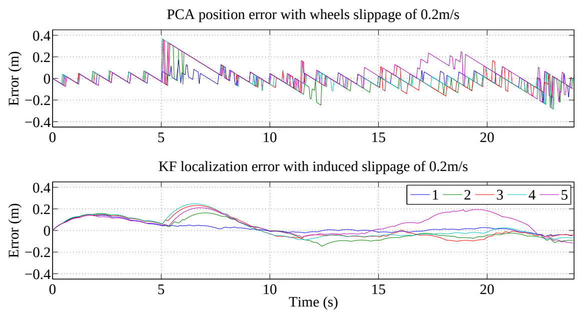


Fig. 8. Error of positioning system when the robot moves with a slip velocity of 0.2 m/s

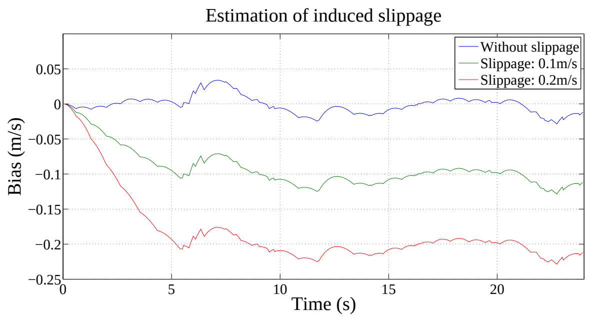


Fig. 9. Results of bias in Kalman Filter for different wheels slippery velocity

D. Preliminary PCA Performance Assessment

PCA has a number of parameters that must be selected prior to the deployment of the positioning and localization system. A trade-off will always be found relating the number of images in the database (eigenspace size) and the accuracy of the positioning sensor proposed. A preliminary study on the impact of changing these parameters will be reported in this section. The results from a set of tests where the image acquisition step varies in the interval $[0.05\ 0.4]$ m, i.e. using between 61 and 8 images, respectively, were performed creating different eigenspaces. Hence, the mobile robot positioning system performance has been tested considering an increase between the eigenspace points used (Table I).

Results show that the PCA positioning system with Kalman Filter were able to identify the correct mobile robot position based on ceiling captured images, even when the distance between knowledge points is increased, reducing the number of images in the eigenspace (figure 10). For a

TABLE I

PCA POSITIONING SENSOR AND LOCALIZATION SYSTEM WITH DIFFERENT IMAGE ACQUISITION STEPS

Distance between images (m)	Sample time (s)	Number of images in PCA	PCA localization $\bar{\sigma}^2$ (m^2)	PCA with a Kalman Filter $\bar{\sigma}^2$ (m^2)
0.05	0.4	61	0.00545	0.00380
0.1	0.8	31	0.00683	0.00436
0.2	1.6	16	0.01063	0.00525
0.3	2.4	11	0.01360	0.00341
0.4	3.2	8	0.06428	0.03844

distance between frames up to 0.3 m, results show that the position error is small, not exceeding 0.15 m. For longer distances between frames, e.g. 0.4 m, the position estimate accuracy degrades gracefully. However, even in this case, the error is below 0.4 m, which allows to conclude that the error is less than the granularity associated with the image acquisition intervals.

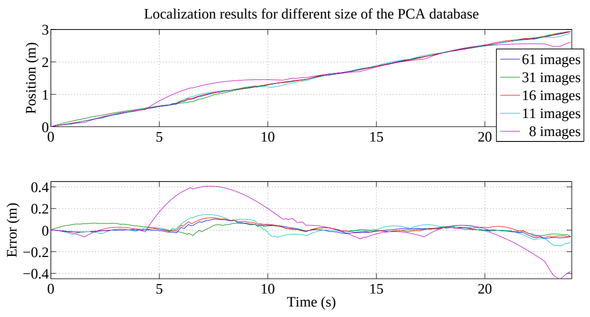


Fig. 10. Results of PCA together with a Kalman Filter

V. CONCLUSIONS

A new positioning sensor and a localization system for mobile robots to operate in unstructured environments is proposed and experimentally validated along a straight line (1D). The positioning sensor resorts to PCA, from the images acquired by a video camera installed onboard, looking upwards to the ceiling. Several tests were performed namely: i) Monte Carlo performance study, ii) global stability validation, iii) real-time slippage estimation, and iv) PCA performance assessment. All tests were successful and allow to conclude that the proposed approach can be useful in a number of mobile robotic applications.

This paper represents the initial step towards a multi-agent system based architecture where a large set of mobile robots will be able to cooperate to perform navigation and formation tasks, featuring obstacle avoidance, human interaction and search and rescue activities. For that purpose, the next step taken was to consider the robots in 2D. Currently, the theoretical part of 2D version has been developed, resorting to a set of recent results reported in [4], and will be subject to intensive validation tests in the near future.

ACKNOWLEDGMENTS

This work was partially supported by Fundação para a Ciência e a Tecnologia, through IDMEC under LAETA.

REFERENCES

[1] Matej Artač, Matjaž Jogan, and Aleš Leonardis. Mobile robot localization using an incremental eigenspace model. In *IEEE International Conference on Robotics and Automation*, 2002.
[2] B. Bacca, J. Salvi, and X. Cufí. Appearance-based mapping and localization for mobile robots using a feature stability histogram. *Robotics and Autonomous Systems*, 59(10):840–857, 2011.
[3] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (SLAM): part II. *Robotics Automation Magazine, IEEE*, 13(3):108 – 117, sept. 2006.
[4] Pedro Batista, Carlos Silvestre, and Paulo Oliveira. Optimal position and velocity navigation filters for autonomous vehicles. *Automatica*, 46(4):767–774, 2010.
[5] C. Cardeira and J. Sá da Costa. A low cost mobile robot for engineering education. In *Industrial Electronics Society, 2005. IECON 2005. 31st Annual Conference of IEEE*, pages 2162–2167, Raleigh, 2005.
[6] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part I. *Robotics Automation Magazine, IEEE*, 13(2):99 – 110, june 2006.
[7] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.
[8] Yasuaki Fukutani, Tomoyuki Takahashi, Masahiro Iwahashi, Tetsuya Kimura, Samsudin Siti Salbiah, and Norrima Binti Mokhtar. Robot vision network based on ceiling map sharing. In *IEEE International Workshop on Advanced Motion Control*, pages 164–169, 2010.
[9] A. Gelb. *Applied optimal estimation*. MIT Press, 1974.
[10] Arturo Gil, Oscar Martinez Mozos, Monica Ballesta, and Oscar Reinoso. A comparative evaluation of interest point detectors and local descriptors for visual SLAM. *Machine Vision and Applications*, 21(6):905–920, 2010.
[11] Marius Hofmeister, Maria Liebsch, and Andreas Zell. Visual self-localization for small mobile robots with weighted gradient orientation histograms. In *International Symposium on Robotics*, pages 87–91, 2009.
[12] WooYeon Jeong and Kyoung Mu Lee. CV-SLAM: a new ceiling vision-based SLAM technique. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3195–3200, 2005.
[13] I. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 2002.
[14] Ben Kröse, Roland Bunschoten, Stephan Ten Hagen, Bas Terwijn, and Nikos Vlassis. Household robots look and learn: environment modeling and localization from an omnidirectional vision system. *IEEE Robotics & Automation Magazine*, 11:45–52, 2004.
[15] Bor-Woei Kuo, Hsun-Hao Chang, Yung-Chang Chen, and Shi-Yu Huang. A light-and-fast slam algorithm for robots in indoor environments using line segment map. *Journal of Robotics*, 2011:—, 2011.
[16] I. Loevsky and I. Shimshoni. Reliable and efficient landmark-based localization for mobile robots. *Robotics and Autonomous Systems*, 58(5):520–528, 2010.
[17] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
[18] S. Maeda, Y. Kuno, and Y. Shirai. Active navigation vision based on eigenspace analysis. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1018–1023, 1997.
[19] Michael Montemerlo, Sebastian Thrun, Daphne Roller, and Ben Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *International Joint Conference on Artificial Intelligence*, 2003.
[20] Paulo Oliveira. MMAE terrain reference navigation for underwater vehicles using PCA. *International Journal of Control*, 80(7):1008–1017, July 2007.
[21] Davide Scaramuzza, Friedrich Fraundorfer, and Roland Siegwart. Real-time monocular visual odometry for on-road vehicles with 1-point ransac. In *IEEE-ICRA09 International Conference on Robotics and Automation, 2009.*, pages 4293 –4299, may 2009.
[22] Wolfram Burgard Sebastian Thrun and Dieter Fox. *Probabilistic Robotics*. Intelligent Robotics and Autonomous Agents. MIT Press, 2005.
[23] C. Siagian and L. Itti. Biologically inspired mobile robot vision localization. *IEEE Transactions on Robotics*, 25(4):861–873, 2009.
[24] Miroslav Trajković and Mark Hedley. Fast corner detection. *Image and Vision Computing*, 16(2):75–87, 1998.
[25] De Xu, Liwei Han, Min Tan, and You Fu Li. Ceiling-based visual positioning for an indoor mobile robot with monocular vision. *IEEE Transactions on Industrial Electronics*, 56(5):1617–1628, 2009.

Motion planning and localization approaches for mobile robot navigation in ITER

A. Vale*, F. Valente*, I. Ribeiro†, J. Ferreira* and R. Ventura†
* Instituto de Plasmas e Fusão Nuclear – Laboratório Associado
Instituto Superior Técnico – Universidade Técnica de Lisboa
† Laboratório de Robótica e Sistemas em Engenharia e Ciência – Laboratório Associado
Instituto Superior Técnico – Universidade Técnica de Lisboa

***Abstract*—The ITER (International Thermonuclear Experimental Reactor) aims to prove the viability of fusion power. During the maintenance, the transport operations has to be carried out by autonomous mobile robots. The very high weight of the loads to be transported, together with bullet-proof reliability requirements, make the deployment of such robots a challenging scientific and technological problem. The paper addresses the problems of motion planning and the localization of these robots. The motion planning is based on line guidance and free roaming approaches to optimize trajectories for a rhombic like vehicle. The localization system is based on a laser range finder network, where two methods for pose estimation are used (Extended Kalman Filtering and bootstrap Particle Filtering). Experimental results, both from simulation and from small prototype are presented, illustrating the described methods.**

I. Introduction

The demand for energy is a critical problem the human societies have to address in a near future. The problem rises from the fact that fossil fuels are finite resources and renewable energies alone will not be enough to meet the demand. In this context, the ITER (International Thermonuclear Experimental Reactor) project aims to prove the viability of fusion power as an alternative and safe energy source. ITER will be built in Cadarache, France.

The Tokamak Building (TB) of ITER (Fig. 1) is where the reactor will be installed. During nominal and maintenance operations, the human presence is forbidden due to the high levels of radiation, and therefore, remote handling (RH) systems will play an important role in the ITER project. In [1] and [2] there is a description of RH systems in ITER. One of such systems is the Cask and Plug Remote Handling System (CPRHS), a mobile vehicle responsible for RH operations of transportation of contaminated components and equipment between the TB and the Hot Cell Building (HCB). The largest CPRHS has dimensions 8.5m x 2.62m x 3.7m (length, width, height) and when fully loaded weights approximately 100T. The CPRHS is divided into three main components: the Cask, that contains the load, the Pallet, that supports the Cask and the Cask Transfer System (CTS). The CTS acts as a mobile robot, by driving the entire vehicle, or by moving independently from the other components. The CTS has a rhombic kinematic configuration, as described in [3] and depicted in Fig. 1. This configuration allows to control the velocity, V_i, and orientation, θ_i, of each wheel $i \in \{R, F\}$. Additionally it allows for both wheels to follow the same path, in this paper referred as line guidance, or for each wheel to follow a different path, referred as free roaming, thus providing an increased flexibility, when moving in the cluttered environments of the TB and HCB.

To perform the required RH operations, the vehicle must move along optimized trajectories and for that purpose a motion planning framework described in previous works [4], [5], is used. This paper introduces two novelties: (1) the requirement that all trajectories are generated in order to maximize the part of the path that is shared by all the trajectories, and (2) experimental results of the approach in a 1:25 scale model real robot with rhombic kinematics.

A problem that is also addressed, in this paper, is the localization of the vehicle, by using a network of laser sensors placed in the scenario, along the lines presented in [6]. For testing purposes of the localization framework, a prototype of the CPRHS was built.

The paper is organized as follows: Section II presents the motion planning methodologies, Section III introduces the localization methods, Section IV presents the obtained results and in Section VI the conclusions and open issues are discussed.

