# Reinforcement Learning for Linked Multicomponent Robotic Systems

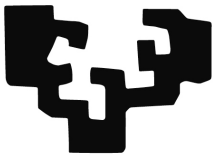
**By**

**José Manuel López Guede**

http://www.ehu.es/ccwintco

Dissertation presented to the Department of Computer Science and Artificial Intelligence in partial fulfillment of the requirements for the degree of

Doctor of Philosophy



*PhD Advisor:*

Prof. Manuel Graña Romay

*At*

The University of the Basque Country
Donostia - San Sebastián
2012

# Contents

# List of Algorithms

# List of Figures

# List of Tables

# Chapter 1

# Introduction

This introductory chapter provides the motivation for this Thesis in Section 1.1. Section 1.2 enumerates the objectives of the work. Section 1.3 highlights the contributions achieved by this Thesis, including the relevant publications in Section 1.3.1. Section 1.4 comments the structure of this dissertation. Finally, Section 1.5 summarizes some notation used in the dissertation.

## 1.1 Motivation

Linked Multi-Component Robotic System (L-MCRS) are a collection of autonomous mobile robots linked through a passive non-rigid physical element. The linking element is usually a one dimensional object, such as a hose or a cable. They can have industrial applications in many areas, from shipyards to building sites, where hoses are the common mean to tranport fluids and energy. Figure 1.1 shows an example of a shipyard where the manipulation of the hose that is being carried out can be modeled through this paradigm. In this dissertation, our objective is the study of ways to develop control systems for the robots attached to the hose in order to accomplish some specific task. However, we have reviewed the accurate modeling of such systems using an ad-hoc theoretical framework, the Generalized Dynamic Splines (GEDS). Such accurate modeling provide an appropriate work-bench for the development of control algorithms:

- its accuracy allows for validation of the ideas and results,
- it allows for the repetition of experiments, which is sometimes quite impor-

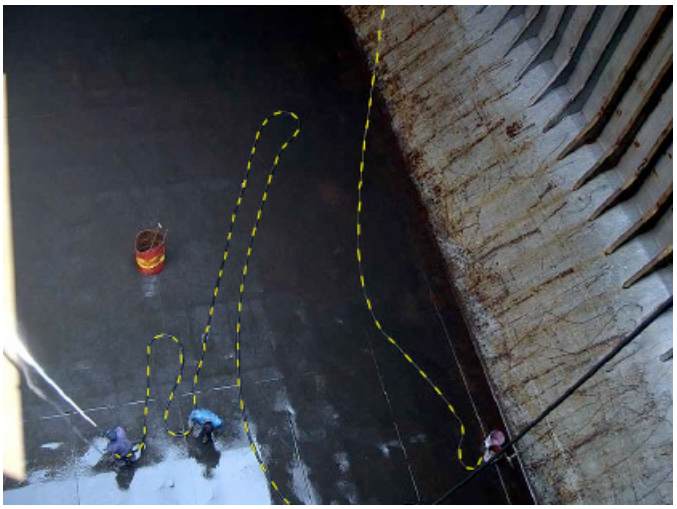


Figure 1.1: Real shypyard, where the L-MCRS paradigm is applicable

tant for the verificationof the correctness of algorithm implementaittions and to factor out some parameter effects, which can be marginalized in the simulations but are difficult to isolate in real life experimentation,

- it is faster than the real-life experiment,
- it avoids degradation of equipment. Degradation is an additional source of noise in the experiments, that can be difficult to identify.

We also recall some proof-of-concept physical realizations done in our research group's environment with a collection of mobile robots autonomously controlled performing the hose transportation along a linear trajectory, or along a path.

The most basic question about these systems is wether the linking element introduces some differential properties and behaviors that distinguish the L-MCRS from a group of disconnected robots. To answer this question we have produced the simplest L-MCRS model, where the linking element is modeled as a compresible spring. We propose a cooperative distributed control system for the mobile robots in the system assuming perfect knowledge of their positions, performing a simple path following task. The null hypothesis is that this cooperative distributed control system will perform identically with a linking element tying the otherwise autonomous robots. That is, the linking element has no effect on the system's control. The system's simulated task is that of

trasporting a hose-like object. The simulation results show that the linking element introduces specific dynamical effects, which the cooperative control can not cope with, so the perturbations in one component's behavior are propagated to the entire system.

The aim of this work is to develop autonomous learning processes to develop the control systems, obtaining

- Adaptative controllers, able to adapt to changing conditions or to a new setting without an accurate modeling of the system and the task to be performed.

- With minimal supervision, that is, they need little specification from the human operator.

Reinforcement Learning methods to allow the agents learn by themselves how to deal with the problem of developing a control system from little external information opposite to classical control schemes. Reinforcement Learning paradigm aims to develop algorithms that allow to train an agent to optimally achieve a goal with minimal feedback information about the desired behavior, which is not precisely specified. Scalar rewards are returned to the agent as response to its actions endorsing or opposing them. RL algorithms have been succesfully applied to robot control design. We select the famous Q-learning approach because it is model free and offers the bigger flexibility, at the price of the greatest computational (time and storage) demands. We have proposed a variation, TRQ-learning that speeds up the learning process by storing visited state transitions and perceived rewards.

We have resticted our works to a specific case, the transportation of the hose's tip to a desired position. The other end being attached to a source. The formulation of the learning system needs the definition of several elements: the configuration space, the simulation model precision, the state variables embodying the perception and prediction abilities of the system, the reward system of values. We have performed such definition for the case of a single robot and two robots moving the hose. Those are prototypical instances of the problem whose solution opens the field for further scenarios.

## 1.2 Objectives

The main objective of the Thesis is the application of Reinforcement Learning approaches to the autonomous development of control architectures for Linked Multi-Component Robotic Systems, in short for a hose manipulation system. This rather ambitious objective has been downscaled to a manageable dimension, defining a specific instance of the problem over which an extensive computational experimentation is affordable for the limited human resources of a Thesis. Specific objectives of the Thesis are the following:

1. Show the specific influence of the linking element in the system dynamics. This objective is a preliminar work to demonstrate the value of this reasearch work. It has some inherent sub-objectives:

   (a) Formulate a minimalist model that can show the intrinsic properties and influence of the linking element. The model minimalism is intended to factor out any additional source of variations in the system's behavior.

   (b) Formulate a control architecture for the disconnected system, without linking elements

   (c) Show experimentally, through simulation of the minimal model, the effect introduced by the linking element.

2. Define an approachable instance of the hose transportation problem that can be dealt with extensively.

3. Obtain a formulation of the Reinforcement Learning problem for an specifici instance of the hose transportation problem. Includes the definition of the system's state and other

4. Define an experimental workbench based on an accurate simulation of the hose transportation for the realization of Reinforcement Learning experience episodes and validation. This workbench is based on previous works of the research group members, requiring fine tuning of the implementation and the selection of appropriate parameters.

5. Perform an extensive experimentation testing the feasibility of Reinforcement Learning for the autonomous development of control architectures of the hose transportation system.

6. Propose alternative learning strategies, able to cope with the time complexity of the learning process in an optimized way.

## 1.3 Contributions of the Thesis

According to the objectives set in the previous section, the achievements of the Thesis reported in this dissertation, are summarized as follows:

1. A minimalistic hose transportation model has been proposed and implemented for simulation.

2. The effect of the linking element has been experimentally demostrated using the above minimalistic model. The effect is according to the intuitions guiding our research endeavors.

3. An accurate model of the hose dynamics has been fine tuned for its used in the Reinforcement Learning experimentation. An appropriate workbench has been build for this purpose and made publicly available through the research group's web site.

4. The specific instance defined is a hose-deployment task, where one end of the hose is attached to a source (power/fluid) and the robots must bring the tip of the hose to a specific position.

5. Alternative state and reward definitions have been identified and formulated as the basis for the Reinforcement Learning.

6. A sound methodological approach has been followed, where accuracy results are provided on the basis of test episodes performed independently of the learning process. For intermediate results, the learning process is stopped and the test is carried out on the basis of the present status of the learned Q-tables.

7. Extensive experimentations have been carried out to validate the alternative approaches using the accurate simulation workbench. High success measured by the number of test episodes reaching the goal state has been obtained for specific combinations of state definition and reward system.

8. A variation of Q-learning, denoted TRQ-learning, has been formulated and tested obtaining faster learning processes.

### 1.3.1 Publications

- J. M. Lopez-Guede, M. Graña, and E. Zulueta, "On distributed cooperative control for the manipulation of a hose by a multirobot system" in Hybrid Articial Intelligence Systems, ser. Lecture Notes in Computer Science, E. Corchado, A. Abraham, and W. Pedrycz, Eds. Springer Berlin / Heidelberg, 2008, vol. 5271, pp. 673-679, ISBN: 978-3-540-87655-7

- J. M. Lopez-Guede, M. Graña, E. Zulueta and Oscar Barambones, "Economical Implementation of Control Loops for Multi-Robot Systems" in Proc. of 15th International Conference on Neural Information Processing of the Asia-Pacific Neural Network Assembly (ICONIP 2008). Advances in Neuro-Information Processing. Eds. Springer Verlag 2009, vol. 5506, pp. 1053-1059, ISBN: 978-3-642-02489-4

- B. Fernandez-Gauna, J. M. Lopez-Guede, and E. Zulueta, "Linked multicomponent robotic systems: Basic assessment of linking element dynamical effect" in Hybrid Articial Intelligence Systems, ser. Lecture Notes in Computer Science, M. Graña Romay, E. Corchado, and M. Garcia Sebastian, Eds. Springer Berlin / Heidelberg, 2010, vol. 6076, pp. 73-79, ISBN: 978-3-642-13768-6

- J. M. Lopez-Guede, E. Zulueta, B. Fernandez and M. Graña, "Multi-Robot Systems Control Implementation" in Robot Learning, Suraiya Jabin (Ed.), pp.137-150. Sciyo 2010. ISBN 978-953-307-104-6

- B. Fernandez-Gauna, J. M. Lopez-Guede, E. Zulueta, and M. Graña, "Learning hose transport control with q-learning", Neural Network World, vol. 20, no. 7, SI, pp. 913-923, 2010, ISSN: 1210-0552

- B. Fernandez-Gauna, J. M. Lopez-Guede, E. Zulueta, Z. Echegoyen, and M. Graña, "Basic results and experiments on robotic multi-agent system for hose deployment and transportation", International Journal of Articial Intelligence, vol. 6, no. S11, pp. 183-202, March 2011, ISSN: 0974-0635

- M. Graña, B. Fernandez-Gauna, and J. M. Lopez-Guede, "Cooperative multi-agent reinforcement learning for multi-component robotic systems: guidelines for future research", Paladyn, Journal of Behavioral Robotics, pp. 1-11, 2011, 10.2478/s13230-011-0017-5. [Online]. Available: http://dx.doi.org/10.2478/s13230-011-0017-5, ISSN: 2080-9778

- J. M. Lopez-Guede, B. Fernandez-Gauna, M. Graña, and E. Zulueta, "Empirical study of q-learning based elemental hose transport control" in Hybrid Articial Intelligent Systems, ser. Lecture Notes in Computer Science, E. Corchado, M. Kurzynski, and M. Wozniak, Eds. Springer Berlin / Heidelberg, 2011, vol. 6679, pp. 455-462, ISBN: 978-3-642-21218-5