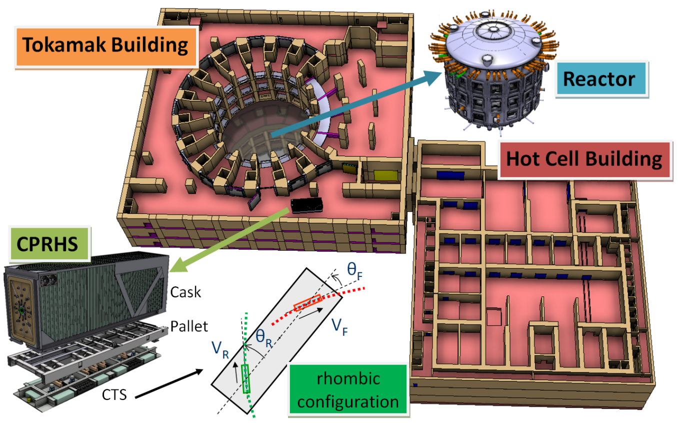


Fig. 1. Models of TB and HCB scenarios. Also displayed are detailed views of the reactor, of the CPRHS, and of the rhombic configuration.

II. Motion Planning

The vehicle is required to move along a path that simultaneously maximizes the clearance and minimizes the distance between the start and the goal poses (position and orientation). Two motion planning methodologies, line guidance and free roaming, were developed.

A. *Line guidance*

The line guidance motion planning requires that both wheels of the vehicle follow the same path and, if adopted in ITER, the CTS will act as an Automated Guided Vehicle (AGV). This methodology is achieved in three main steps, [4], shown in Fig. 2: (1) geometric path evaluation, (2) path optimization, and (3) trajectory evaluation.

(1) Given the start and goal points, the map (a 2D projection at floor level of the scenario's 3D model, consisting in a set of line segments that defines walls and other obstacles) is decomposed into a set of triangles, by using Constrained Delaunay Triangulation, [7], to account for all walls. Then, the algorithm finds all sets of sequence of triangles that contain and link the start and goal points. Each sequence of triangles is converted into a sequence of points (mid point of the common edge of two consecutive triangles) yielding a path, shown in top left of Fig. 2. The shortest path is chosen as the geometric path.

(2) The initial geometric path does not guarantee a collision free path for a rigid body, with dimensions, and the path is not smooth (top center of Fig. 2). The optimization phase is a trade off between two criteria: clearance from obstacles, by increasing the distance from the vehicle to walls, and path smoothness, that results in shorter and smoother paths. The optimization procedure uses the elastic band concept, [8], where the path is modelled as an elastic band, similar to a series of connected springs subject to two types of forces: internal and external forces. The first are the internal elastic forces, whose magnitude is proportional to the amplitude of displacement and determine that the path becomes shorter. The repulsive forces are responsible for keeping the path, and thus the vehicle, away from the obstacles.

(3) The final trajectory is obtained by defining the velocity of the vehicle at each point of the optimized path, shown in top right of Fig. 2. In order to reduce the risk of collision in the case of a major malfunction, the velocity is reduced once the distance to the nearest obstacle decreases below a threshold value.

There are particular situations where given the above approach it is not possible to obtain a feasible solution, as illustrated in Fig. 3 - Left, where a clash occurs. By considering maneuvers in the motion planning procedure, it is possible to overcome this problem in these particular situations. A maneuver exists when the vehicle stops and changes its motion direction, in order to achieve a specified orientation, as illustrated in Fig. 3 - Right. A maneuver requires splitting the path in two sub-paths with the constrain that the final pose of the first sub-path is the initial pose of the next sub-path. Multiple maneuvers can be considered, with the path optimization being applied to each sub path. The point(s) of maneuver are introduced manually and its position can be set to be fixed or adjusted during optimization.

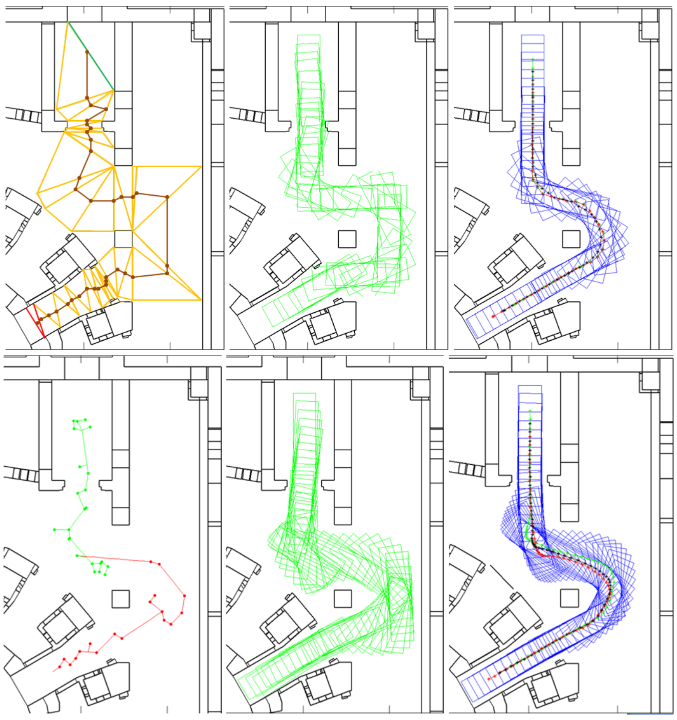


Fig. 2. Top (from left to right) - geometric path, poses over the geometric path and final optimized path shared by both wheels; Bottom (from left to right) - search for initial path by RRT, poses over the initial path and final optimized path with each wheel following its own path.

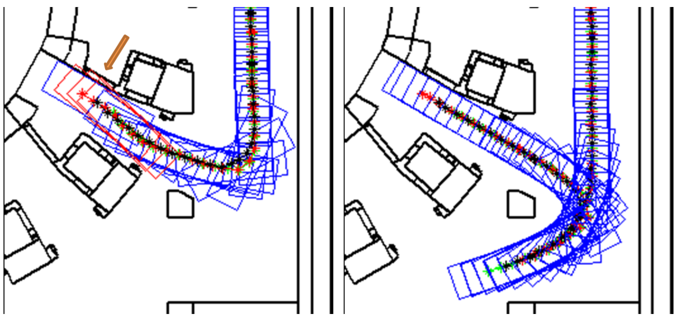


Fig. 3. Left - path shared by both wheels of the vehicle, with collision; Right - path with each wheel following its own path, without collision.

B. *Free roaming*

The free roaming motion planning does not constrain both wheels to the same path and each wheel can follow a different path. This methodology draws inspiration from the elastic band concept, [8], and was proposed in [5]. The vehicle's poses along the path act as rigid bodies, connected through internal interactions and subjected to external repulsive forces, resulting from the closest obstacles.

The initial path is given by the Rapidly-Exploring Random Tree (RRT), [9], which provides a collision free sequence of poses between a start and goal poses. The initial path, however, does not guarantee the maximization of clearance to obstacles nor path smoothness. The optimization procedure works then as a post processing method, that improves the quality of the initial path. Each pose is treated as a rigid body, subjected to two types of forces: internal forces and external forces. The internal forces are the elastic force and torsional torque, originated from the virtual elastic and torsional springs, responsible for keeping consecutive poses connected and thus guaranteeing path smoothness. The external forces are the repulsive forces and torques that act on the rigid body (the vehicle's pose), resulting from obstacle proximity.

The final trajectory is generated by defining the velocity as a function of the minimum distance to obstacles, as described in section II-A. This motion planning methodology allows to fully explore the flexibility of the rhombic configuration, since the wheels are not constrained to follow the same path. In Fig. 4, it is shown an example where this methodology finds a solution that does not exist with the previous approach [1].

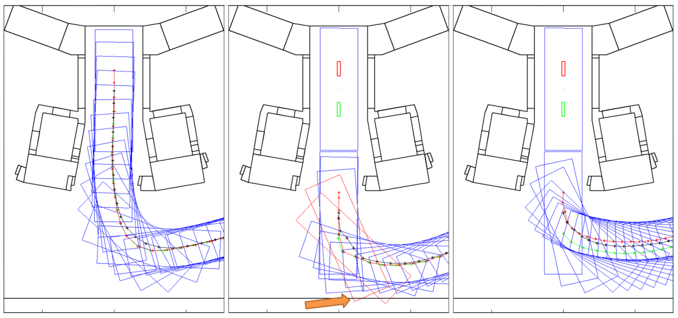


Fig. 4. Left - path shared by both wheels for a vehicle entering a port cell in TB, Center - path shared by both wheels, with collision, for a second vehicle entering the same port cell; Right - solution without collision with each wheel of the second vehicle following its own path.

C. *Maximization of the common path of different paths*

Given a set of paths that share the same starting pose, but differ on the arriving location, it becomes apparent that in terms of minimizing the areas accessed by the vehicle, it is logic, and required in ITER, to maximize the part of each path that is common to all paths. This leads to the definition of a common path, that is shared by all optimized paths, as shown in Fig. 5, where the common path starts in the lift area, covers a circular area around the reactor and returning to the lift. When a new path is generated, only the part that differs from the common path is optimized. This is performed by finding the nearest point on the common path, to the goal point. Usually, this nearest point is not the best starting condition for the path optimization, because it may require for the vehicle to make a sharp turn. A user defined threshold sets how further back from this nearest point the splitting point is defined (Fig. 5 - Right). When the start and goal configurations are defined, the optimization procedure can begin with either of the approaches described in II-A or II-B.

III. Localization

The localization problem consists in the estimation of the real pose relative to a global reference frame. During RH

[1]It is here assumed without proof that if the optimized path found by one of the presented algorithms incurs in a clash, then there is no feasible solution under the given constraints.

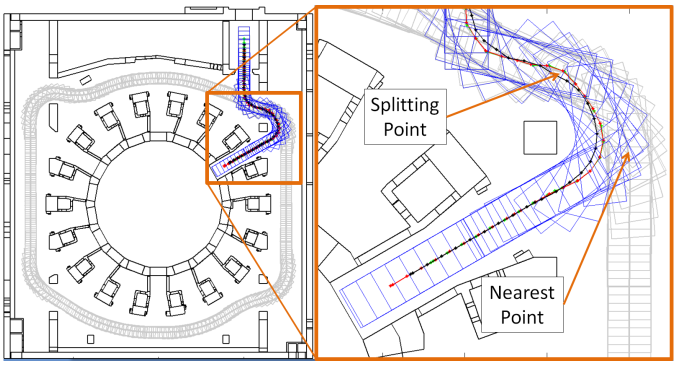


Fig. 5. Left - common path in TB depicted in grey; Right - detail view of an optimized path between the splitting point and the final point.

operations in ITER, the main radiation source will be the vehicle load, which means that on board sensors have a long and intense exposure which can shorten their lifetime. To overcome this constraint, it is proposed to install the sensors for vehicle localization on the building walls, reducing the exposure to radiation. This poses the challenge of where to place the sensors in the scenario and of how to perform the localization. Laser Range Finders (LRF) were adopted as sensors, since they are accurate and can be well shielded from radiation. The integration of a network with several sensors is necessary to cover all possible vehicle positions.

A. *Sensor Network Optimization*

A LRF sensor network is composed by several sensors, each one with the possibility of having a different parameterization. Sensor position and orientation are variable parameters chosen in the optimization process. Sensor field of view, angular resolution and standard deviation for distance measurement errors are fixed parameters that depend on the equipment. The optimization, herein described, maximizes, for a given number of sensors, the area obtained by the union of several visibility polygons, returning the sensor network parameterization with maximum coverage. An example LRF sensor network, with two sensors, is shown in Fig. 6 with visibility polygons for the respective sensors. The benefit of adding one more sensor to a network decreases as the number of sensors already present increases, [6]. The number of sensors to install is not optimized, it is picked based on a cost-benefit analysis of adding an extra sensor.

B. *Bayesian approaches for Localization*

Localization systems, with the framework presented in Fig. 6, give an estimation of vehicle pose integrating the measurements coming from the previously optimized LRF sensor network and the vehicle odometry.

Acquired measurements are distances from the corresponding LRF sensor to the nearest obstacle, in each direction. These directions depend only on sensor angular resolution and on the field of view. For each measurement acquisition, the directions are considered fixed and only the distances differ according to the surrounding obstacles.

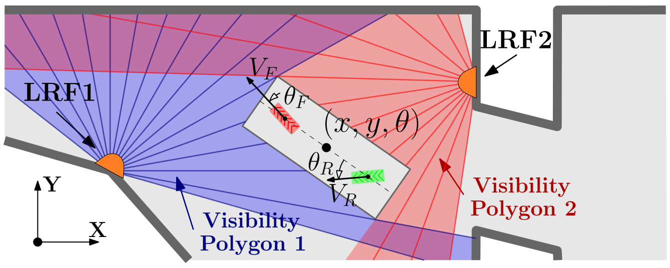


Fig. 6. Sensor Network and Localization system framework.

The two localization methods presented in this paper are standard Bayesian approaches, the Extended Kalman Filter (EKF), and bootstrap Particle Filter (PF), with new observation models, developed for this framework. The observation models are the main innovative contribution and this section is focused on its understanding.

1) Extended Kalman Filter: EKF uses a Jacobian matrix to relate the errors between the real and predicted measurements given estimated pose (innovation). There are four sets of measurements: i) the ones hitting the vehicle in real pose, ii) hitting the vehicle in predicted pose, iii) not hitting the vehicle in real pose, iv) not hitting the vehicle in predicted pose. A measurement is integrated by EKF only if it belongs to both i) and ii). This means that not all measurements can be integrated, only the ones that hit the vehicle. The residuals corresponding to measurements hitting the scenario walls have a Jacobian entrance equal to zero, because the distance measured to the wall do not depend directly on vehicle pose. This fact forces the method to neglect some information. Moreover, if the prediction is too far from reality, EKF does not integrate any measurement.

EKF predicted pose must be always near the real pose, otherwise the update step of EKF is ineffective. To overcome this problem, the number of measurements integrated on each iteration, is monitored, and, every time it drops below a certain threshold, EKF is restarted. It is possible to get a position estimation, for the restarting iteration, transforming measurements hitting the vehicle into Cartesian points and doing his mass center. The initial belief given for the restarting iteration of EKF has this mass center position, a random orientation and a high uncertainty. As the position is close to real one, the estimation converges to the real and uncertainty reduces.

The restarting step enables the global localization on the scenario, something that is not possible, using EKF, with on-board sensors.

2) Particle Filter: PF uses a set of particles to represent hypothetical poses of the vehicle. For each one of these poses, the observation model compares the predicted measurements with the real ones, assigning a likelihood to the respective particle. It is possible, with this approach, to integrate all measurements from the network, and to adapt the observation model to the framework.

Observation model is a likelihood function that assumes measurements independent but not identically distributed. It distinguish two different distributions if the predicted measurements hit the hypothetical vehicle or not.

Let $\mathcal{N}(\mu, \sigma^2)$, be a normal distribution with μ mean and σ^2 variance. σ^2 is the variance assumed for the measurements, always greater then real measurement variance. $\mathcal{U}(a, b)$ is an uniform distribution with limits a and b.

For each measurement, the distribution is,

i) If it hits the vehicle, a linear mixture of:
   - $\mathcal{N}(d, \sigma^2)$, modeling measurements that hit the vehicle also in reality;
   - $\mathcal{N}(D, \sigma^2)$, modeling measurements that hit the walls in reality;
   - $\mathcal{U}(0, range)$, modeling outliers.

   The weight of $\mathcal{N}(d, \sigma^2)$ should always be greater then the others to reinforce the particles with correct prediction.

ii) If it hits the walls, a linear mixture of:
   - $\mathcal{N}(D, \sigma^2)$, modeling measurements that hit the walls also in reality;
   - $\mathcal{U}(0, D)$, modeling measurements that hit the vehicle in reality;
   - $\mathcal{U}(0, range)$, modeling outliers.

   The weight of $\mathcal{N}(D, \sigma^2)$ should be the highest to ensure that correct predictions have greater likelihood.

Being d the predicted distance to the vehicle, D the known distance to the nearest wall and $range$ the maximum range of the sensor. PF have the possibility for global localization, using the same principle used for EKF. When the measurement likelihood drops abruptly, the probability of generating particles around the measurement mass center rises, and, as PF can represent multi-modal distributions, a new mode starts to appear on this mass center. After some resample steps, all particles migrate near the correct pose of the vehicle and the likelihood rises again. The situation is similar to kidnapped mobile vehicle, but, with a global network of sensors, an approximation of the real pose, the measurement mass center, is easily discovered.

IV. Simulated Results

The simulation results were obtained with a software application tool developed in MATLAB environment: the Trajectory Evaluator and Simulator (TES), as illustrated in Fig. 7. The TES was developed not only to generate trajectories for ITER scenarios, but also to generate trajectories in a general map. The TES has a diversity of features that, besides trajectory generation, allow to do reports with information on the minimum distances to obstacles along the trajectories, as well as the location of the critical points in the scenario, to assess the risk of collision. It provides the area spanned by the vehicle along the path and it provides the option to export this information as a 3D CAD model to a CAD software, such as CATIA. Besides trajectories, TES can also simulate a basic guidance and localization system for the vehicle, controlling it along a given trajectory and giving a pose estimation based on simulated noisy LRF measurements.

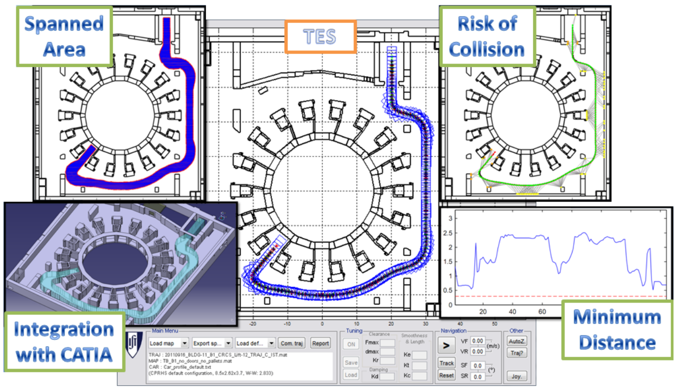


Fig. 7. TES main window and features.

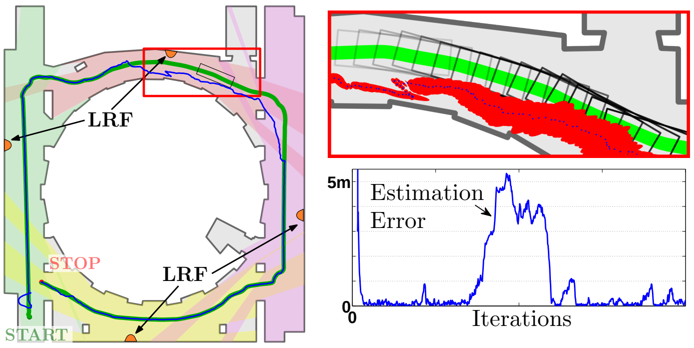


Fig. 9. Real trajectory (green) EKF estimated trajectory (blue) (left), Zoom box with estimated positions (blue) and certainty ellipses (red) (top right), Position error along the path (bottom right)

A. *Optimized Trajectories*

In TB, the CPRHS is required to dock in predefined locations during nominal operations. However, if a malfunction occurs during docking, trajectories for rescue missions were also required, where a CPRHS has to provide assistance to an already docked CPRHS (e.g., Fig. 3 Center and Right). In Fig. 8, optimized paths for docking and the corresponding optimized rescue paths are shown for one of the floors of TB. A total of 232 trajectories were computed in TB. The HCB is where the CPRHS is required to dock for loading/unloading operations and where parking areas for the vehicle are provided. On the right side of Fig. 8 it is shown optimized parking paths for one of the levels of HCB. A total of 304 trajectories were generated in HCB.

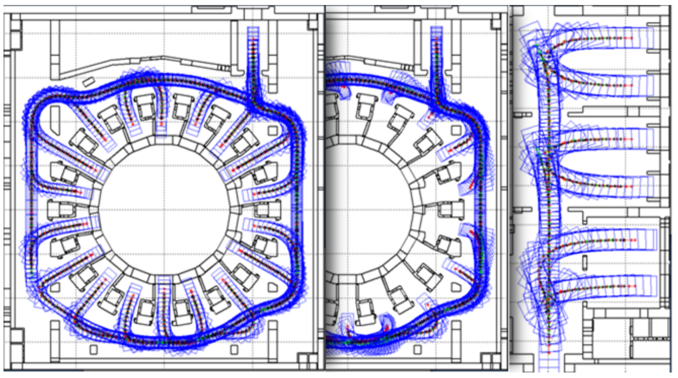


Fig. 8. Left - optimized paths for rescue missions of the CPRHS in TB; center - optimized paths for docking missions of the CPRHS in TB; right - optimized paths for parking missions of CPRHS in HCB.

B. *Localization*

The two localization approaches were implemented and compared in TES. The simulation results, consider a network of 4 sensors, with angular resolution of 1^o and standard deviation for distance measurement of $10cm$. Both approaches can localize the vehicle correctly due to the adopted observation models. EKF performance, in Fig. 9, presents some limitations in accuracy and robustness but it has a very interesting computation performance. In particular situations, like the one highlighted in Fig. 9 (top right), EKF loses stability and the estimation error becomes very high. Red ellipses show that EKF estimation has an unacceptable high uncertainty, facing the tight safety margins inside ITER.

PF performance, in Fig. 10, presents very reliable results, it is very accurate and robust to all situations tested in ITER scenarios. The results show very small uncertainty, shown by the small red ellipses in Fig. 10 (top right), low estimation errors, and no losses of stability. PF downside is the high computation effort required, it is 30 times slower then EKF for the sensor network used on this simulation. PF integrates all measurements while EKF takes only the ones hitting the vehicle. Position estimation errors, for EKF and PF, along the trajectory, are presented on Fig. 9 and Fig. 10 (bottom right), respectively. Both approaches are initialized with random estimation, explaining the high estimation error in the beginning. Both approaches are able to converge from this random pose due to the global localization feature explained on previous section. Comparing both approaches, EKF estimation error is unacceptable for an ITER application while PF present very reliable results. On this simulation, mean position estimation error for EKF is $0.9m$ while for PF it is $0.06m$. Maximum estimation error, after global localization, is $5.3m$ for EKF and for PF is $0.19m$. From these two localization approaches, PF is the most appropriate for application with this framework. EKF performance rises with the number of sensors installed on the scenario. With many sensors it is guaranteed that EKF always integrate many measurements becoming more accurate. PF becomes more accurate with more sensors, but with very high computation cost.

V. Experimental Results

The experimental setup, shown in Fig. 11 (left), includes a single Hokuyo LRF sensor and a CPRHS prototype built in LEGO Mindstorms. The LRF sensor has a field of view of 240^o and angular resolution of 0.36^o. The prototype, with a 1:25 scale, describes a simple trajectory in a rectangular map without obstacles. For simplicity, the described trajectory is not a result from optimization, it is the result from simple commands sent directly from operator to the vehicle.

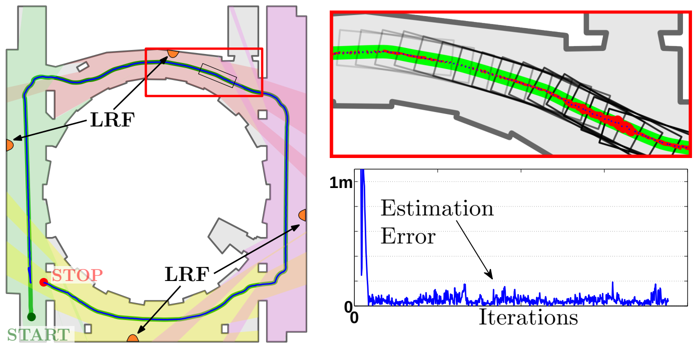


Fig. 10. Real trajectory (green) PF estimated trajectory (blue) (left), Zoom box with estimated positions (blue) and certainty ellipses (red) (top right), Position error along the path (bottom right)

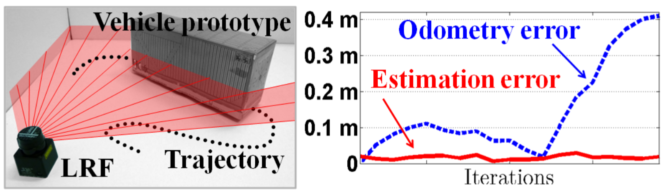


Fig. 11. Experimental framework (left), Error along trajectory (right)

Way points along the trajectory were manually registered and compared with the ones estimated by the localization system (Fig. 12). The position estimation error, presented in the plot of Fig. 11 (right), is below 0.3cm and the orientation error is less then 8^o. The results are biased by the erroneous odometry, due to prototype encoder resolution and wheel slippage. The error between real position and integrated position, given the vehicle odometry, is also shown in Fig. 11 (right).

The estimations, for this experiment, were computed offline. The data acquisition rate was $5Hz$, but PF mean computation rate, with 300 particle, was $4Hz$. The PF approach achieves good results, even with a highly erroneous odometry, the main problem for a real time implementation is the computational effort required.