- B. Fernandez-Gauna, J. M. Lopez-Guede, and M. Graña, "Towards concurrent q-learning on linked multi-component robotic systems" in Hybrid Articial Intelligent Systems, ser. Lecture Notes in Computer Science, E. Corchado, M. Kurzynski, and M. Wozniak, Eds. Springer Berlin / Heidelberg, 2011, vol. 6679, pp. 463-470, ISBN: 978-3-642-21218-5

- B. Fernandez-Gauna, J. M. Lopez-Guede, and M. Graña, "Concurrent modular q-learning with local rewards on linked multicomponent robotic systems" in Foundations on Natural and Articial Computation, ser. Lecture Notes in Computer Science, J. Ferrández, J. Alvarez Sánchez, F. de la Paz, and F. Toledo, Eds. Springer Berlin / Heidelberg, 2011, vol. 6686, pp. 148-155, ISBN: 978-3-642-21343-4

- J. M. Lopez-Guede, B. Fernandez-Gauna, M. Graña, and E. Zulueta. "Further Results Learning Hose Transport Control with Q-learning", JoPhA, Journal of Physical Agents, 2011, ISSN: 1888-0258 (accepted, in press)

- J. M. Lopez-Guede, B. Fernandez-Gauna, M. Graña, and E. Zulueta, "TRQ-learning: Improving the control of single robot hose transport", Cybernetics and Systems, 2011, ISSN: 0196-9722 (accepted, in press)

- J. M. Lopez-Guede, B. Fernandez-Gauna, M. Graña, and E. Zulueta. "Visual Detection in Linked Multi-Component Robotic System", Robotic Vision: Technologies for Machine Learning and Vision Applications. Ed. José García Rodríguez and Miguel Cazorla. IGI Global. 2011 (accepted, in press)

## 1.4 Structure of the dissertation

Chapter 2 contains a detailed description of the target system, including a formal model used for the accurate simulations done for the computational learning experiments. The description in this chapter is the physical basis for the

remaining chapters of the dissertation, fixing also some of the objectives of the work.

Chapter 3 contains a distributed approach to the consensus based control of the system. The chapter contains a simplified model of the system described in Chapter 2, which is enough for the purposes of the chapter. The experimental works show that a control system designed for the team of independent robots can not cope with the effects introduced by the linking element, which introduces perturbations that propagate through the system.

Chapter 4 provides a review of the Reinforcement Learning methods that are applied to the autonomous learning of the control of the system based on the experience obtained from the accurate simulation of the model described in Chapter 2.

Chapter 5 gives the computational experiment results obtained on the prototypical system with one and two robots moving the tip of the hose to a desired position while the other end is attached to a source point.

Chapter 6 gives the conclusions of the Thesis and some ideas for future research.

## 1.5 Notation

| | |
|---|---|
| $X$ | set of all state variables |
| $S$ | set of all posible states of the state space, defined as the cartesian product of the range of values of the state variables $X$, such that $S = S^G \cup S^F \cup S^I$ |
| $S^G$ | subset of states where the goal has been reached, such that $S^G \subset S$ |
| $S^F$ | subset of states where a failure or a forbidden situation occurs, such that $S^F \subset S$ |
| $S^I$ | subset of inconclusive states, such that $S^I \subset S$ and $S^I = S - S^G - S^F$ |
| $A$ | set of all available actions |
| $A_s$ | set of all available actions in the state $s$ |
| $T$ | state transition function, such that $T: S \times A \to S$ or $T: S \times A \times S \to \mathbb{R}$ |
| $R$ | reward function, such that $R: S \times A \to \mathbb{R}$ |

| | |
|---|---|
| $\|S\|$ | number of elements of a set $S$ |
| $t$ | discrete time step |
| $s$, $s_t$ | state, usually at time $t$, such that $s \in S$ and $s_t \in S$ |
| $s'$, $s_{t+1}$ | state, usually at time $t+1$, such that $s' \in S$ and $s_{t+1} \in S$ |
| $a$, $a_t$ | action, usually at time $t$, such that $a \in A$ and $a_t \in A$ |
| $r$, $r_t$, $r(s)$ | immediate reward, usually at time $t$, such that $r \in \mathbb{R}$ and $s \in S$ |
| $r_g$ | immediate reward of a goal state, such that $r_g \in \mathbb{R}$ |
| $r_f$ | immediate reward of a goal state, such that $r_f \in \mathbb{R}$ |
| $r_i$ | immediate reward of a goal state, such that $r_i \in \mathbb{R}$ |
| $\mathbf{p}_r$ | position of the robot $r$ in two-dimensional coordinates $(x, y)$ |
| $\mathbf{p}_g$ | goal position in two-dimensional coordinates $(x, y)$ |
| $d(\mathbf{p}_1, \mathbf{p}_2)$ | euclidean distance between the two-dimensional points $\mathbf{p}_1$ and $\mathbf{p}_2$ |
| $i$ | binary value that is set if the line $\overline{\mathbf{p}_r\mathbf{p}_g}$ intersects the hose, such that $i \in \{0, 1\}$ |
| $c$ | binary value that is set if the box with corners $\mathbf{p}_r$ and $\mathbf{p}_d$ intersects the hose, such that $c \in \{0, 1\}$ |
| $\mathbf{p}_1$, $\mathbf{p}_2$ | positions in two-dimensional coordinates $(x, y)$ |
| $\mathbf{V}_r$ | vector of binary values, one for each feasible action, that is set if there will be collision after performing the corresponding action |
| $V^*(s)$ | state value function, value of state $s \in S$ an optimal policy, such that $V^* : S \to \mathbb{R}$ |
| $V^\pi(s)$ | state value function, value of state $s \in S$ under a policy $\pi$, such that $V^\pi : S \to \mathbb{R}$ |
| $V$ | estimated of a state value function $V^*(s)$ or $V^\pi(s)$, such that $V : S \to \mathbb{R}$ |
| $Q^*(s, a)$ | state-action value function, value of taking $a \in A$ in $s \in S$ under an optimal policy, such that $Q^* : S \times A \to \mathbb{R}$ |

| | |
|---|---|
| $Q^{\pi}(s, a)$ | state-action value function, value of taking $a \in A$ in $s \in S$ under a policy $\pi$, such that $Q^{\pi} : S \times A \to \mathbb{R}$ |
| $Q(s, a)$ | estimates of a state-action value function $Q^{*}(s, a)$ or $Q^{\pi}(s, a)$, such that $Q : S \times A \to \mathbb{R}$ |
| $\pi$ | policy, such that $\pi : S \to A$ or $\pi : S \times A \to \mathbb{R}$ |
| $\pi^{*}$ | optimal policy maximizing the state value function or the state-action value function |
| $\pi_{\epsilon}(s, Q)$ | $\epsilon$-greedy policy returning the best action available in state $s \in S$ according to the current estimates $Q$ of the optimal state-action value function $Q^{*}$ |
| $\alpha$ | step-size parameter that determines how new and old information are averaged, such that $0 < \alpha \leq 1$ |
| $\gamma$ | discount rate paremeter that determines the importance if future rewards, such that $0 < \gamma \leq 1$ |
| $\epsilon$ | probability of choosing a random action in $\epsilon$-greedy policy, such that $0 \leq \epsilon \leq 1$ |

# Chapter 2

# The hose transportation problem

In this chapter we introduce the general problem attacked in the Thesis, a collection of mobile robots moving a one-dimensional object, which corresponds to a hose in many potential applications. Section 2.1 gives a general introduction. Section 2.2 provides description of some physical systems implemented to provide proofs of concept of the intrinsic issues of hose transportation. Section 2.3 provides the formal description of the model used in the more accurate simulations.

## 2.1 Introduction

Multi-Component Robotic Systems (MCRS) [9] are currently the focus of great scientific interest because they are expected to provide solutions to the growing set of applications that require groups of autonomous robots able to dynamically adapt to changing environments and act in a coordinated way to perform tasks that could not be performed by a single robot, or performing them in a more efficient or economical way. Examples of such tasks are cooperative mapping of an environment [**?**], establishing dynamic communication links, and driving a hose to a goal [10, **?**]. Among the desired properties of a MCRS control algorithm, the most salient are:

- Resource scalability. Applications may require teams of robots of different sizes depending on task specific parameters. The addition of new robots

must not degrade the operation of the whole system, implying that the resources (memory and communication bandwidth) used by the control algorithm must not grow exponentially.

- Automating system design. It is not desirable to rely the success of a system on the expertise of the designer and it is preferable to develop tools that can automatically tailor the system to new complex environments. Automatic decomposition of complex tasks allows the definition of the robot team coordination as workload distribution. .

- Heterogeneity. MCRS applications may include heterogeneous groups of robots with different capabilities, including both sensors and actuators. Thus, control algorithms should be able to deal with heterogeneous inputs and outputs, and also minimize the impact of less performing robot individuals on the whole system's performance.

- Decentralized control to obtain higher fault-tolerance than with centralized control. Decentralized control improves scalability because control complexity grows linearly with the number of robots.

- Accurate control. Some mechanism is required to compensate for the inherent delay introduced by the sensory-devices, control algorithm and communications.

- Robustness to partial and noisy sensor data. In complex and noisy environments it is a requirement that agents are able to carry on their tasks even if they only have inaccurate and partial knowledge of the environment.

- Reasonable development time. Designing and fine-tuning the control algorithm should require an affordable amount of time.

Often, developing control algorithms for MCRS cannot be approached analytically. Sometimes there is not even a proper model for the system to be controlled and, even when such a model is available, system complexity hind their analytical resolution. As an alternative to traditional control theory, Artificial Intelligence techniques have been explored to provide robotic systems with tools that enable them to learn by interacting with the environment, which can be either the real world or a simulated one.

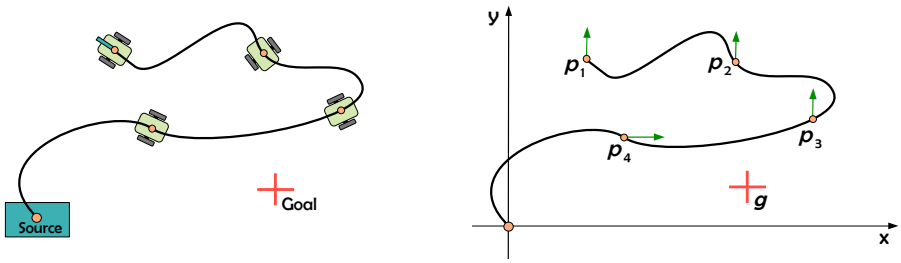
It is possible to distinguish three basic categories of MCRS:

- Distributed MCRS (D-MCRS), which are groups of uncoupled robots.
- Linked MCRS (L-MCRS) are defined as a collection of autonomous robots linked by a non-rigid physical link.
- Modular MCRS (M-MCRS), which are rigidly linked modular robots.