VI. Conclusions and Open Issues

For the generation of the optimized paths, the two proposed path planning methodologies were used. The majority of the trajectories are feasible with the line guidance approach, incorporating maneuvers whenever necessary. The free roaming approach is of great importance in rescue situations, where the flexibility of the rhombic configuration is explored to compute feasible paths, where the line guidance method fails.

The implemented localization methods, EKF and PF, using a sensor network of LRF outside of the vehicle, were able to locate the vehicle in a simulated environment. The PF method proved to more reliable than the EKF, even in the case of sensor failure, however, the performance is highly dependent of the total coverage by the sensor network.

Further work is required to evaluate the performance of the localization methods in terms of sensor redundancy and compare the localization results between line guidance and free roaming trajectories. Control methods in real time that take into consideration the experimental localization results, should also be addressed.

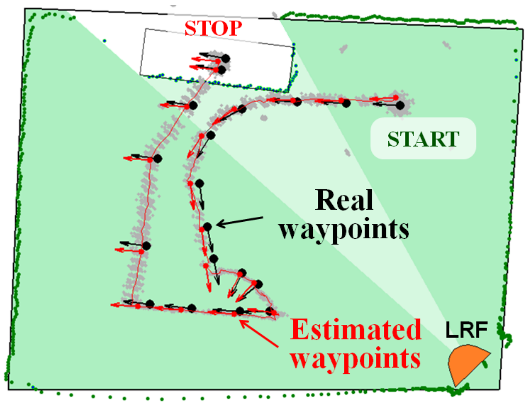


Fig. 12. Experimental trajectory with real vehicle poses (black) and Estimated poses (red)

Acknowledgment

The work was supported by the grants F4E-2008-GRT-016 (MS-RH) and F4E-GRT-276-01 (MS-RH) funded by the European Joint Undertaking for ITER and the Development of Fusion for Energy (F4E) and by FCT in the frame of the Contract of Associate Laboratories of Instituto de Plasmas e Fusão Nuclear/IST and Laboratório de Robótica e Sistemas em Engenharia e Ciências/IST (PEst-OE/EEI/LA0009/2011). The views expressed in this publication are the sole responsibility of the authors. F4E is not liable for the use which might be made of the information in this publication.

References

[1] I. Ribeiro, C. Damiani, A. Tesini, S. Kakudate, M. Siuko and C. Neri, *The Remote Handling Systems for ITER*, Fusion Engineering and Design, vol. 86, pp. 471-477, 2011.
[2] C. Gonzlez, C. Damiani, J-P. Friconneau, A. Tesini, I. Ribeiro and A. Vale, *ITER Transfer Cask System: status of design, issues and future development.*, Fusion Engineering and Design, vol. 85, pp. 2295-2299, 2010.
[3] I. Ribeiro, P. Lima and R. Ferreira, *Conceptual Study on Flexible Guidance and Navigation for ITER Remote Handling Transport Casks*, Proc. of the 17th IEEE/NPSS Symp. on Fusion Engineering, pp. 969-972, San Diego, USA 1995.
[4] D. Fonte, F. Valente, A. Vale and I. Ribeiro, *A motion planning methodology for rhombic-like vehicles for ITER remote handling operations*, 7th Symp. on Intelligent Autonomous Vehicles, Italy, 2010.
[5] D. Fonte, F. Valente, A. Vale and I. Ribeiro, *Path Optimization of Rhombic-Like Vehicles: An Approach Based on Rigid Body Dynamic*, Proc. of the 15th IEEE International Conf. on Advanced Robotics, pp. 106-111, Tallin, Estonia 2011.
[6] J. Ferreira, A. Vale and R. Ventura, *Optimizing range finder sensor network coverage in Indoor Environment*, 7th Symp. on Intelligent Autonomous Vehicles, Italy, 2010.
[7] L. P. Chew, *Constrained Delaunay Triangulations*, Proc. of the Third Annual Symp. on Computational Geometry, pp 215-222, Waterloo, Ontario, Canada, 1987.
[8] S. Quinlan and O. Khatib, *Elastic Bands: Connecting Path Planning and Control*, Proc. IEEE Conference Robotics and Automation, vol. 2, pp 802-807, Atlanta, USA, 1993.
[9] S. M. La Valle and J. J. Kuffner, *Rapidly-exploring random trees: Progress and prospects*,In B. R. Donald, K. M. Lynch and D. Rus, editors, Algorithmic and Computational Robotics: New Directions, pp 293-308, A K Peters, Wellesley, MA, 2001.
[10] J.S. Gutmann, W. Burgard, D. Fox, and K. Konolige, *An experimental comparison of localization methods*, Proc. of the IEEE/RSJ International Conference, pp. 736–743, IEEE, 2002.

BatSLAM: Combining Biomimetic Sonar with a Hippocampal Model

Jan Steckel
University of Antwerp
FTEW MTT Department
Prinsstraat 13, B-2000 Antwerpen
Belgium
Jan.Steckel@ua.ac.be

Dieter Vanderelst
University of Antwerp
FTEW MTT Department
Prinsstraat 13, B-2000 Antwerpen
Belgium

Herbert Peremans
University of Antwerp
FTEW MTT Department
Prinsstraat 13, B-2000 Antwerpen
Belgium

***Abstract*—Using RatSLAM, a system based on a hippocampal model for Simultaneous Localization and Mapping (SLAM) of mobile robots, as inspiration we propose a novel sonar based SLAM system, called BatSLAM. To provide information about the environment a single, biomimetic sonar sensor is used. The biomimetic sonar system consists of a Polaroid transducer generating the emitted signal and two plastic replicas of the pinnae of the bat Micronycteris *microtis*, equipped with Knowles subminiature microphones implementing the receiver Head Related Transfer Function (HRTF). Experimental evidence is presented to show that the readings provided by this sonar system contain sufficient information to allow construction of a consistent map in a normal, unmodified office environment.**

***Index Terms*—Ultrasonics, Bat echolocation, Biomimetic Sonar, SLAM**

I. Introduction

A. Biomimetic Sonar Systems

Bats have evolved a very complex yet robust sonar system capable of echolocation performances far beyond the current state of the art in robotics [1], [2]. Using their sonar systems, bats are able to navigate in complex environments, detect and hunt prey, find roosting places and build a spatial memory map [3]. Inspired by bat echolocation, a few robotic implementations of biomimetic sonar systems exist. We propose to use this biomimetic sonar system as sensor input to a bio-inspired SLAM system.

We show that this system is capable of building a consistent map in a normal office environment, indicating that the sonar data contains enough information for the description of places in space. By utilizing only biologically plausible signal processing techniques that have been shown to exist in the bat's auditory cortex, we postulate this as a hypothesis for bat spatial map building.

B. Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) is a technique for the construction of maps of the environment based on sensory input about the robot's egomotion (shaft encoders, inertial systems, ...) and information about the position of external landmarks in relation to the robot [4], [5]. Traditionally, this is done using probabilistic methods such as Kalman filters [6] (assuming Gaussian noise), Extended Kalman filters [4],or more recently particle filters [7]. These methods all attempt to generate metric maps of the environment so that a higher level planner can steer the robot to fulfill some meaningful task. Topological methods for SLAM have been investigated as well [8]. These methods generate topological maps of the environment. While some of the original SLAM work has been performed using sonar sensors ([5], [9]), these have now mostly been replaced by sensors providing more fine-grained information such as vision sensors, cameras or laser scanners [4].

Parallel to the traditional SLAM systems based on probabilistic methods, biologically inspired SLAM systems have also been developed. A highly successful example thereof is the RatSLAM system [10]–[12], which is inspired on the rat's hippocampal pose cell network. Neuro-physiological experiments [13] suggest that the mammalian hippocampus in general and the bat's hippocampus in particular [14] contain so called place cells [15] which encode the absolute position of the animal in it's environment. The pose cell network can be envisioned as a Competitive Attractor Network (CAN, [16], [17]), which performs path integration using information from proprioceptive sensors as well as information about environmental landmarks coming from exteroceptive sensors. In this respect this scheme is quite similar to the traditional SLAM systems where odometry is used to generate a hypothesis of landmark displacement and the exteroceptive sensors are used to check this hypothesis and correct the odometric input by minimizing some error criterion.

The rest of the paper is structured as follows: section II explains the biomimetic sonar system and the processing involved. Section III provides insights in the cooperation of the RatSLAM core with our biomimetic sonar system. Section IV shows experimental results from a mapping experiment, followed by some conclusions.

II. Sonar Systems: the biomimetic approach

A. Bat echolocation

Bats navigate in their environment by emitting high-frequency vocalizations and interpreting the returning echoes [1]. Elsewhere we have shown that the bat's facial features such as the noseleaf and outer ears (pinnae) interact with the sound field during emission and reception respectively, thereby performing spectrospatial filtering on the returning echoes [18]. This interaction of the echo sound field with the outer ears encodes reflector location in a diverse set of monaural and binaural cues such as echo spectrum, interaural intensity differences (IID) and interaural time differences (ITD) [19], [20], giving rise to the so called Head Related Transfer Function (HRTF) [21]–[23] introduced in spatial hearing theory [24]. For sonar systems we have coined the term Echolation Related Tranfer Function (ERTF) to denote the combination of the HRTF with the spectrospatial emission pattern, encapsulating all spectrospatial transformations performed on the signals from the point of signal emission to the point of signal reception.

After the reception and filtering of the echoes by the pinnae, the signals are processed in the cochlea resulting in a time-frequency representation. A somewhat simplified, yet effective, functional model of the cochlea consists of a bank of bandpass filters with a gammatone response [24], [25], followed by half-wave rectification, compression and lowpass filtering to extract the envelope of the signals in each frequency channel [26].

It has been shown that bats make use of landmarks while executing navigation tasks [27]. This evidence in combination with the fact that place cells have also been found in the bat's hippocampus [14] advocates in favor of the hypothesis that bats make use of a system similar to the RatSLAM system, with the vision module replaced by a sonar module, to construct maps of their environments and use these maps during the execution of navigation tasks.

B. Echo Signal representation in biomimetic sonar systems

As stated above, during emission the signals are filtered by the emitter spectrospatial sensitivity pattern. Next, the emitted vocalizations are reflected by the environment and filtered by the bat's HRTF upon reception. Eqs. (1) and (2) describe the echo filtering process resulting in the left ear filter $H_e^L(f,\theta)$

$$H_e^L(f,\theta) = H_{em}(f,\theta)\cdot H_a(f,r)\cdot H_r(f)\cdot H_h^L(f,\theta), \quad (1)$$

and the right ear filter $H_e^R(f,\theta)$

$$H_e^R(f,\theta) = H_{em}(f,\theta)\cdot H_a(f,r)\cdot H_{refl}(f)\cdot H_h^R(f,\theta) \quad (2)$$

with $H_{em}(f,\theta)$ the emitter directivity and $H_h^L(f,\theta)$, $H_h^R(f,\theta)$ the left and right receiver directivities for frequency f and direction θ. The angle θ denotes the unique azimuth and elevation combination that specifies a reflector direction relative to the sonar system in a 3D world. The filtering due to sound propagation through air is taken into account by the factor $H_a(f,r)$. This factor includes frequency independent attenuation due to spherical spreading, frequency dependent absorption and the propagation delay introduced by the finite speed of sound [28]. Lastly, the filtering due to the interaction of the sound field with the shape of the reflector is denoted by $H_{refl}(f)$. In general, this filter will depend on the pose of the reflector relative to the direction of the incident sound field ([20], [29].

As all the signal operations are linear in the frequency domain, the ERTF, which is the combination of the emission and the reception directivity patterns can be written as

$$\begin{aligned} H_{ERTF}^L(f,\theta) &= H_{em}(f,\theta)\cdot H_{h(f,\theta)}^L \quad (3)\\ H_{ERTF}^R(f,\theta) &= H_{em}(f,\theta)\cdot H_{h(f,\theta)}^R \quad (4)\end{aligned}$$

To generate the received signal at the input of the cochlea for the left and the right ear in the presence of multiple reflectors, we continue the analysis in the time domain. The received signals at the left cochlea ($s_e^L(t)$) and the right cochlea ($s_e^R(t)$) can be written as

$$s_e^L(t) = \sum_{i=1}^{n_e} h_e^L(t;\theta_i) * h_{refl}^i(t) * s_c(t-\delta_i) \quad (5)$$

$$s_e^R(t) = \sum_{i=1}^{n_e} h_e^R(t;\theta_i) * h_{refl}^i(t) * s_c(t-\delta_i) \quad (6)$$

with $s_c(t)$ denoting the emitted call, $h_e^L(t;\theta_i)$ and $h_e^R(t;\theta_i)$ the impulse responses of the ERTF filters in the time domain for the left and right ears, δ_i the delay introduced by the i-th reflector, θ_i the direction of the i-th reflector, h_{refl}^i the filtering due to the i-th reflector, and n_e the total number of reflectors present in the environment.

In order to analyze the features of these binaural echo signals in a biologically relevant time-frequency representation, we model the processing performed by the bat's cochlea. The operation of the cochlea can be approximated using a bank of gammatone bandpass filters, followed by half-wave rectification, compression and a lowpass filter. The gammatone response $h^{gt}(t;f_c(n))$ for the n-th frequency channel with center frequency $f_c(n)$ and bandwidth $B(n)$, can be written as [25]

$$h^{gt}(t;f_c(n)) = t^3\cdot e^{-2\pi B(n)\cdot t} cos(2\pi f_c(n)\cdot t) \quad (7)$$

Applying first a half-wave rectification and compression function $g^{am}\{\}$ and then a lowpass filter $h^{LP}(t)$ to the outputs of the gammatone filters $h^{gt}(t;f_c(n))$, the cochleogram representations $S_L^{gt}(t,f_c(n))$ for the left ear and $S_R^{gt}(t,f_c(n))$ for the right ear can be written as

$$S_L^{gt}(t,f_c(n)) = g^{am}\left\{s_e^L(t) * h^{gt}(t;f_c(n))\right\} * h^{LP}(t) \quad (8)$$

$$S_R^{gt}(t,f_c(n)) = g^{am}\left\{s_e^R(t) * h^{gt}(t;f_c(n))\right\} * h^{LP}(t) \quad (9)$$

with the operator $*$ denoting time-domain convolution.

C. Implementation of biomimetic Sonar System

The hardware for the biomimetic sonar system used in this work has been described in detail elsewhere ([29]). The system consists of a Polaroid transducer ([30]), driven by a custom made high-voltage amplifier. Connected to this

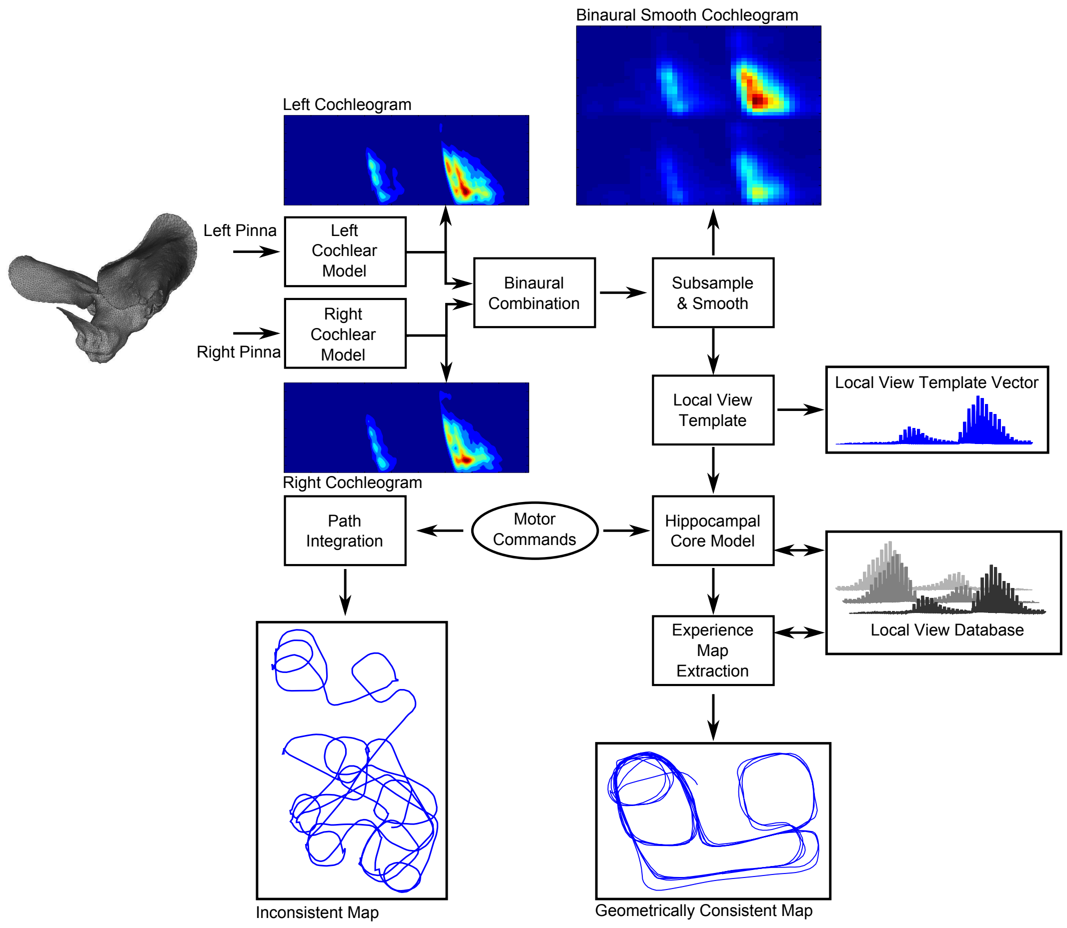


Fig. 1. Overview of the BatSLAM system architecture. The echoes are filtered using the pinna and emitter spatial sensitivities and analyzed using a functional model of the mammalian cochlea. The cochlear responses are subsampled, smoothed and concatenated, yielding a local view descriptor which is fed to the SLAM core system consisting of a place cell (P-cell) network and experience learning module. Motor commands sent to the robot are also fed into the P-cell network to perform crude path integration. Using only path integration without the BatSLAM system, no consistent map of the robot's trajectory can be constructed, while extra sensory input from the sonar enables the BatSLAM system to build a consistent map of the environment.

amplifier is a custom-made Digital-to-Analog converter which generates the emitter signal. Two plastic replica of the pinnae of Micronycteris *microtis*, equipped with two Knowles FG-23329 sub-miniature microphones ([31]) generate the receiver HRTF in this system. Figure 2 shows the emitter spatial sensitivity patterns, HRTF patterns for the left pinna and the combined ERTF patterns for the left pinna.

As the sensory input to the BatSLAM system consists of binaural cochleograms derived from the signals coming out of the left and right pinnae , it is interesting to investigate the differences between the two ERTF patterns. Indeed, the per-frequency Interaural Intensity Differences (IID) have been shown to be an important cue for azimuthal localization of targets [24]. Figure 3 shows the left ERTF, the right ERTF and the IID (calculated by subtracting the right ERTF from the left ERTF). It should be noted that the IID patterns contain a high amount of information, advocating the use of a binaural system over a monaural system, because the binaural cues enrich the sensor information, alleviating the view recognition task.

To process the received left and right ear echo signals and calculate the left and right cochleograms, we have implemented a gammatone filterbank as described in [32], [33]. The filterbank consists of 25 constant-Q (Q=10) channels spaced logarithmically between 20kHz and 80kHz, followed by half-wave rectification and a leaky-integrator lowpass filter with a cutoff frequency of 250Hz.

III. BatSLAM

In the RatSLAM algorithm, Local View (LV) cells are used to store scenes the robot has encountered previously and to build links between the sensor information (associated with a LV cell) and the pose cell network. As the robot explores the environment, new LV cells are added to the database as the new sensor data is experienced by the robot. Links between the pose cell network and the LV cells are strengthened using Hebbian learning. Details of the LV cell architecture can be found in [10].

For the biomimetic sonar system to replace the vision sensor in the original RatSLAM system, special care in the preparation of the sonar data has to be taken. First, the monaural cochleogram needs to be subsampled in time (to make the system more robust for small variations in robot

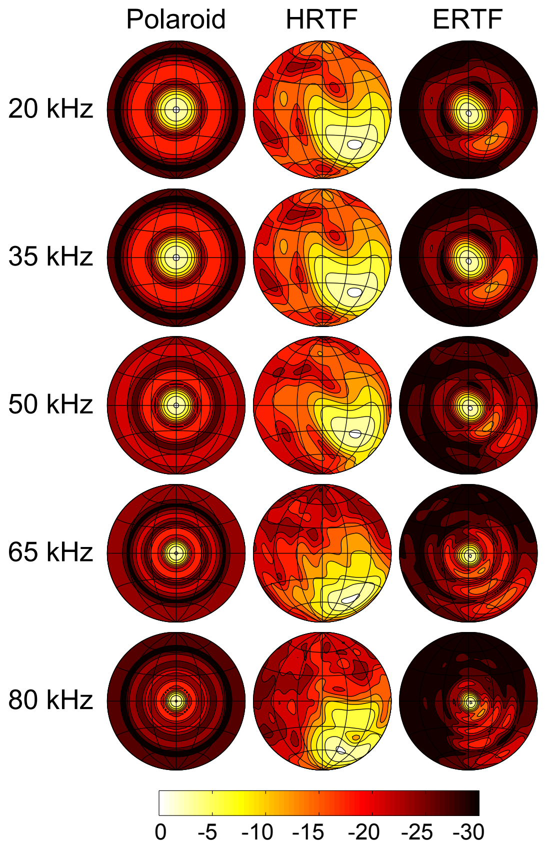


Fig. 2. HRTF, emitter and ERTF directivity patterns for the robotic setup projected using a Lambert equal area projection ranging from 20kHz to 80kHz (the frequency range of the filterbank). The HRTF is the HRTF of the replica of the plastic pinnae from Micronycteris *microtus*, the emitter directivity pattern is the directivity of the Polaroid transducer. 30° grid lines, 3dB contour lines

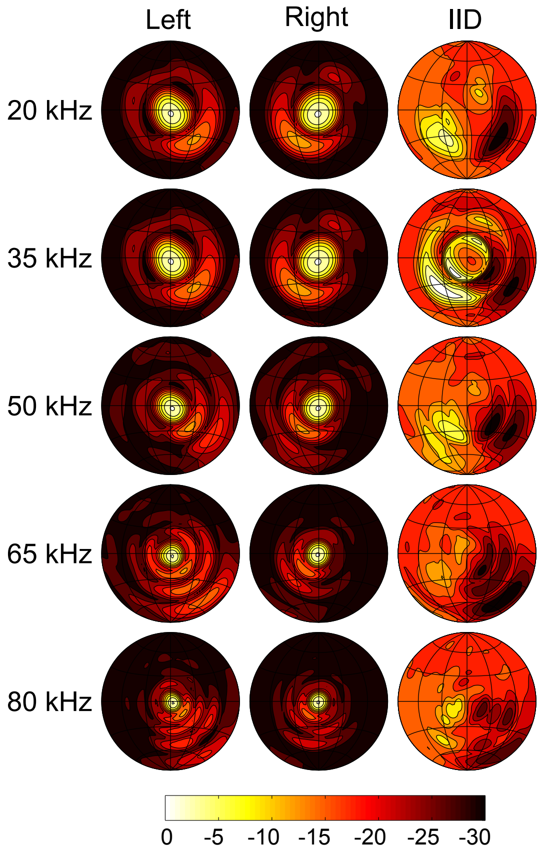


Fig. 3. ERTF Directivity for left and right ear and also IID, illustrating the importance of having two ears using a Lambert equal area projection, 30° grid lines, 3dB contour lines

position) and smoothed with a Gaussian smoothing filter (to further increase the system's invariance to small position differences). Additionally, the two subsampled and smoothed monaural cochleograms are concatenated to form a single binaural cochleogram $S^{gt,SS}_{Bin}$

$$S^{gt,SS}_{Bin} = [S^{gt,S}_{L} * G^{S}, S^{gt,S}_{R} * G^{S}] \tag{10}$$

with $S^{gt,S}_{L}$ the left subsampled cochleogram, $S^{gt,S}_{R}$ the right subsampled cochleogram, and G^{S} the Gaussian smoothing filter. As the intensity differences between the left and the right ear are very informative [24], the monaural cochleograms are not normalized. Figure 3 shows the IID patterns for the robotic setup. The third column reveals that there is much spatial information encoded by the differences between the left and the right echo signals. However, for the system to be robust to small variations in echo strength (due to emitter fluctuations, position and orientation errors), the binaural, smoothed cochleogram $S^{gt,SS}_{Bin}$ is normalized to have an energy content of one, yielding the normalized binaural smoothed cochleogram $\widetilde{S}^{gt,SS}_{Bin}$

$$\widetilde{S}^{gt,SS}_{Bin} = \frac{S^{gt,SS}_{Bin}}{\sum_t \sum_f |S^{gt,SS}_{Bin}|^2} \tag{11}$$

We propose to use this smoothed and subsampled cochleogram as the local view descriptor LV. To test if a local view has occurred before, the RatSLAM core system calculates the euclidean distance d_i between the current LV and all the stored LV templates in the database

$$d_i = \sqrt{\sum (\widetilde{S}^{gt,SS}_{Bin} - LV_i)^2} \tag{12}$$

with LV_i denoting the i-th template in the database and $\widetilde{S}^{gt,SS}_{Bin}$ the current smoothed cochleogram. If the euclidean distance is below a certain threshold, the LV is considered to correspond with a previous one, and energy is injected into the corresponding region of the place-cell network by the RatSLAM core. The threshold τ is calculated adaptively using a scaled version of the RMS value of the ensemble of all euclidean distances d_i

$$\tau = \alpha_t \cdot \sqrt{\frac{\sum_{i=1}^{n_{LV}} d_i^2}{n_{LV}}} \tag{13}$$

with n_{LV} the number of local views in the database and α_t a scaling factor to tune the system. In our experiments, α_t

was set to 0.5. The active LV cell is the one with the lowest distance d_i falling below the threshold. If no distance d_i falls below the threshold, the scene is considered as a new one, and no energy is injected from the LV network into the pose cell network.