Modeling the non-rigid links of L-MCRS is a non-trivial issue, but one which is critical in order to be able to study those systems either analytically or via simulation. Some dynamic modeling techniques for non-rigid uni-dimensional objects are reviewed in [15, 10]: differential equations[32], rigid element chains[18], spring mass systems[17], combining spline geometrical models and physical dynamical models[33], and spline models combined with the Cosserat rod theory [40], also known as Geometrically Exact Dynamic Splines (GEDS). Throughout this Thesis, we will use GEDS, as we believe it is the most adequate model for uni-dimensional objects.

The study of L-MCRS is a novel research and no relevant information about this subject can be found in the literature, besides our group own publications. We are currently focused on the accurate modeling of these systems and development of control algorithms. We have started dealing with control and modeling of these systems in [15, 10] and [13]. The latter showed that even a simple spring model of the physical-link in a cooperative control problem introduces highly non-linear dynamics in the system, making it a hard task to derive the control commands. The paradigm of L-MCRS is exemplified by the task of carrying a hose from the origin point to a predefined destination using a collection of autonomous robots attached to the hose. Because of the inherent complexity of robot dynamics, further increased by the complex model of the hose, the system is not solvable in an analytic fashion.

One of our first works is [?], where the problem is addressed under the point of view of cooperative control [35], i.e., there the cooperation constraint, the cooperation objective and the coordination function are specified. Besides, we discussed how to solve the problem with total information (i.e., a centralized system) and with a decentralized system too. However, this is a speculative theoretical work without any implementation, so it does not include any concrete results. In [?], still within the framework of cooperative control, a computational model for L-MCRS based on elastic traction forces is used providing some results are obtained through simulations, for a task which consisted in the tracking of a predefined path. Consensus-based methodologies to approach the




(a) (b)

Figure 2.1: Hose transportation. Visual representation of (a) the real system, and (b) the model, arrows represent the forces exerted by the robots on the hose.

distributed multi-vehicle cooperation problem are addressed too. Our last work within the framework of cooperative control is [?], that extends the results of [?] and explores the capabilities of heuristic control when the task of following a predefined path is carried out. Heuristic control is also used in [15, 10], where the cooperative control has not been used.

We have followed an approach based on Reinforcement Learning (RL) [38] for the autonomous learning of optimal control policies. RL techniques have been used successfully in several areas of robotics, as navigation [6], indoor navigation [5], cooperative navigation task [27] and automatic path search [28].

### 2.1.1 Driving the hose tip

An specific hose transportation problem is illustrated in Figure 2.1: a set of mobile robots attached to a hose is required to transport the tip of the hose to some predefined goal position, the other end of the hose is attached to a fixed position, a source of some fluid. The mobile robots' actions consist in their motion, affecting the configuration of the hose which is a passive linking element introducing non-linear dynamical effects and additional physical constraints in the system. The goal of the control algorithm is for robot in $p_1$ to reach $g$, which is the predefined goal position. Figure 2.1(b) illustrates the forces exerted by the robots as arrows raising from the hose and pointing in the robot motion directions. This is the problem that has been specifically dealt with in this Thesis applying reinforcement learning to the autonomous design of the robot controllers.

## 2.2 Proof-of-concept physical instances

In order to perform proof-of-concept physical realizations our group has gone to the very basic problems. The first one, reported in [10, 43], is a robotic system controlled by a heuristic algorithm performing the transportation of the hose in a straight line in an environment without obstacles from an initial arbitrary configuration of hose and robots. The second is the control of a similar robotic team to follow the reference positions along a path, under proportional controls unaware of the linking elements.

### 2.2.1 Following a line

Although the task is the simplest one that can be defined, it is still a non-trivial task which poses several problems that may be pervasive in any attempt to build more sophisticated system. Besides, going from the theoretical analysis and simulations of previous section to the real life implementation means that we have to deal with the embodiment problem: in a physical implementation the real robots and other systems that are being used impose restrictions and conditions that must be coped with in order to obtain a working system realizing the proposed task. The hose transportation following a line may have rather diverse solutions depending on the embodiment of the task, that is, the actual robots employed and the actual physical features of the passive element. The experimental system consists of three small SR1 robots carrying an electrical cable of 1 cm. of diameter which plays the role of the linking element. Each robot was attached to the cable by means of a bearing which allows the robot to rotate freely under it.

#### 2.2.1.1 Perception

The centralized control system's perception was based on a zenital camera whose field of view is a 3 meters length floor region covering the scene encompassing the three robots and the hose. Image segmentation is based on computing a Specular Free (SF) image [39]. For robot segmentation adn identification, image preprocessing consists in substracting the minimum color value from the maximum color at each pixel independently. The RGB image is then transformed to HSV space and its intensity channel is replaced with the computed cromatic image $C$, so that white/gray pixels become black ones. This HSV image is then transformed back to RGB space, where it can be easily segmented

because the robots are painted of a saturated blue color. The hose is easily segmented because it is a dark object over a light background. This global peception produces a relative localization of the robots and the hose, but not absolute coordinates in the world's system. All the control decisions are based on qualitative characterizations of the hose and robot positions.

#### 2.2.1.2 Control heuristic

The robotic platform has limited computing capabilities, therefore the algorithms to decide the control commands are computed in a separate single computer and then communicated to the robots. However, each action of the individual robots is computed independently, without taking into account the state of the other robots, as if they were computed by each of the independent agents. Each robot's control will be determined only by the perception of the segment of the hose that is immediately ahead of him and the information about the orientation of the robot leader, and its own orientation. Trailing robots will try to have the same orientation as the leader, which is remotely controlled. The system's scalability allows the extension to any number of robots.

Each robot's speed is given by a control heuristic that takes into account the state of the hose segment ahead of it. This state is a function of the curvature of the hose segment. If two robots are too close, the hose segment between them will fold and increase its curvature, with the risk of forming loops. The robot at the rear of such a hose segment must reduce its speed. On the other hand, if the two robots attached to the hose segment are separated enough the hose segment will be very close to the straight line. The hose folding is proportional to the ratio of the sides of the rectangle that encloses the hose segment in the image shown in figure 2.2.

#### 2.2.1.3 Experiment realization

In figure 2.3 an example[1] of the realization of the hose transportation task defined above is shown. A human operator is controlling the head robot, giving it the starting order. In each frame, the robots are marked with their state (advancing, stretching, shrinking) and the curvature of their respective hose segment. Detected hose segments were marked in red in the original colored

[1] Some additional videos can be found at the research group web site:
http://www.ehu.es/ccwintco/index.php/DPI2006-15346-C03-03-Resultados-videos-control-centralizado

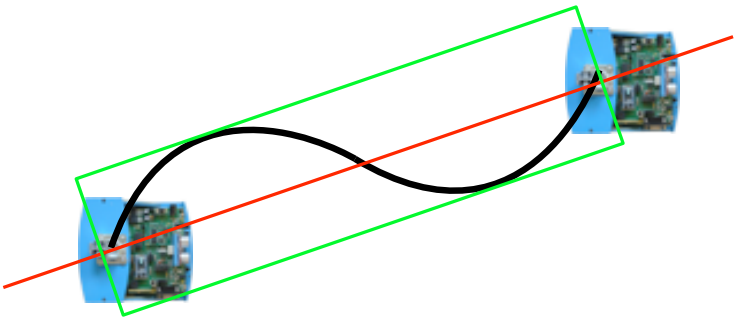


Figure 2.2: Hose state calculation. The proportion between the width and length of the box will be the measure of the state of the hose segment.

video. The six frames are extracted from the video generated in the test and shows how the system reacts to the different states of the hose:

- Figure 2.3a: Starting position. The leader starts towing the hose, while the 2nd and 3th robots wait for it to stretch enough to start moving.
- Figure 2.3b: The first segment's curvature falls below straight threshold. The 2nd robot starts advancing at cruise speed. The 3th robot keeps waiting .
- Figure 2.3c: The first segment is too stretched. The 2nd robot accelerates to a fast speed regime to shrink it. The 3th robot keeps waiting.
- Figure 2.3d: The first segment's curvature is within cruising limits, the 2nd robot brakes itself to attain cruise speed. Second segment's curvature also enters within cruising limits, therefore the 3th robot starts advancing at cruise speed.
- Figure 2.3e: Second segment curvature raises again above straight threshold, the 3th robot stops. The 2nd robot keeps advancing at cruise speed.
- Figure 2.3f: Second segment falls below stright limits , the 3th robot accelerates to fast speed. The 2nd robot keeps advancing at cruise speed.

Unsucessful repetitions of the experiment show a variety of effects that hind the system to accomplish its task. Sometimes the rear robot loses completely its orientation because the cable rigidity can not be overcome by the wheel servos. Other common effect is that the cable bending impedes the middle robot to advance. All these effects are produced by the interaction of the linking element elastic and inertial forces with the dynamics of the three robot system.

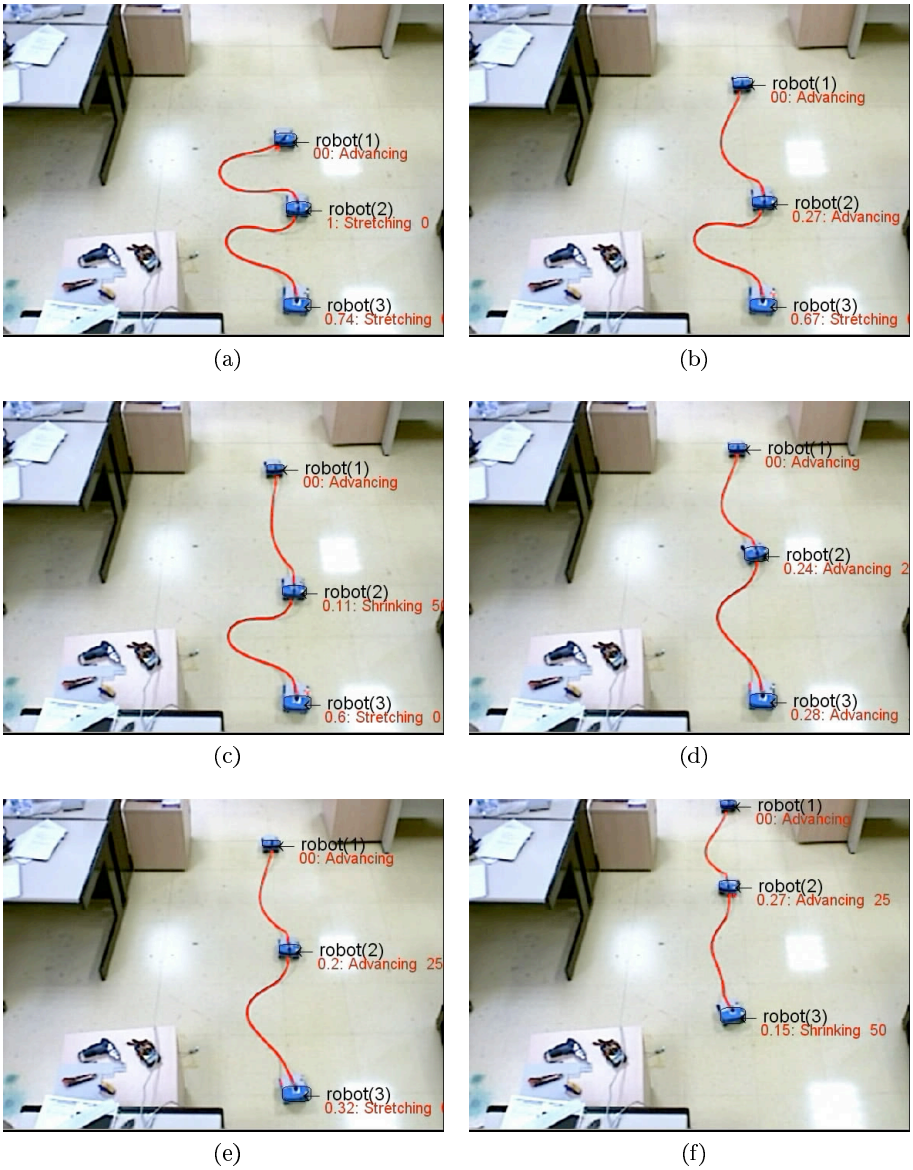



Figure 2.3: Frames extracted from the video of an example realization of the hose transportation task following a straight line.