IV. Experimental Results

To verify the operation of the BatSLAM system, we have tested the system in a realistic office environment. We drove the robot during 15 minutes on a route consisting of numerous turns, a multitude of closed loops, long hallways, open areas and tight corners, and recorded 1200 sonar snapshots. The robot had maximal linear speeds of 0.3m/s and maximal rotational speeds of 20°/s We ran the BatSLAM algorithm on the recorded data, and constructed the experience map for the entire run. Figure 5, left column shows the trajectory generated by odometry through path integration only. Large angular errors can be observed, and no consistent map is built from the data. Figure 5, right column shows the constructed map from the BatSLAM system. Consistency is preserved throughout the entire sequence. It should be noted that the geometric properties of the map are not 100% consistent with the reality. This is due to the fact that the map that is constructed with the BatSLAM system (using the RatSLAM core) has some geometric properties, but is in fact a topological map, where experiences are connected to other experiences through links on a graph representation.

In order to establish the uniqueness of local views, we have highlighted four LV's in figure 4. The red dot is the position where the LV was first encountered, the black dots are the experiences where the same LV is being recognized by the system again. It should be noted that for VT223, which is located in a long hallway, several other LV's in the same hallway are linked to this VT. This is caused by the highly similar properties of the environment. However, the SLAM system is able to recover from these minor ambiguities with relative ease. Figure 4 also shows the cochleogram $\widetilde{S}_{Bin}^{gt,SS}$ assigned to the first LV.

V. Conclusions

We have demonstrated a biologically inspired SLAM system based on the RatSLAM architecture. The vision module from the original system has been replaced by a biomimetic sonar system. The local views are described using a smoothed version of the cochlear output. We have demonstrated the capabilities of mapping a large office environment using the BatSLAM system.

Acknowledgments

This work was funded by the EU CHIROPING (Chiroptera, Robots, Sonar) project. This project is funded by the Seventh Framework Programme ICT Challenge 2: Cognitive Systems, Interaction, Robotics. The funder had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript. The Matlab code for the RatSLAM model has been kindly provided on the website of the authors thereof and can be found at [34]

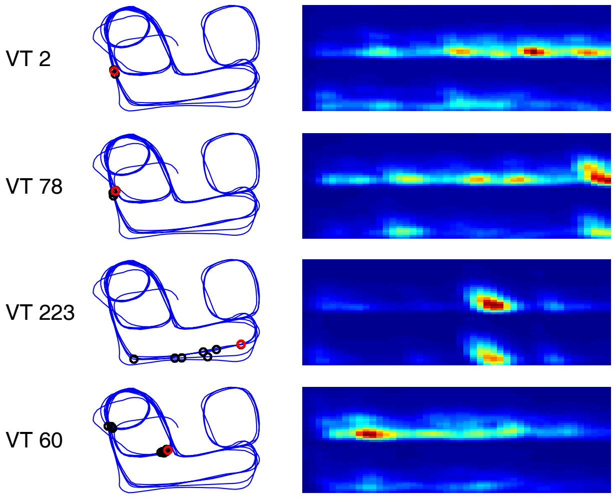


Fig. 4. Local view which are recognized multiple times. The red dot is the experience where the LV is first encountered, the black dots are the experiences where the LV is recognized. The second column shows the cochleogram $\widetilde{S}_{Bin}^{gt,SS}$ associated with the red LV.

References

[1] J. Thomas, C. Moss, M. Vater, and P. Moore, *Echolocation in bat and dolphins*. University Of Chicago Press, 2002.

[2] D. Griffin, "Listening in the dark: the acoustic orientation of bats and men." 1958.

[3] G. Neuweiler, *The biology of bats*. Oxford University Press, USA, 2000.

[4] H. Choset, *Principles of robot motion: theory, algorithms, and implementation*. The MIT Press, 2005.

[5] J. Leonard and H. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot," in *Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on*. Ieee, 1991, pp. 1442–1447.

[6] M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *Robotics and Automation, IEEE Transactions on*, vol. 17, no. 3, pp. 229–241, 2001.

[7] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *International Joint Conference on Artificial Intelligence*, vol. 18. LAWRENCE ERLBAUM ASSOCIATES LTD, 2003, pp. 1151–1156.

[8] H. Choset and K. Nagatani, "Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization," *Robotics and Automation, IEEE Transactions on*, vol. 17, no. 2, pp. 125–137, 2001.

[9] J. Tardós, J. Neira, P. Newman, and J. Leonard, "Robust mapping and localization in indoor environments using sonar data," *The International Journal of Robotics Research*, vol. 21, no. 4, p. 311, 2002.

[10] M. Milford, G. Wyeth, and D. Prasser, "RatSLAM: a hippocampal model for simultaneous localization and mapping," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 1. IEEE, 2004, pp. 403–408.

[11] M. Milford and G. Wyeth, "Mapping a suburb with a single camera using a biologically inspired slam system," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 1038–1053, 2008.

[12] G. Wyeth and M. Milford, "Spatial cognition for robots," *Robotics & Automation Magazine, IEEE*, vol. 16, no. 3, pp. 24–32, 2009.

[13] D. Derdikman and E. Moser, "A manifold of spatial maps in the brain," *Trends in cognitive sciences*, vol. 14, no. 12, pp. 561–569, 2010.

[14] N. Ulanovsky and C. Moss, "Hippocampal cellular and network activity in freely moving echolocating bats," *Nature neuroscience*, vol. 10, no. 2, pp. 224–233, 2007.

[15] E. Rolls and G. Deco, *Computational neuroscience of vision*. Oxford University Press, 2002.

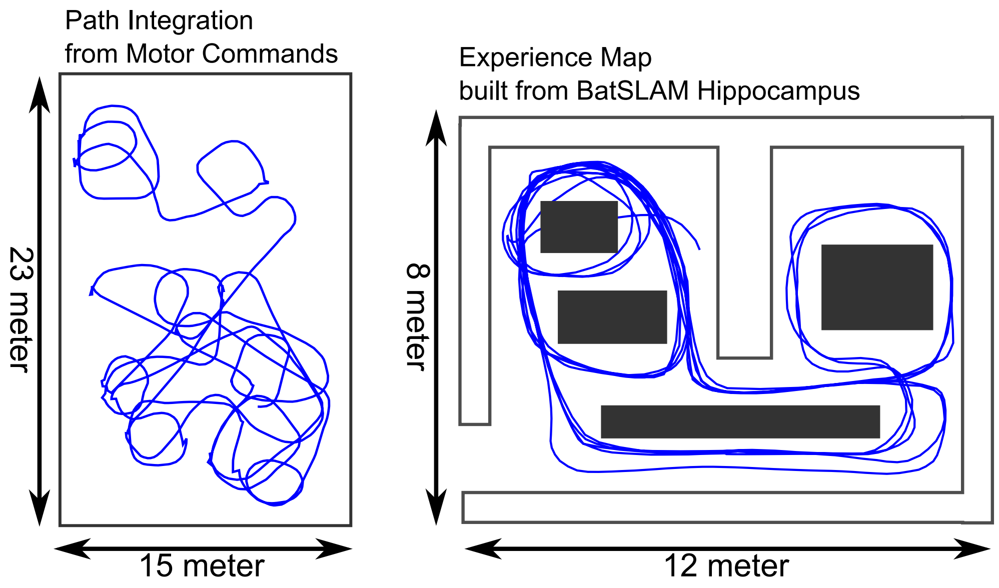


Fig. 5. Left: Resulting map built using path integration from motor commands. Right: Resulting map built using the BatSLAM system

[16] S. Stringer, T. Trappenberg, E. Rolls, and I. Araujo, "Self-organizing continuous attractor networks and path integration: one-dimensional models of head direction cells," *Network: Computation in Neural Systems*, vol. 13, no. 2, pp. 217–242, 2002.

[17] S. Stringer, E. Rolls, T. Trappenberg, and I. De Araujo, "Self-organizing continuous attractor networks and path integration: two-dimensional models of place cells," *Network: Computation in Neural Systems*, vol. 13, no. 4, pp. 429–446, 2002.

[18] D. Vanderelst, F. De Mey, H. Peremans, I. Geipel, E. Kalko, and U. Firzlaff, "What Noseleaves Do for FM Bats Depends on Their Degree of Sensorial Specialization," *PloS one*, vol. 5, no. 8, p. e11893, 2010.

[19] N. Ulanovsky and C. Moss, "What the bat's voice tells the bat's brain," *Proceedings of the National Academy of Sciences*, vol. 105, no. 25, p. 8491, 2008.

[20] J. Reijniers, D. Vanderelst, and H. Peremans, "Morphology-Induced Information Transfer in Bat Sonar," *Physical Review Letters*, vol. 105, no. 14, p. 148701, 2010.

[21] M. Obrist, M. Fenton, J. Eger, and P. Schlegel, "What ears do for bats: a comparative study of pinna sound pressure transformation in chiroptera." *The Journal of experimental biology*, vol. 180, p. 119, 1993.

[22] U. Firzlaff and G. Schuller, "Spectral directionality of the external ear of the lesser spear-nosed bat, Phyllostomus discolor," *Hearing research*, vol. 181, no. 1-2, pp. 27–39, 2003.

[23] M. Aytekin, E. Grassi, M. Sahota, and C. Moss, "The bat head-related transfer function reveals binaural cues for sound localization in azimuth and elevation," *The Journal of the Acoustical Society of America*, vol. 116, p. 3594, 2004.

[24] J. Blauert, *Spatial hearing: the psychophysics of human sound localization*. The MIT Press, 1997.

[25] R. Patterson, I. Nimmo-Smith, J. Holdsworth, and P. Rice, "An efficient auditory filterbank based on the gammatone function," *APU report*, vol. 2341, 1987.

[26] P. Saillant, J. Simmons, S. Dear, and T. McMullen, "A computational model of echo processing and acoustic imaging in frequency-modulated echolocating bats: The spectrogram correlation and transformation receiver," *The Journal of the Acoustical Society of America*, vol. 94, p. 2691, 1993.

[27] M. Jensen, C. Moss, and A. Surlykke, "Echolocating bats can use acoustic landmarks for spatial orientation," *Journal of Experimental Biology*, vol. 208, no. 23, p. 4399, 2005.

[28] A. Pierce, *Acoustics: An introduction to its physical principles and applications*. Acoustical Society of Amer, 1989.

[29] F. Schillebeeckx, F. De Mey, D. Vanderelst, and H. Peremans, "Biomimetic sonar: Binaural 3d localization using artificial bat pinnae," *The International Journal of Robotics Research*, vol. 30, no. 8, pp. 975–987, 2011.

[30] C. Biber, S. Ellin, E. Shenk, and J. Stempeck, "The polaroid ultrasonic ranging system," *67th Convention of the Audio Engineering Society*, October 1980.

[31] *FG-23329-manual*, Knowles, http://www.knowles.com.

[32] J. Reijniers and H. Peremans, "Biomimetic sonar system performing spectrum-based localization," *Robotics, IEEE Transactions on*, vol. 23, no. 6, pp. 1151–1159, 2007.

[33] J. Steckel, J. Reijniers, A. Boen, and H. Peremans, "Biomimetic target localisation using an EMFi based array," in *Applications of Ferroelectrics, ISAF 2008.*, vol. 3. IEEE, 2008, pp. 1–2.

[34] M. Milford. (2011) Robot Navigation From Nature: Simultaneous Localization, Mapping, and Path Planning Based on Hippocampal Modelsl. [Online]. Available: http://ratslam.itee.uq.edu.au/

Regularized Linear Regression for Distance Estimation with an RGB-D Sensor

Evangelos Georgiou
Centre for Robotics
King's College London
London, United Kingdom
evangelos.georgiou@kcl.ac.uk

Professor Jian S. Dai
Centre for Robotics
King's College London
London, United Kingdom
jian.dai@kcl.ac.uk

Professor Michael Luck
Department of Informatics
King's College London
London, United Kingdom
michael.luck@kcl.ac.uk

***Abstract*— In the field of robotics, computer vision is becoming an essential component for robot navigation for path planning, obstacle avoidance and self-localization. A challenge in this field is dealing with high resolution data and adapting this data to position and orientation. This paper presents a regression model for adapting the inputs of an RGB-D sensor to an estimated distance, taking into account the multiple features of the image information. This machine learning heuristic model uses supervised learning methodology to change and adapt to data from a dynamic environment using an image with multiple parameter features.**

Machine Learning, Supervised Learning, Linear Regression, Regularization, RGB-D, Monocular Sensor, Non-holonomic Mobile Robot

I. Introduction

The regularized linear regression methodology has increasingly been a successful method with numerous applications. Regularized linear regression, also known as Lasso [1], was selected to help build a model for distance estimation that could deal with multiple features from an RGB-D image sensor. By constraining the norm of the coefficient vector, this method simultaneously avoids over-fitting to the training data and achieves sparsity in the obtained coefficients. The sparsity has two important benefits- it improves the interpretation by explicitly showing the relationship between the response and the features [1], and, at the same time, it is computationally efficient because it reduces the number of non-zero coefficients.

Regularization is also used for many other machine learning problems, like logistic regression [2], graphical model selection [3], and principal component analysis [4].

Developing an efficient algorithm that implements regularized linear regression is not a trivial task. Of the different approaches, Efron et al.'s [5] paper on Least Angle Regression (LARS) show a method where features are sequentially selected, ensuring that they are equiangular and that the coefficient paths are piecewise linear with respect to the regularization parameters. While it has the ability to discover the full regularization path, the LARS is modeled such that it selects one feature at a time, which makes its computation numerically expensive when an efficient method is required.

This paper sets out to present a distance estimation model using regularized linear regression. The model will be based on the features that will come from an image captured using an RGB-D sensor that is mounted on an autonomous non-holonomic mobile robot. The mobile robot requires an accurate model to be able to utilize the data for self-localization, path planning, and obstacle avoidance.

II. Experiment Platform

The sensor information is collected by a non-holonomic autonomous mobile robot that will use the distance estimation model for self-localization, path planning and obstacle avoidance.

A. *The KCLBOT: A Non-holonomic Mobile Robot*

The platform used to deliver the experiment is a mobile robot nicknamed the "KCLBOT" [6] [7] which was developed at King's College London.

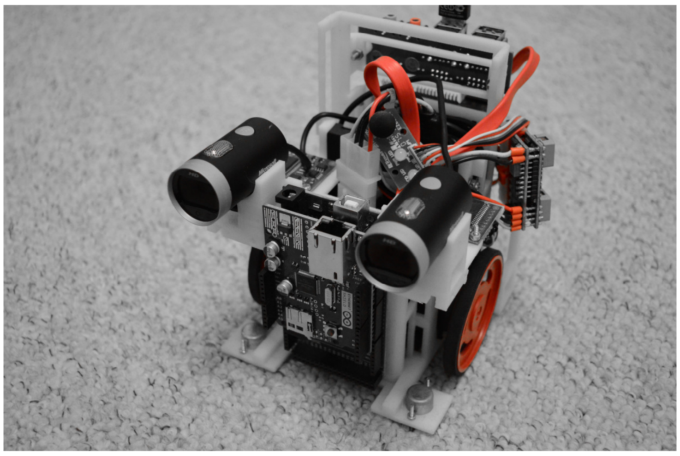


Figure 1. The KCLBOT – An Autonomous Mobile Robot

The mobile robot, seen in Figure 1, holds the imaging sensor used for the experiment. The robot has an onboard Pico-ITX computer [8], which interfaces the sensor with the software used to capture the images and stores the images for offline analysis.

B. *Non-holonomic Mobile Robot Dynamics*

The mobile robot behaves as a maneuverable [9] configured mobile robot and has the movement constraints such that the platform has to rotate to the desired direction angle before it translates to a desired goal position.

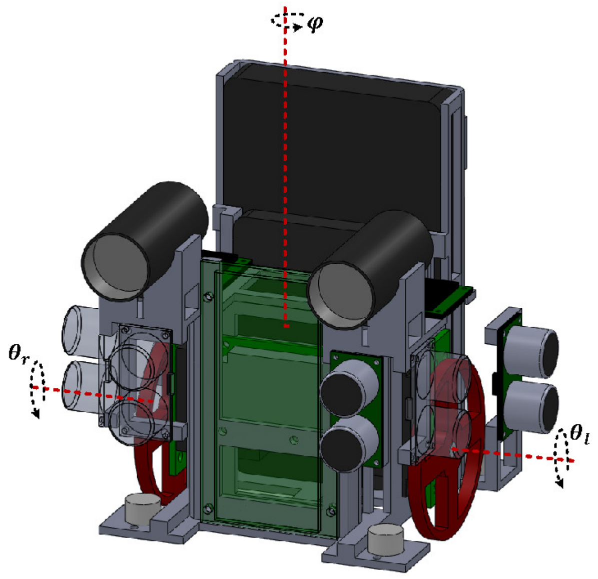


Figure 2. The KCLBOT – An Autonomous Mobile Robot

As the mobile robot is subject to slip generated by the rotating wheels and limited number of controllable degrees of freedom, the mobile robot is modeled as non-holonomic [10].

$$\dot{y}_c \cos(\phi) - \dot{x}_c \sin(\phi) - d\dot{\phi} = 0 \tag{1}$$

Where x_c and y_c are Cartesian-based coordinates of the mobile robot's center of mass, and ϕ describes the heading angle of the mobile robot, which is referenced from the global x-axis. The constraint equation (1) is holonomic.

$$\dot{y}_c \sin(\phi) + \dot{x}_c \cos(\phi) + L\dot{\phi} = r\dot{\theta}_r \tag{2}$$

$$\dot{y}_c \sin(\phi) + \dot{x}_c \cos(\phi) - L\dot{\phi} = r\dot{\theta}_l \tag{3}$$

Where $\dot{\theta}_r$ and $\dot{\theta}_l$ are the angular displacements of the right and left mobile robot wheels, respectively, and where r describes the radius of the mobile robot's driving wheels and L is the distance between the wheels. Equations (2) and (3) are non-holonomic.

While the holonomic (1) and the two non-holonomic equations, (2) and (3), do not directly affect the linear regression model, it is important to consider them as they affect the sensor's orientation and position.

III. Machine Learning with Linear Regression

The motivation for using a linear regression model for estimating distance based on a RGB-D sensor input is that it allows for the input of multiple features associated with the image capture. An example of the features that can be used is the exposure of a single shutter cycle, aperture size, lens focal length and ISO compensation.

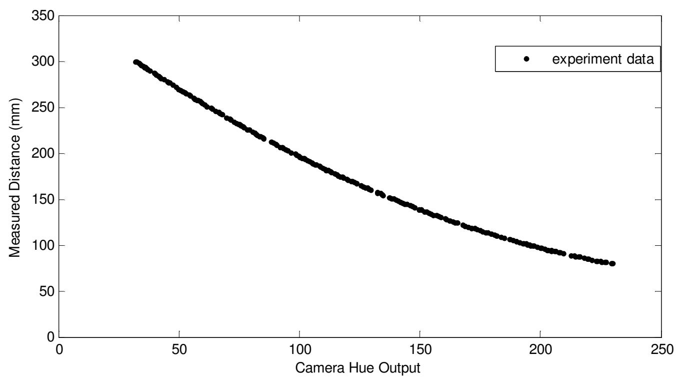


Figure 3. Fixed Object Capture - Experiment Data

Using the Microsoft Kinect RGB-D sensor [11], 220 image samples were taken of a fixed object between 800mm and 3,000mm. The fixed object was measure empirically assuming a tolerance of ±10mm. The results of the experiment are presented in Figure 3 above.

A. *The Cost Function*

The objective of utilizing linear regression is to achieve the minimum valuation of the cost function via a numerical gradient descent method. The cost function $J(\theta)$ is derived on the sum of squares for error (SSE) methodology, where the cost function $J(\theta)$ is described as follows:

$$J(\theta) = \frac{1}{2m}\sum_{t=1}^{m}(h_\theta(x^{(t)}) - y^{(t)})^2 \tag{4}$$

Where m describes the number of experiment samples, $h_\theta(x)$ describes the heuristic hypothesis of the behavior of the camera hue values mapped as x and y is the empirically measured distance values that correspond to the hue values mapped in x. The hypothesis $h_\theta(x)$ is constructed using a linear model and is represented as follows:

$$h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 \tag{5}$$

Where θ describes the parameters of the model, which will be manipulated to minimize the cost function $J(\theta)$.

B. *Implementing Gradient Descent*

A common method for solving θ from (5) is by utilizing a batch gradient descent algorithm, which is represented as follows:

$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{t=1}^{m}(h_\theta(x^{(t)}) - y^{(t)})x_j^{(t)} \tag{6}$$

Where θ_j is updated simultaneously for j values using an iterative approach to get θ_j to converge closer to the optimal value that will return the lowest cost from $J(\theta)$. The α scalar value is used as a learning rate to help converge θ_j to the lowest cost value. Typically, a small value is used to help the convergence process as a large value can affect getting to the local or global optimum value.

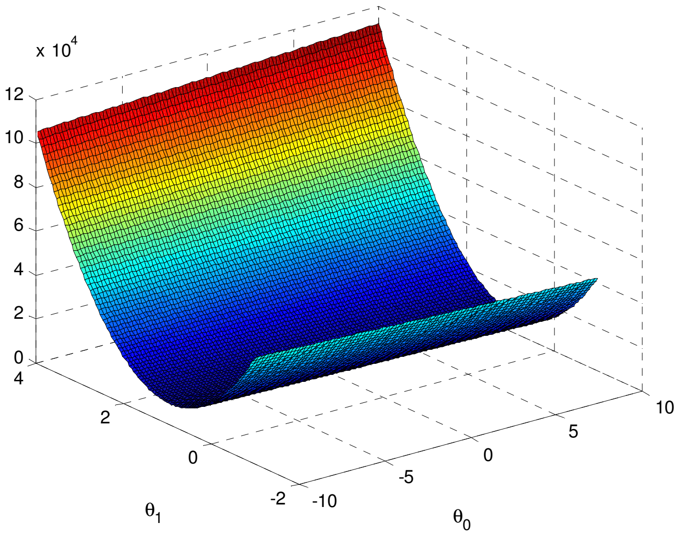


Figure 4. Gradient Descent Experiment Plot

With only two parameters, as in Figure 3, it is easy to visualize the cost function utilized by the gradient descent approach. When dealing with a higher order of parameters it is more difficult to visualize gradient descent.

C. *An Analytical Solution*

Having a cost function (4) and a gradient descent model (6) , it is possible to complete the hypothesis (5).

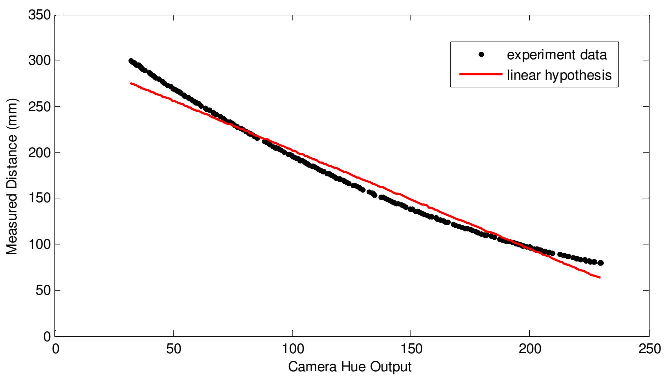


Figure 5. Linear Hypothesis for Experiement Data

From Figure 4, which is a plot of the hypothesis (5) compared to the experiment data, it can be observed that the hypothesis is under-fitting our experimental results. It would not be ideal to use this linear hypothesis to fit our experimental data and a polynomial based hypothesis should offer a more ideal fit to the available data.

$$h_\theta(x) = \theta_0 + \theta_1 x_i + \theta_2 x_i^2 + \ldots + \theta_m x_i^m \tag{7}$$

The hypothesis (7) presents a polynomial based equation, which will offer a more ideal fit to the experiment data. For this experiment, a second order polynomial is used to define the hypothesis. As the gradient descent approach is a very computation heavy method, to solve the θ parameters for the hypothesis it is more ideal to use a normal equation, which is the closed-form solution to the linear regression problem faced.

$$\theta = (X^T X)^{-1} X^T \vec{y} \tag{8}$$

Using the normal equation (8), the solutions to the parameters are computed and used to describe the polynomial based hypothesis.

$$h_\theta(x) = 359.26 - 1.9492.x + 0.0032.x^2 \tag{9}$$

The polynomial based hypothesis (9) provides an accuarate fit to the experiment data.