### 2.2.2 Following a path

The second proof-of-concept experiment consists in an independent realization of the centralized perception and control system. Again the number of SR1 robots is three, but the linking element is an almost weightless cable that does not have strong inertial or elastic forces. The linking element effect is some kind of non-linear constraint that does not allow the robots to spread beyond some spatial limits given by the size of the hose segments. However, such a simple physical model still introduces strong effects. In fact, the aim of this experiment is to show that a conventional centralized control unaware of the linking element could have a hard time trying to comply with the task of moving the robots along the predefined path.

#### 2.2.2.1 Perception

The monitoring camera is positioned in a zenithal location covering the experimental field. A reference object, a chessboard pattern, is positioned in a predefined position to obtain absolute world coodinates via a camera calibration based on the pattern. The robot telltales are blue paper triangles adhered to the top of the robots. The blue color is saturated and easily segmented by standard color segmentation routines provided by OpenCV. The triangles provide location and orientation information, because they point to the front of the robot. The perceived coordinates of the triangle middle point are transferred to the control module. No effort is made to segment the linking element, though it is black and relatively easy to detect.

#### 2.2.2.2 Control system

The control of each SR1 robot is a straightforward PID controller that tries to move the robot according to a given reference position. The abstract reference position moves along a path, while the robot following it can wander from the reference path due to a multitude of effects, such as communication noise or error, servos inefficiency or wheel slippage. In fact, the unaccounted for linking element will introduce some undesired effects. The robots have three speed values: zero, half and full speed. They can also be switched off, meaning that the communication of control commands to the robot is cut off.

#### 2.2.2.3 Experiment instances.

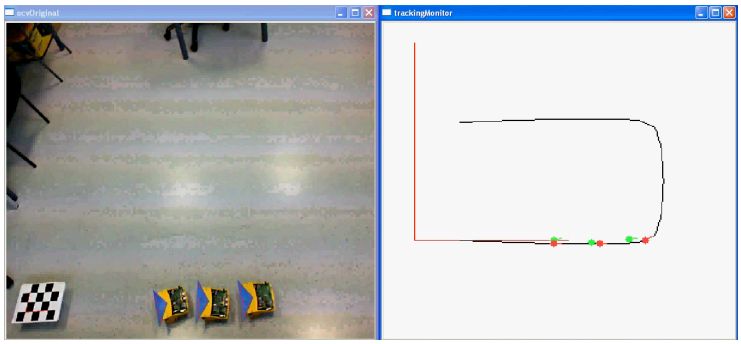
The displays in the figures have two simultaneous screen-shots. Left is the actual image camera, right is the virtual world showing the predefined path (black line), the actual robot position (green dot with a line for orientation), and the reference position that must be reached at this time by the robot (red dot with orientation line).

The first experiment is the unlinked system of the three robots following the predefined path. Figure 2.4 shows some snapshots of the system at the beginning of the experiment, at the middle trajectory and the near the final positions. It can be appreciated tha the robots follow easily and accurately the desired positions as they move along the path.

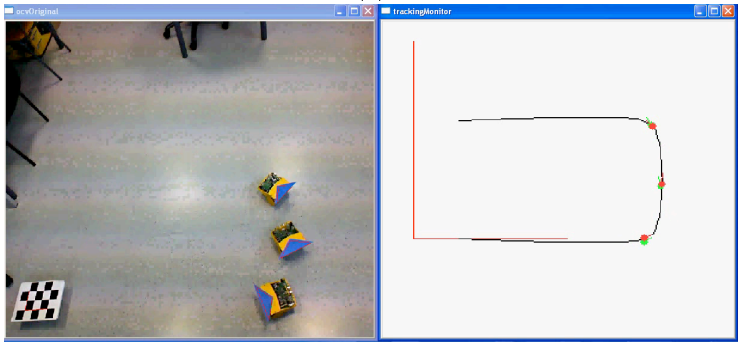
The second experiment introduces the link between the robots. Its effect is that trajectory wanderings of the leader robot are propagated to the other robots, which toward the end are mispositioned. The tracking introduces additional inertia on the robots as well as some kind of oscillations unseen in the unlinked system. Figure 2.5 shows three snapshots of a typical run of this experiment. It can be appreciated that the system becomes unstable at the middle of the trajectory, which towards the end shows a poor positioning of the robots relative to the reference positions, with eccentric orientations. The hose has been propagating positioning and orientation errors of the forward robots to the rear robots, while at the same time introducing some kind of inertia on the forward robots due to their link to the rear robot, impeding its course corrections.

The third experiment includes the effect of the tail robot, the third robot, is dragging the other robots moving more slowly than them. The first effect is that the robots are unable to follow their references. Moreover, the middle robot is moved out of the path, taking a "shortcut". Finally, the first robot is also displaced from the desired trajectory as a consequence to its reaction to the dragging of the rear robots. These effects can be appreciated in figure 2.6.

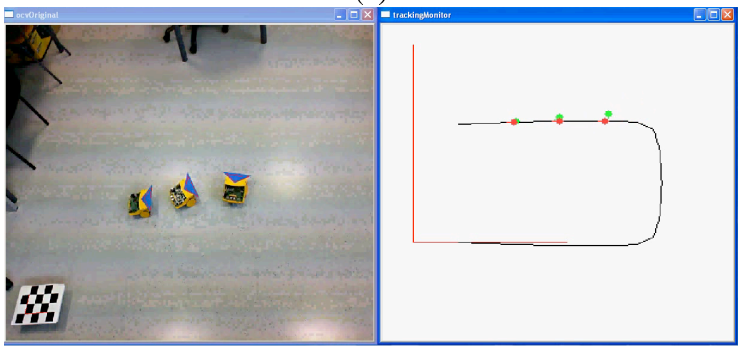
The last experiment is illustrated in figure 2.7. The last robot is switched off simulating a communication failure. The effect is catastrophic, the robots lag from their references since the begining. They follow from far the desired trajectory and never recover from the lag. However, there is some hint towards robustness, because the last robot has been moving, albeit far from the correct reference position. In an unlinked system, the last robot would have remained stuck at the initial position.




(a)

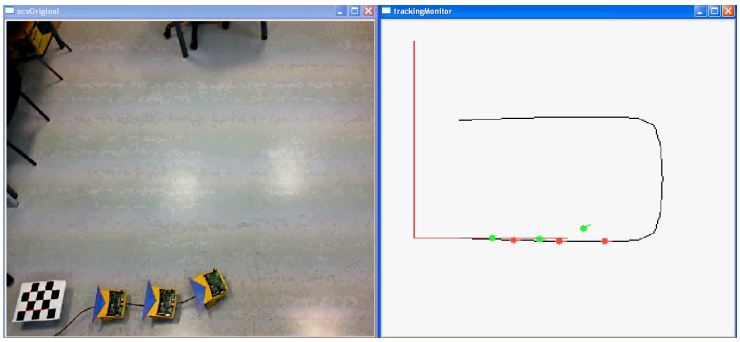



(b)

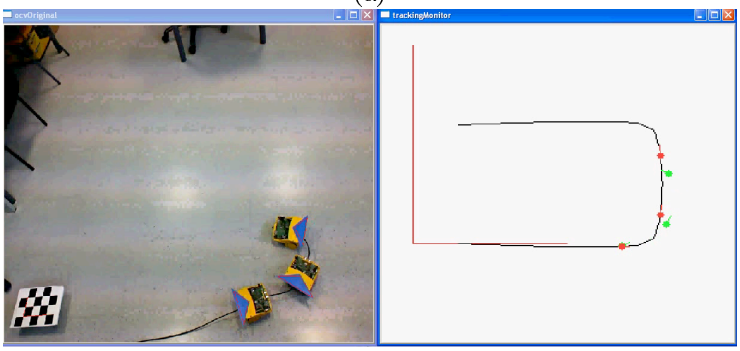



(c)

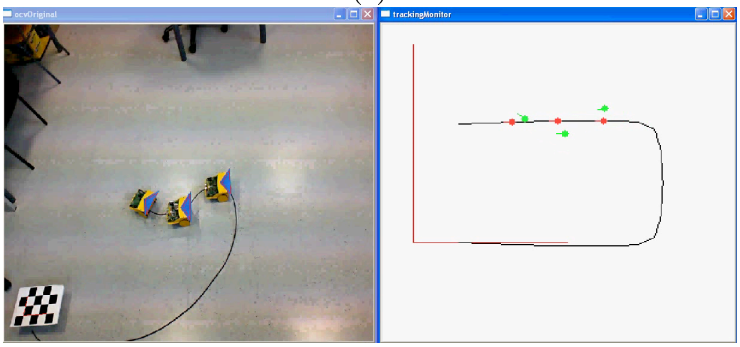
Figure 2.4: Unconstrained path following experiment. Snapshots at three time instants: (a) begining, (b) middle trajectory, (c) trajectory end.