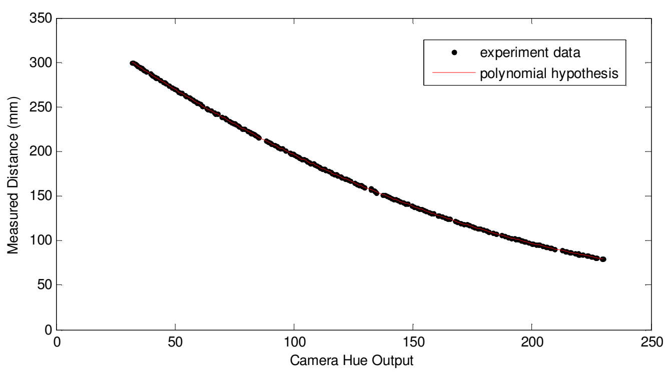


Figure 6. Polynomial Hypothesis for Experiement Data

To validate hypothesis (9), it is implemented in the cost function (4). This returns a negligible SSE. An alternative visual comparison is presented in Figure 5.

D. *Regularization*

In equation (5), only a single feature, hue, is considered, which is easily solved by an analytical solution (8), as presented. The image data from the sensor returns multiple features and, for this experiment, we will consider the color space values red, green and blue, the image hue, saturation and the V value. The hypothesis (5) will now consider the seven features of x_{0-6} and the linear regression gradient descent solver (6) for θ_{0-6}. For the gradient descent solver (6) to work, the numeric values of the features having difference ranges that need to normalized as follows:

$$x_i = \frac{x_i - \mu_i}{s_i} \tag{10}$$

Where μ describes the average value of the feature and s is the standard deviation of the feature. If the features are not normalized (10), features with high numeric values will hold a high weight when the solver is implemented.

To assist the gradient descent (6) solver with many features, the cost function (5) also needs to be manipulated.

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x)-y)^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right] \quad (11)$$

$$\theta_j := \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(t)})-y^{(t)})x_j^{(t)} - \frac{\lambda}{m}\theta_j\right] \quad (12)$$

Where λ is a constant used to control the compensation strength of the regularization introduced in equations (11) and (12). It should be noted that the new gradient descent model is not implemented on the initial θ_0 variable. The main reason for introducing regularization is to prevent a hypothesis model that over-fits the experiment data.
By implementing the gradient descent solver the following results are returned for θ:

TABLE I. GRADIENT DESCENT SOLUTIONS

θ_0	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
307.11	0.05	-0.127	0.0428	-1.171	-4.8	39.9296

With the θ values in Table I, the hypothesis is able to estimate distance based on the multiple features of the sensor image.

IV. EXPERIMENT STATISTICAL ANALYSIS

Having a hypothesis from the regularized gradient descent solver, in Table I, to validate the hypothesis, the error between the empirically measured values are compared to the output of the derived hypothesis.

A. Descriptive Statistics

Using an experiment sample size of 220 images with multiple features from the RGB-D sensor, the error difference is considered for the analysis.

TABLE II. STATISTICAL ANALYSIS OF EXPERIMETNAL ERROR DATA

Method	Evaluation
Mean	-0.004864
95% Confidence Interval for Mean (LB)	-0.106329
95% Confidence Interval for Mean (UB)	0.096601
5% Trimmed Mean	-0.031310
Median	-0.152314
Variance	0.583
Standard Deviation	0.7636119
Standard Error Mean	0.0514827
Minimum	-1.3223
Maximum	2.1117
Range	3.4340
Interquartile Range	1.0723
Skewness	0.534
Kurtosis	-0.411

The descriptive statistics from Table II give a general idea of the performance of the hypothesis. The mean error value is negative 0.005, which is relatively low. From the presented descriptive statistics, it is difficult to tell if the data is skewed and as such it is better to consider the median value which is negative 0.152, which is also comparably low. The confidence intervals, which are 2 standard deviation from the mean show a low error rate.

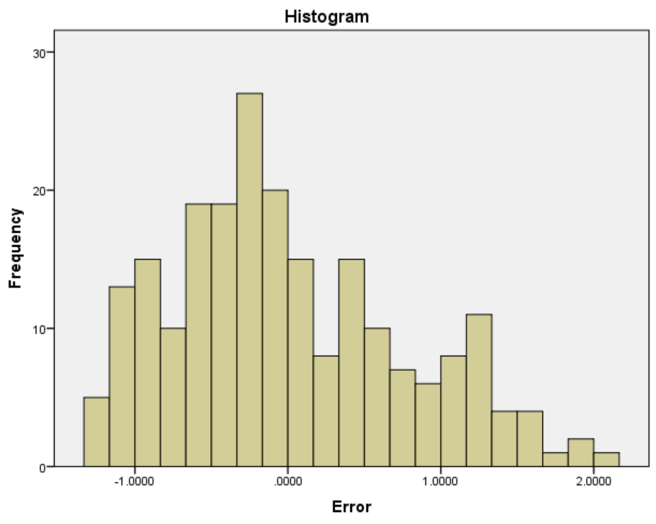


Figure 7. Histogram: Frequency vs Error

The histogram in Figure 7 shows that error is roughly Gaussian distributed between -1.5mm and 2mm, with the peak very to 0mm.

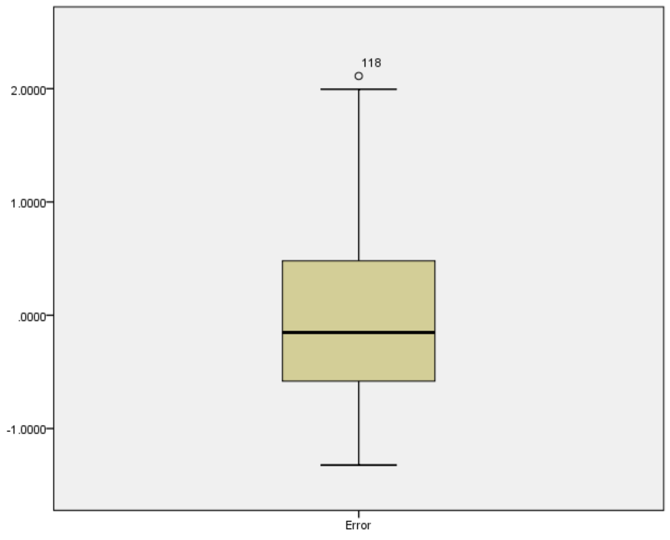


Figure 8. Boxplot of the Hypothesis Error

The boxplot in Figure 8 highlights very favorable results, presenting the mean value close to 0mm and very narrow interquartile range. The interquartile range is 1.0723mm and has a minimum value of -0.106mm and a maximum value of 0.097mm. An outlier is also present at data sample 118, which is ignored.

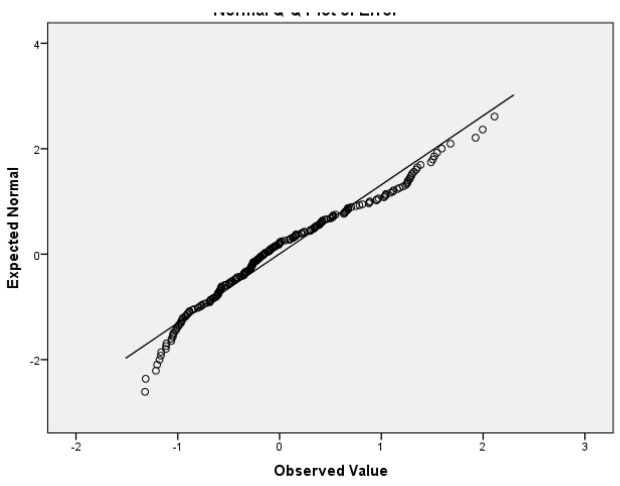


Figure 9. Normal Q-Q Plot of Error

The normal Q-Q plot shown in Figure 9 shows the performance of the observed values against the expected values. The observation is that errors normally distributed centrally, with anomalies at the tail ends.

B. *Experiment Validation*

Utilizing the descriptive statistics in Table II, a non-parametric test is required to validate the effectiveness of the hypothesis, from Table I, for the distance estimation model using multiple features. The ideal analysis test for a non-parametric independent one-sample set of data is the Kolmogorov-Smirnov test [12] for significance.

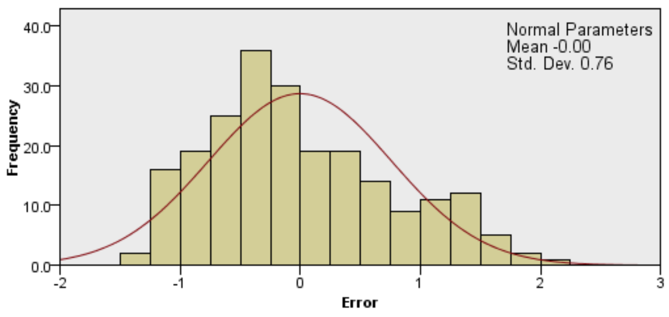


Figure 10. One Sample Kolmogorov-Smimov Test

For the experimental data presented in this paper, the null hypothesis is that the distribution of error is normal with a mean value of -0.005 and a standard deviation of 0.76. Based on the significance level of 0.05, a significance of 0.068 is returned using the one sample Kolmogorov-Smirnov test. The strength of the returned significance value allows us to retain the null hypothesis and say that the distribution of error is normal with a mean value of -0.004864 and a standard deviation of 0.7636119. The test results are presented in Figure 10.

V. CONCLUSION

This paper presents a regularized linear regression distance estimation model from the features that are collected from an RGB-D sensor image. The efficiency of the model is shown in the statistical analysis with mean error rate of -0.005. The model is versatile such that it can cope with additional features being added in the future, and because of the regularization of the model, fear of over-fitting will not be a concern.

REFERENCES

[1] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society Series B Methodological,* vol. 58, no. 1, pp. 267-288, 1996.
[2] H. L. S. Lee, P. Abbeel, A. Ng "Efficient L1 Regularized Logistic Regression."
[3] P. R. M. Wainwright, J. Lafferty, *High-dimensional graphical model selection using ℓ1-regularized logistic regression*: MIT Press, 2006.
[4] J. H. H. Shen, "Sparse principal component analysis via regularized low rank matrix approximation," *Journal of Multivariate Analysis,* vol. 99, pp. 1015 – 1034, 2008.
[5] T. H. B. Efron, I. Johnstone, R. Tibshirani, "Least angle regression," *The Annals of Statistics,* vol. 32, no. 2, pp. 407-499, 2004.
[6] E. Georgiou, Dai, J., Luck, M., "The KCLBOT: A Framework of the Nonholonomic Mobile Robot Platform Using Double Compass Self-Localisation " *Mobile Robots - Current Trends*, Z. Gacovski, ed.: Intech Open, 2011.
[7] E. Georgiou, J. Dai, and M. Luck, "The KCLBOT: Exploiting RGB-D Sensor Inputs for Navigation Environment Building and Mobile Robot Localization," *International Journal of Advanced Robotic Systems,* vol. 8, no. 4, pp. 194-202, Sep, 2011.
[8] Via Technologies Inc. "Artigo Pico-ITX," http://www.via.com.tw/en/initiatives/spearhead/pico-itx/index.jsp.
[9] G. Campion, Bastin, G., and D'Andrea-Novel, B. , "Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile Robots," *IEEE Transactions on Robotics and Automation,* vol. 12, no. 1, pp. 47–62, 1996.
[10] I. a. M. Kolmanovsky, N. , "Developments in nonholonomic control problems," *IEEE Control Systems Magazine*, pp. 20-36, 1995.
[11] "The PrimeSensor," http://www.primesense.com.
[12] G. Zhang, Wang, X., Liang, Y., and Li, J. , "Fast and Robust Spectrum Sensing via Kolmogorov-Smirnov Tes," *IEEE TRANSACTIONS ON COMMUNICATIONS,* vol. 58, no. 12, pp. 3410-3416, 2010.

CENTROALGORITMI

SPR SOCIEDADE PORTUGUESA DE ROBOTICA WWW.SPR.UA.PT

Universidade do Minho Escola de Engenharia Dep. Electrónica Industrial