(a)

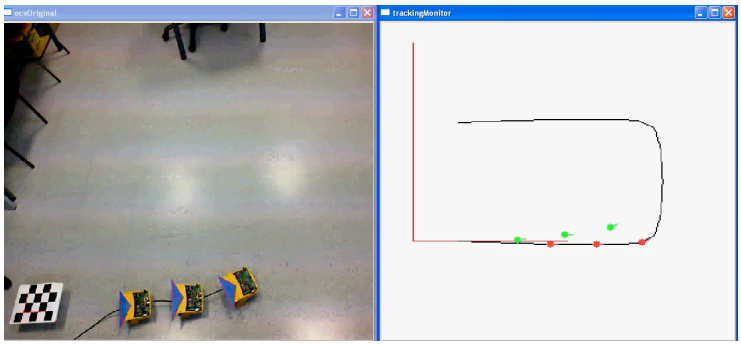



(b)

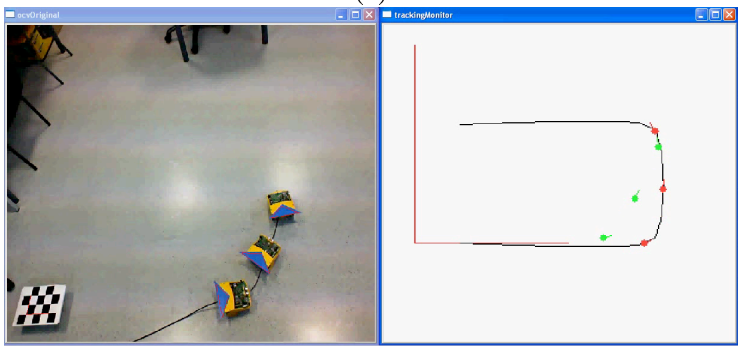



(c)

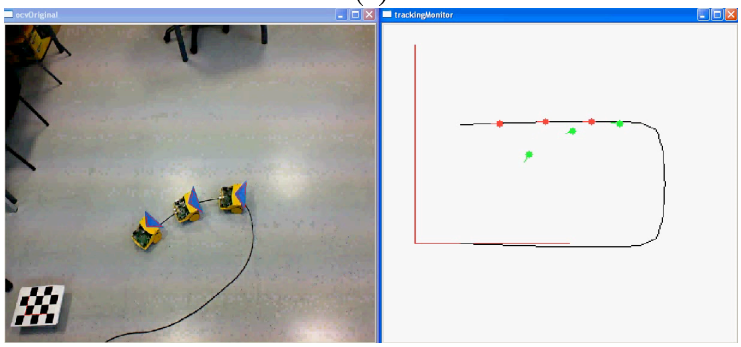
Figure 2.5: Path following experiment with linked robots. Snapshots at three time instants: (a) begining, (b) middle trajectory, (c) trajectory end.




(a)

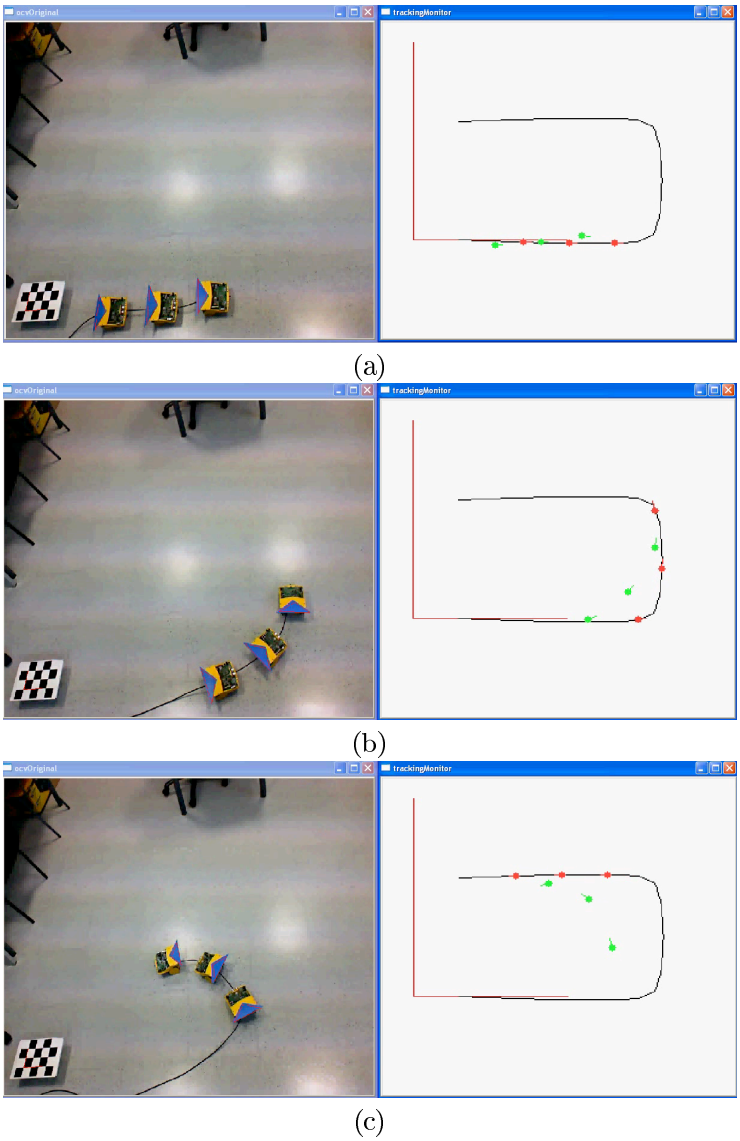



(b)




(c)

Figure 2.6: Path following experiment with linked robots, last robot dragging. Snapshots at three time instants: (a) begining, (b) middle trajectory, (c) trajectory end.




(a)

(b)

(c)

Figure 2.7: Path following experiment with linked robots, last robot switched off. Snapshots at three time instants: (a) begining, (b) middle trajectory, (c) trajectory end.

## 2.3 Hose Model

We summarize here the hose physical model which is the base for the simulation used to train and test the control system of the robot moving the hose [10, 15]. The combination of spline geometrical modeling and physical dynamical models was introduced by [33]. They allow a continuous definition of uni-dimensional objects. An inconvenient of the spline model is that, since it is exclusively based on the spline control points, it is unsuitable for representing the hose torsion. The work of [40] has improved the spline representation by combining the spline-based modeling with the Cosserat rod theory [1, 36], allowing to model the twisting of the hose. This approach, known as Geometrically Exact Dynamic Splines (GEDS), represents the control points of the splines by the three Cartesian coordinates plus a fourth coordinate representing the twisting state of the hose.

### 2.3.1 Geometry of the hose

The Cosserat rod theory [?, ?] is usually used in modeling uni-dimensional objects because it permits to model its physical behavior. In Cosserat rod theory an uni-dimensional object is described by a curve $\mathbf{r}(u)$ and a coordinate frame of director vectors $[\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3](u)$ attached to each point of the curve. The parameter $u$ goes from one end of the curve, for $u = 0$, to the other for $u = L$, being $L$ the length of the hose. The curve and the director vectors are joined into a coordinate frame $E(u) = [\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{r}](u)$. A graphic representation of the hose by the curve and the frame director vectors is shown in figure 2.8.

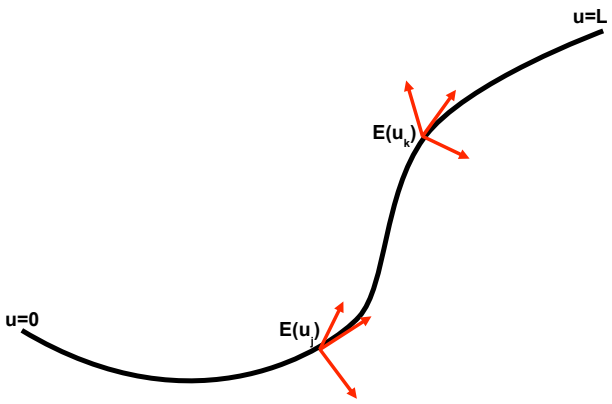



Figure 2.8: Cosserat rod model of a hose.

The GEDS describe the uni-dimensional object by a spline model taking into account the Cosserat rod approach in order to model the twisting behavior of the hose. A spline is a piecewise polynomial function. See figure 2.9 for an illustration. Splines define a curve by means of a collection of Control Points, which define a function that allows to compute the whole curve.

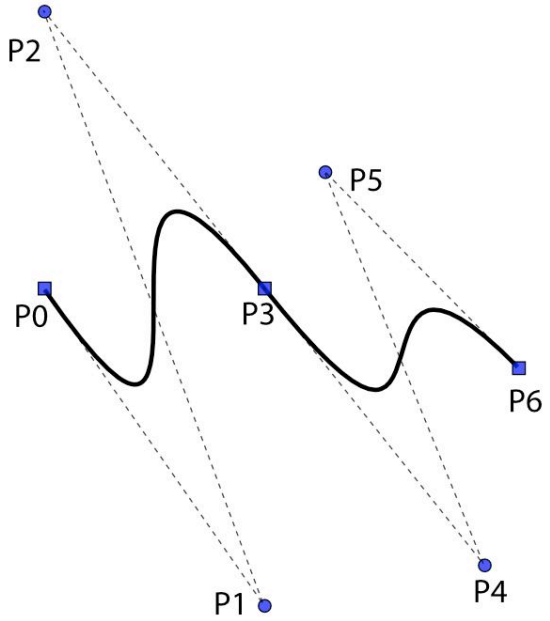



Figure 2.9: Cubic spline.

The spline expression for a curve $\mathbf{q}(u)$ is a linear combination of control points $\mathbf{p}_i$ where the linear coefficients are the polynomials $N_i(u)$ which depend on the parameter $u$ defined in $[0, 1)$. The spline equation is:

$$\mathbf{q}(u) = \sum_{i=0}^{n} N_i(u) \cdot \mathbf{p}_i, \tag{2.1}$$

where $N_i(u)$ is the polynomial associated to the control point $\mathbf{p}_i$, and $\mathbf{q}(u)$ is the point of the curve at the parameter value $u$. It is possible to travel over the curve by varying the value of parameter $u$, starting at one end for $u = 0$ and finishing at the other end for $u = 1$. We use a B-spline model for the hose, requiring a set of control points, a set of knots and a set of coefficients, one for each control point, ensuring that all curve segments are joined together satisfying certain continuity conditions [3].

Given $n+1$ control points $\{\mathbf{p}_0, \ldots, \mathbf{p}_n\}$ and a knot vector $\mathbf{U} = \{u_0, \ldots, u_m\}$,

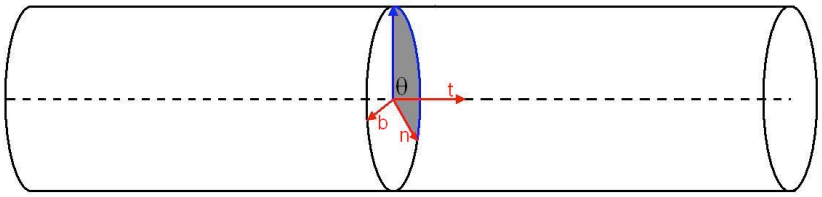



Figure 2.10: Hose section.

the B-spline curve of degree $p$ defined by these control points and knot vector **U** is:

$$\mathbf{q}(u) = \sum_{i=0}^{n} N_{i,p}(u) \cdot \mathbf{p}_i, \tag{2.2}$$

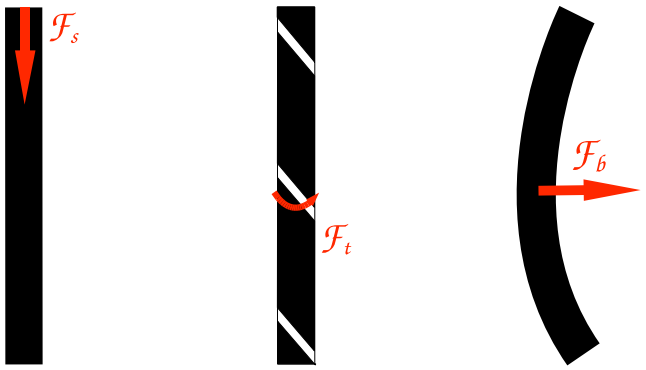
where $N_{i,p}(u)$ are B-spline basis functions of degree $p$ ($p = 3$ in this work), built using the Cox de Boor's algorithm.

$$\begin{aligned} N_{i,0}(u) &= \begin{cases} 1 & u_i \leq u < u_{i+1} \\ 0 & c.c. \end{cases}, \\ N_{i,p}(u) &= \frac{u - u_i}{u_{i+p} - u_i} \cdot N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} \cdot N_{i+1,p-1}(u). \end{aligned}$$

Because the control points of the curve will vary in time, we rewrite equation (2.2) in terms of the time parameter $t$:

$$\mathbf{q}(u,t) = \sum_{i=0}^{n} N_{i,p}(u) \cdot \mathbf{p}_i(t). \tag{2.3}$$

This extended model receives the name of *Dynamic splines,* and it is the model that we have used to model the hose. If we want to take into account the hose internal dynamics, we need also to include the hose twisting at each point given by the rotation of the transverse section around the axis normal to its center point, in order to compute the hose potential energy induced forces. In the GEDS approach, the hose model follows the Cosserat rod approach characterizing it by the curve given by the transverse section centers $\mathbf{c} = (x, y, z)$, and the orientation of each transverse section $\theta$. This description is summarized by the following notation: $\mathbf{q} = (\mathbf{c}, \theta) = (x, y, z, \theta)$. In figure 2.10, the relation between the Cosserat rod director vectors and the twisting angle $\theta$ is shown, where vector **t** represents the tangent to the curve at point **c**, and vectors **n** and **b** determine the angle $\theta$ of the transverse section at point **c**.




(a) Stretching force. (b) Twisting force. (c) Bending force.

Figure 2.11: Forces induced by the potential energy of the hose.

### 2.3.2 Hose dynamical model

From the Cosserat representation and applying the Lagrange equation (2.4) a mathematical relation between the potential energy $\mathcal{U}$, the Kinetic energy $\mathcal{T}$ and the generalized external forces $\mathbf{F}$ is obtained.

$$\frac{d}{dt}\left(\frac{\partial \mathcal{T}}{\partial \dot{\mathbf{p}}_i}\right) = \mathbf{F}_i - \frac{\partial \mathcal{U}}{\partial \mathbf{p}_i}. \tag{2.4}$$

The kinetic energy is the motion energy, while the potential energy is the energy due to the hose configuration. Let $\mathbf{F} = \{\mathbf{F}_0, \mathbf{F}_1, \ldots, \mathbf{F}_n\}$ denote the model of the external forces acting on the hose spline model control points. Each $\mathbf{F}_i$ acts on control point $\mathbf{p}_i$. It is usually assumed that mass and stress are homogeneously distributed among the $n+1$ degrees of freedom of the hose spline control model.

### 2.3.3 Potential Energy

It is necessary to determine the forces that will be generated on the hose as a consequence of its potential energy due to its physical configuration.

In figure 2.11 we can appreciate the forces and torques $\mathbf{F}_U = (\mathbf{F}_s, \mathbf{F}_t, \mathbf{F}_b)^T$ that deform the hose because of its potential energy. The stretching force, $\mathbf{F}_s$, is the force along the normal to the hose transverse section and its application results in its lengthening. The tension torque, $\mathbf{F}_t$, makes the transverse section to rotate around the center of the section. The bending torque, $\mathbf{F}_b$, modifies the

orientation of the transverse section. The forces acting on the transverse section plane are neglected, because of the Kirchhoff assumption that the transverse sections are rigid and that only the hose curvature may be distorted.

In mechanics and physics the Hooke's law provides an approximation for linear-elastic materials. This law establishes that the extension of a spring is in direct proportion to the load applied to it. Summarizing, the Hooke's law for a spring-mass system establishes:

$$F = -kx, \tag{2.5}$$

where $x$ is the displacement of the spring due to the load applied to it, $k$ is the spring constant and $F$ the restoring force experimented by the spring due to its material properties. In general the Hooke's law is applied to elastic materials because their behavior is similar to the spring as its molecules return to the initial state of stable equilibrium, quickly regaining the object its original shape after a force has been applied.

Let us denote the length of the hose as $L$, and the area of the transverse section as $A$, then the hose length extension is linearly proportional to the deformation resistance of the hose:

$$\triangle L = \frac{F}{EA}L, \tag{2.6}$$

where $E$ is the modulus of elasticity, which is the mathematical description for the hose resistance to be deformed when a force is applied to it.

Isolating the value of $F$ in equation 2.6 we have:

$$F = EA\frac{\triangle L}{L}. \tag{2.7}$$

Defining $\epsilon$ as the deformation of the hose relative to the transverse area, $\epsilon = A\frac{\triangle L}{L}$, we can rewrite 2.7 as:

$$F = E\epsilon, \tag{2.8}$$

which is a version of the Hooke's law for an elastic uni-dimensional object.

When a small deformation is considered for a relative big radius of the hose length in comparison with the radius of the transverse section, it is said that the hose is in a linear elasticity dynamic regime, and then the force equation 2.8 may be applied for each of the stretching, twisting and bending forces. The matricial version for the stretching, twisting and bending forces is:

$$F_U = H\epsilon = \begin{pmatrix} E_s & 0 & 0 \\ 0 & E_t & 0 \\ 0 & 0 & E_b \end{pmatrix} \cdot \epsilon. \tag{2.9}$$

The deformation vector, $\epsilon = (\epsilon_s, \epsilon_t, \epsilon_b)^T$, is composed of the stretching deformation $\epsilon_s$, the twisting deformation $\epsilon_t$ and the bending deformation $\epsilon_b$. The Hooke matrix, $H$, is composed of the stretching rigidity $E_s$, the twisting rigidity $E_t$ and the bending rigidity $E_b$.

Maintaining the spring-mass system analogy, the potential energy $U$ is defined as $U = \frac{1}{2}kx^2$, that in the case of the hose is defined by the following integration from $u = 0$ up to $u = L$:

$$U = \frac{1}{2}\int_0^L \epsilon^T \mathbf{F}_U du. \tag{2.10}$$

Using the definition of $\mathbf{F}_U$ from equation 2.9 in equation 2.10 we have:

$$U = \frac{1}{2}\int_0^L \epsilon^T H\epsilon du$$

Note that this model is appropriated for a hose that in rest configuration is stiffed and not twisted or bended, but for a cable as a telephone cord or a spring the rest configuration of the hose is different to zero, so $\epsilon$ should be replaced by $(\epsilon - \epsilon_0)$, being $\epsilon_0$ the rest strain.

When the objects lie in fact in the 2D space we can obviate the moments. Therefore, the potential energy $\mathcal{U}$ is defined by the following integration along the hose, from $u = 0$ up to $u = L$:

$$\mathcal{U} = \frac{1}{2}\int_0^L (\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_0)^T \mathbf{F}_{\mathcal{U}} du, \tag{2.11}$$

where $\mathbf{F}_{\mathcal{U}} = (\mathcal{F}_s, \mathcal{F}_t, \mathcal{F}_b)^T$ are, respectively, the stretching force, and the torsion and bending moments suffered by the hose due to its configuration, $\boldsymbol{\epsilon} = (\epsilon_s, \epsilon_t, \epsilon_b)^T$ is the deformation vector.

### 2.3.4 Kinetic energy

The kinetic energy $T$ is composed of the translation energy $T_t$ and the rotation energy $T_r$.

$$T = T_t + T_r. \tag{2.12}$$

The kinetic energy is given by:

$$T_t = \frac{1}{2}\mu A \int_0^L \dot{\mathbf{q}}^2 du, \tag{2.13}$$

$$T_r = \frac{1}{2}\mu \int_0^L \mathbf{\Omega}^T I \mathbf{\Omega} du, \tag{2.14}$$

where $A$ is the area of the transversal section,$\mathbf{\Omega}$ is the angular velocity, $\mu$ is the linear density and $I$ is the polar momentum of inertia.

A simplified version of the kinetic energy expression is given by defining the inertial matrix $J$, which is invariant over all the hose points because of the assumption of constant hose diameter.

$$J = \begin{pmatrix} \mu & 0 & 0 & 0 \\ 0 & \mu & 0 & 0 \\ 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & I \end{pmatrix}.$$

The kinetic energy of the hose $T$ is then defined by:

$$T = \frac{1}{2}\int_0^L \frac{d\mathbf{q}^T}{dt} J \frac{d\mathbf{q}}{dt} du. \tag{2.15}$$

### 2.3.5 Dynamic model

The kinetic energy model takes into account the translational and rotational motions of the hose, therefore, we can determine from it the acceleration of every hose point, by deriving equation 2.15. The left hand term of equation 2.4 becomes:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\mathbf{p}}_i}\right) = \frac{1}{2}\int_0^L \frac{d}{dt}\frac{\partial\left(\dot{\mathbf{q}}^T J \dot{\mathbf{q}}\right)}{\partial \dot{\mathbf{p}}_i} du. \tag{2.16}$$

Next, we consider the derivative of the potential energy relative to a generalized coordinate:

$$\frac{\partial U}{\partial \mathbf{p}_i} = \frac{1}{2}\int_0^L \frac{\partial \epsilon^T H \epsilon}{\partial \mathbf{p}_i} du. \tag{2.17}$$

The aim of the physical modeling is to determine the accelerations of the hose, in terms of its geometrical model, therefore the accelerations are obtained in the GEDS model substituting $\mathbf{q}$ in the expression of equation 2.16 by the right side of equation 2.3:

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{\mathbf{p}}_i} = \sum_{j=0}^{n} J \frac{d^2\mathbf{p}_j}{dt^2} \int_0^L (N_i(u)N_j(u))du \tag{2.18}$$

Defining:

$$\mathbf{M}_{ij} = J \int_0^L (N_i(u)N_j(u))du$$

and

$$\mathbf{A} = \left[\frac{d^2\mathbf{p}_j}{dt^2}\right],$$

The Lagrange equation (equation 2.4) becomes:

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{\mathbf{p}}_i} = \sum_{j=0}^{n} \mathbf{M}_{i,j}\mathbf{A}_j. \tag{2.19}$$

Using equations 2.19 and 2.17 the Lagrange equation is written in a matrix form:

$$\mathbf{MA} = \mathbf{F} + \mathbf{P}, \tag{2.20}$$

where $\mathbf{P} = \left[\frac{\partial U}{\partial \mathbf{p}_i}\right]$.

### 2.3.6 Hose-robots model

The whole system model, composed by the robots and the hose-like linking element, is built from the uni-dimensional element GEDS model by specifying the positions $u_r$ of the robots along the hose. A configuration $h$ of the hose-multi-robot system is defined as:

$$h = \{\mathbf{p}, \mathbf{U}, \mathbf{U_r}\}, \tag{2.21}$$

where

- $\mathbf{p}$ is the control point vector of the hose B-spline model,
- $\mathbf{U}$ is the collection of knots in the B-spline model,

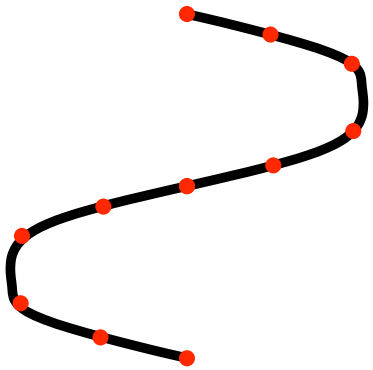


Figure 2.12: Uniform selection of the interpolating points.

- $\mathbf{U_r} \subset \mathbf{U}$ is the collection of knots that correspond to robot attachments to the hose.

The robot knot vector $\mathbf{U_r} = \{u_{r_i}\}$ contains the values of the arclength parameter $u$ where the robots are attached to the hose. The spatial position of the $i$-th robot $\mathbf{r}_i$ is given by the expression:

$$\mathbf{q}(u_{r_i}) = \sum_{i=0}^{n} N_i(u_{r_i}) \cdot \mathbf{p}_i = \mathbf{r}_i.$$

The information we have about the hose is a sequence $\tau$ of sampling points of the hose center curve, containing the Cartesian position of every point $(x, y, z)$ and its torsion angle $\theta$. Then, we construct the initial hose configuration $h_0$ by an interpolation method that generates the control points of the B-spline cubic curve that interpolates this points.More precisely, we make an uniform sampling of the hose center curve to obtain the sequence of points $\tau$ in order to get an uniform B-spline interpolant. This distribution of the sampling points optimizes the performance of the IFBA, avoiding the occurrence of spurious peaks, protuberances and loops. The sampling is done taking into account the number of knots we want to use, depending on the relation between precision and computing time that we desire.

The uniform selection of the interpolating points $\tau$ is obtained by dividing the hose length into $n$ segments, being $n + 1$ the number of control points, and choosing for each division point the hose point closest to it. Figure 2.12 shows the hose in black and the selected interpolating points in red, for a number of control points $n = 10$.

Equation 2.20 relates the acceleration at the control points with the internal energy of the hose and the external forces applied to it. Among the external forces $\mathbf{F}$ that act on the control points, we differentiate those resulting from the ones applied by the robots $\mathbf{F}_p$ from other external forces $\mathbf{F}_e$:

$$\mathbf{F} = \mathbf{F}_p + \mathbf{F}_e. \tag{2.22}$$

So, we rewrite equation 2.20 in order to determine the forces that the robot must exert on the hose attachment points as a function of the desired accelerations on the control points, the external forces on the hose and the energy configuration:

$$\mathbf{F}_p = M\mathbf{A} - \mathbf{F}_e - \mathbf{P}. \tag{2.23}$$

Since the hose dynamics are defined on the control points $\mathbf{p}_i$, a force $\mathbf{f}$ applied on a particular point of the hose is decomposed into the forces $\mathbf{f}_i$ resulting at each spline control point $\mathbf{p}_i$. The partial derivative of a point $\mathbf{q}(u)$ in the curve respect to the control point $\mathbf{p}_i$ is:

$$\frac{d\mathbf{q}(u)}{d\mathbf{p}_i} = N_i(u). \tag{2.24}$$

For a control point $\mathbf{p}_i$, the corresponding force $\mathbf{f}_i$ is computed as:

$$\mathbf{f}_i = \mathbf{f}\frac{\partial \mathbf{q}}{\partial \mathbf{p}_i} = \mathbf{f} \cdot N_i. \tag{2.25}$$

Defining the Jacobian matrix $J_{rq}$ of the robot contact points with the hose as a function of the control points, we have:

$$J_{pr} = \begin{pmatrix} \frac{\partial \mathbf{q}(u_{r_1})}{\partial \mathbf{p}_0} & \cdots & \frac{\partial \mathbf{q}(u_{r_l})}{\partial \mathbf{p}_0} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{q}(u_{r_1})}{\partial \mathbf{p}_n} & \cdots & \frac{\partial \mathbf{q}(u_{r_l})}{\partial \mathbf{p}_n} \end{pmatrix} = \begin{pmatrix} N_0(u_{r_1}) & \cdots & N_0(u_{r_l}) \\ \vdots & \ddots & \vdots \\ N_n(u_{r_1}) & \cdots & N_n(u_{r_l}) \end{pmatrix}, \tag{2.26}$$

where $u_{r_j}$ is the attachment point of the robot $\mathbf{r}_j$ to the hose.

We use the Jacobian matrix $J_{pr}$, defined in equation 2.26, to obtain the relation between the applied forces in the robot attaching points $\mathbf{F_r}$ and the resulting forces on the control points $\mathbf{F_p}$:

$$\mathbf{F_p} = J_{pr} \cdot \mathbf{F}_r. \tag{2.27}$$

### 2.3.7 The control problem

Inversion of the previous control law gives us the desired accelerations on the control points, so we can derive the desired forces $\mathbf{F}_p$ that must be resulting on the control points from the robot actions. Therefore we have to determine the forces $\mathbf{F}_r$ that the robots must apply on their contact points with the hose, by inversion of equation 2.27. But in general, matrix $J_{pr}$ is not invertible.

The control problem consists in:

1. Determining the desired positions of the hose control points, and
2. Deriving the necessary forces exerted by the robots in order to ensure that the hose control points reach the desired positions.

Determining the desired positions depends on the accuracy of the hose analytical model as well as on the accurate definition of the task required from the system, provided in the form of a sequence of hose configurations, ideally given by the spline parameters. This is often not possible, because determining the sequence of hose configurations may be equivalent to sove a complex planning problem.

The derivation of the necessary robot actions (motions, forces) is hampered by the uncertainty and noise inherent to robotic systems. Identification of the current system state (the hose configuration, the position of the robots) has an inherent degree of uncertainty, due to sensor noise, processing delays and signal processing issues. Therefore, the decided action may be irrelevant or inadequate to the current state. Moreover, the result of the robot actions may be different from the command definition due to the environment or to inner conditions of the robot.

All these problems point to the need of autonomous adaptive control learning systems applied to the solution of the control problem.

# Chapter 3

# Hose transportation as a cooperative Multi-Agent Systems

In this chapter we show the effect of the linking element, the hose, in an indirect way. We propose the framework of Cooperative Multi-Agent Systems (MAS) to build a distributed control system for the robots which works fine for a set of disconnected robots. The experiments performed on the simulated system show that when there is a hose lingking the robots, the control system is heavily affected and the distributed control is no longer effective.

The chapter is organized as follows: Section 3.1 gives an introduction to the chapter. Section 3.2, gives a formal problem statement, some basic definitions, a dynamic model for the system and the definition of two performance measures, so that the individual and overall system performance can be measured and compared. Section 3.3 details how MAS consensus-based methodologies are applied to build the control of this system. Section 3.4 shows some experiemental results from the simulation of system settings that serve to make our point. Finally, our conclusions are given in Section 3.5.

## 3.1 Introduction

There are several reviews giving different categorizations of multi-robot systems [7, 4, 11, 8] focusing in different aspects. The kind of systems we are dealing wiht in dissertation are Linked Multi-Component Robots (L-MCRS) : autonomous robot units linked through a non-rigid linking element. The linking element can be elastic, flexible or inert. It introduces a highly non-linear component in the system dynamics, because forces affecting the robots can be dumped, amplified or distorted through it. L-MCRS is a new paradigm in robotics, with a very clear real life metaphor: the transportation of hoses, wires, cables and other near unidimensional elements. Contrary to uncoupled multirobot systems for which there are huge amounts of literature, including a methodology to develop cooperative distributed control [35] systems, there is scarce relevant previous literature on L-MCRS control [15, 10, 25, 29]. The transportation, deployment and manipulation of a long[1] almost uni-dimensional object is a nice example of a task where a MAS may easily improve over a single robotic agent. It needs the cooperative works of a team of robots. In some wildly unstructured environments like shipyards or large civil engineering constructions a typical required task is the transportation of fluid materials through cables, pipes or hoses. The manipulation of these objects is a paradigmatic example of a L-MCRS, where the carrier robot team will have to adapt to changes in the dynamic environment, avoiding mobile obstacles and adapting its shape to the changing path until it reaches its destination.

The assertion that the existence of a non-rigid linking element defines a new kind of systems is very strong, needing some justification in the form of instances of problems or tasks. If we deal with the control of the entire system we would like to show that the existence of the non-rigid linking element can be highly disruptive of any control architecture designed to control the system without taking the linking element into account. We need to define a very precise task, because the control architecture can be dependent on the intended task. The line of reasoning of this chapter is the following:

1. We define the task as the path that must followed by the robots of the team. Such path independent of the number of robots and the existence of the linking element

[1] The adjective "Long" used here is relative to the size of the individual robots. The object's length must be some order of magnitude greater than the robot's size.

2. We define a control architecture suitable to deal with a team of independent robots.

3. We define a simplified system model in order to allow for the observation of the most elementary effects.

4. We simulate the system performing the task under the defined control architecture.

5. We observe deviations from the reference behavior obtained when the linking element is present.

We resort to system simulation because it is easiest to assess the effect of changes in the system, and the efflects can be attributed to the new elements. Besides, simulation allows reproducibility of the results. Real life experiments, such as the ones reported in Chapter 2 to provide proof-of-concept realizations, are affect by noise in the sensing device, in the environment, the communication channel, the specifics of the system embodiment (the hose characteristics, the robots' specifications, the actual robot internal state), the initial configuratoins, and a long list of conditions that can affect the outcome of each experiment, being irrelevant to the main hypothesis under test: the effect of the linking element.

### 3.1.1 The task

The general structure and composition of this hose transportation robotic MAS would be that of a group of robots attached to the hose at fixed or varying points. The robots would search for space positions in order to force the hose to adopt a certain shape that adapts to the environment, while trying to lead the head of the hose to a goal destination where the corresponding fluid will be used for some operation. In figure 3.1 we give a rough illustration of this problem. The pipe at the lower left corner represents the fluid source, the hose is represented by the black thick line, and the small robots attached to it try to move it so that its head (attached to a robot) reaches the goal represented by the circle in the top right corner. The other objects in the scene represent changing environment conditions that may force changes in the hose spatial disposition. This general form can take multiple implementations depending on several elements of its design:

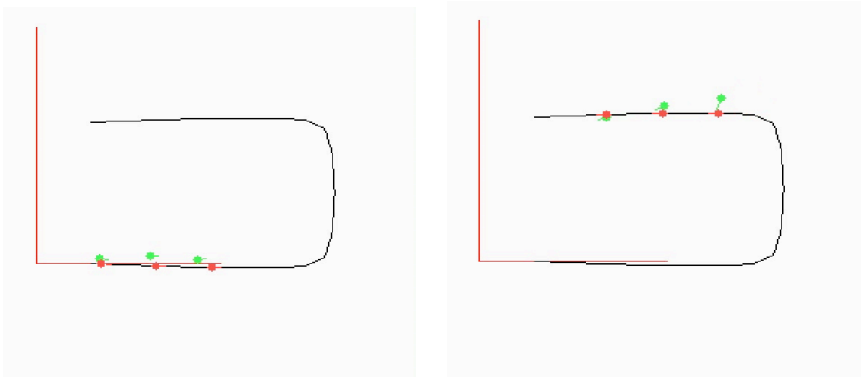


Figure 3.1: Hose transportation and deployment by three robots between starting and goal positions along a path. The red dots represent the reference robot positions on the path, the green dots represent the actual robot position. The hose beween robots is not depicted.

- Robot-hose attachment: Robots could be fixed to a point of the hose, they can move along it, or they can pull it through special gripping mechanisms.
- There can be a centralised control which determines the positions of each of the robots or, in a true MAS approach, it can be decentralised, taking each robot its own control decision.
- Robots can be homogeneous or heterogeneous, having different configurations and tasks (e.g., "pulling" robots, which tow the hose, and "cornering" robots, which take fixed positions and give shape to the hose).
- Perception can be global, with some agent acquiring a global view of the system, or local, with every robot acquiring information of its close surrounding, which can be shared with the remaining robots or not.
- The task to be performed is that of transporting the hose while following a predefined path.

### 3.1.2 Control architecture

Controlling individual wheeled mobile robots to follow a predefined path has been approached in several ways: smoothed bang-bang controllers [23], PID and adaptive controllers [19], fuzzy controllers [34, 24], tracking-error model-based predictive controllers [22], or through dynamic feedback linearization [31].

Some authors have work on the path following problem for independent and disconected multi-robot systems keeping robot formations along the path [37] using a cooperating Multi-Agent Systems (MAS) approach. MAS have been proposed in several application domains as a way to fulfill more efficiently a task by cooperation between several autonomous agents [12]. This paradigm has a very direct application in robotics, as the physical limitations of the real-life robots and the environments they are supposed to work into impose severe restrictions to their capability to fulfill some tasks, up to the extent that there are complex tasks that can not be accomplished by a single robot and must be performed necessarily by a multi-robot system [7].

We introduce a distributed control schema for the case of carrying the hose along a pre-specified path. In this chapter we follow the consensus-based approach presented in [35] as the cooperative control paradigm used to derive a cooperative control scheme for the team of robots carrying the hose. Cooperative control requires data exchange betusing some specific simulation results, that the cooperative control approach of [35] does not cope with the non-linear physical interactions (couplings) between the robot units induced by a simple elastic linking element.

### 3.1.3 Simplified model

In Chapter 2 we have introduced the hose transportation problem, giving a detailed analytical dynamical model which allows the accurate simulation of the system under various conditions. In this chapter we will use a simplification of the GEDS model, which is suitable for the experiments carried out. We are asumming the simplest model: a compressible spring, one that produces no effect when compressed and exerts some force on the pulling robots when extended above a critical length. The hose-like linking element will not interfere with the robots when its length is below this critical length.

## 3.2 Formal problem statement

We formalize the task of transporting a hose-like physical element by a group of mobile robots. Each robot is attached to the hose, so that the hose segment between each pair of robots is $L$ meters long. If the hose is compressed or folded there will be no dynamical effect on the robots, that is, they will not be pushed or pulled by the hose to recover its linear shape. The hose is assumed

to be elastic, that is, a hose segment can be stretched beyond its nominal size of $L$ meters, however there is a spring-like force acting on the robots trying to receover this nominal length. We do not consider that the hose can be broken if some plasticity limit is reached. We do not consider the situation when a robot is blocked in its way by the hose itself.

### 3.2.1 Definitions and restrictions

The virtual path to be followed by the robots is defined as a function of the distance traveled along the path $s$:

$$H(s) = (h^x(s), h^y(s))$$

The following path tracking control framework is assumed: each physical robot follows the reference position of a virtual robot that walks over the virtual the desired path. The physical robot controller tries to minimize the error between its position and that of the virtual robot. We know the $i$-th physical robot's position in bi-dimensional space, denoted as $\mathbf{P}_i \equiv [P_i^x \; P_i^y]$, and the position of its corresponding virtual robot along the desired trajectory $s_i$. The $i$-th virtual robot's position in bi-dimensional space is always given as a position in the virtual path $H(s_i)$.

We define a function

$$\varphi_H(s, L) = \min_{s'>s} \left\{ \left\| \vec{H}(s) - \vec{H}(s') \right\| = L \right\},$$

providing the closest ahead position along the path $s'$ which is $L$ meters distant from $H(s)$. This function gives the position of the next robot ahead in the hose configuration to a given one. We will need to know the desired positions of the robots along the hose so that the rear robot is in a given position, in order to define the desired system configuration both for the control system and for the performance metrics. The following recursive function provides this information for each robot unit along the hose:

$$\psi_H(s, L, i) = \begin{cases} s & i = 0, \\ \psi_H(\varphi_H(s, L), L, i-1) & else. \end{cases} \tag{3.1}$$

The hose will be modelled by a series of line segments connecting the robots. If these hose segments have to be of constant length $L$, then we are imposing

the following restriction to the system: robots must be at constant distance $L$ in the plane. Not all paths can be traveled by pairs of vehicles keeping a fixed distance between their positions in the plane. As a working hypothesis, we assume that there is a solution for this problem for $H(s)$ if $\varphi_H(s, L)$ is a continuous monotonically increasing function defined for every value of $s$. This condition means that the path does not have foldings of radius lower than $L$. Nevertheless, we allow hose segments to change their length along the simulation introducing a corresponing error measure. This error measure may a non-zero minimum value due to the path's topological properties.

### 3.2.2 Simplified Dynamic model

The GEDS model described in Chapter 2, which has been used also in [10], includes three different kind of forces induced by the potential energy of a hose:

- stretching/compression forces resulting from the pulling/pushing along the hose main axis that produce some increase or decrease of the hose length,
- twisting forces as a result of rotations of the hose around the hose main axis, and
- bending forces, appearing as the hose opposses rotation of a section around any axis orthogonal to the main hose axis.

We have simplified the model retaining only the stretching forces arising when the hose length is increased due to the robot separation, assuming that other hose-related forces can be neglected. The traction force $\mathbf{T}_i$ between $i$-th and $(i+1)$-th robots are modeled as a clamped spring neglecting compression forces and taking into account only elastic forces. Therefore, no force is applied if the euclidean distance between two robots is less than $L$:

$$\begin{aligned} T_i^x &= K^s * \max(0, \|\mathbf{P}_i - \mathbf{P}_{i+1}\| - L) * \cos(\beta_i), \\ T_i^y &= K^s * \max(0, \|\mathbf{P}_i - \mathbf{P}_{i+1}\| - L) * \sin(\beta_i), \end{aligned} \tag{3.2}$$

where $K^s$ is the spring constant and $\beta_i$ is the angle formed between the line connecting the $i$-th and $(i+1)$-th robots and the $x$ axis.

The following kinetic equations have been used:

$$\begin{aligned} \dot{\mathbf{V}}_i &= \frac{\mathbf{F}_i}{m}, \\ \dot{\mathbf{P}}_i &= \mathbf{V}_i, \end{aligned}$$

where $m$ is the mass of the robots, $\mathbf{V}_i \equiv \left(V_i^x, V_j^y\right)$ represents the $i$-th vehicle's velocity vector and $\mathbf{F}_i \equiv (F_i^x, F_i^y)$ represents the force vector applied by the $i$-th vehicle. Including the spring model describing the stretching forces (eq. 3.2) into our dynamic model, we obtain:

$$\dot{\mathbf{V}}_i = \frac{\mathbf{F}_i - \mathbf{T}_{i-1} + \mathbf{T}_i}{m}. \tag{3.3}$$

### 3.2.3 System performance measures

We need to define system performance measures so that we can provide quantitative comparisons of the system behavior under different conditions, i.e. with and without the hose dynamics. Two functions have been used to measure individual error:

- Mean error in the square euclidean distance between robots $e_i^{dis}$, that is, how much their relative distances differ from the stated nominal distance $L$:
$$e_i^{dis} = \frac{1}{t}\int_0^t \left(\|\mathbf{P}_i - \mathbf{P}_{i+1}\| - L\right)^2.$$

- Mean error euclidean distance between robots and their desired position $e_i^{pos}$:
$$e_i^{pos} = \frac{1}{t}\int_0^t \left(\|\mathbf{P}_i - H(s_i)\|\right)^2.$$

Using these two functions, the global system error has been measured as the sum of the mean square deviations:

$$e^{dis} = \sum_{i=0}^{n-2} e_i^{dis},$$

and

$$e^{pos} = \sum_{i=0}^{n-1} e_i^{ref}.$$

The $e^{dis}$ gives a measure of the streess that the hose is suffering because of the deviations of the robots from their reference positions. Only the extension beyond the nominal length has some dynamic effect in the simplified model, however, compression is also an issue, and a performance measure for the control strategies. The $e^{pos}$ gives a measure of the control error which is independent of the existenco of a hose, it can be applied to linked and non-linked systems.

## 3.3 Cooperative control

In this section we formulate the decentralized distributed control system that will compute the decision of the control commands on each individual robot. This control strategy is optimal for systems of non-linked robots. In[35] two basic consensus-based methodologies are described to approach distributed multi-vehicle cooperation problems: with and without an optimization objective. In the first case, an objective is desired to be optimally achieved while in the second, only cooperation among the individuals is desired. We follow the first approach, applying it to the virtual robots, which are controlled in a cooperative way. The essence of the methodology can be summarized in four steps [35]:

- (a) definition of the cooperation in terms of an objective function and a constraint function,
- (b) definition of the coordination variables and coordination functions[42],
- (c) design of a centralized cooperation scheme and
- (d) specification of a consensus-based distributed cooperation scheme.

The are two ways to represent the system state [35]:

1. Group-level reference state: The individual decisions are derived from the values of a set of global state variables
2. Individual vehicle states: Each individual acts according to its own state and its knowledge of the the states of its neighbors.

In this chapter, we follow a group-level reference. We use as the reference for the entire system the desired position of the last robot, denoted $\xi$. It is the base-position of the robot team formation. Using the previously defined function of

equation (3.1), the desired virtual position along the hose for each robot in the formation is expressed as $\psi_H(\xi, L, i)$, where $i$ is the index of a robot in the team formation (counting from zero).

### 3.3.1 Cooperation Constraint and Objectives

First of all, the cooperation constraint must be identified. This is a function that provides the formal description of the conditions under which cooperation is considered successful. This function is defined to be zero when ideal cooperation is achieved. Sometimes a tolerance value $\epsilon$ is defined, so when the cooperation constraint function reaches a value under $\epsilon$ the system has achieved a level of $\epsilon$-cooperation.

From the problem definition of the hose transportation problem, a fixed $L$ distance is to be kept between the positions of every pair of consecutive robots. This is the major constraint relating the behavior of the robots, so this has been chosen as the cooperation constraint objective function. It is formalized as:

$$J_{con} = \sum_{i=0}^{n-2} | \, \|H(s_i) - H(s_{i+1})\| - L|.$$

Notice that the constraint is not defined on the positions along the path or on the physical robots actual positions, but on the virtual spatial positions that the robots must follow.

The cooperation objective function captures all the auxiliary objectives as a positive definite function. We are intested in preserving the robot formation carrying the hose, therefore it is defined as follows:

$$J_{obj} = \int_t^{\infty} (s_i(\tau) - \psi_H(\xi(\tau), L, i))^2 d\tau.$$

The effect of minimizing this function is that each virtual robot's controller tries to minimize the distance for its own reference from "where it is" ($s_i$) to "where it should be" ($\psi_H(\xi, L, i)$) according to the team formation base position $\xi$.

### 3.3.2 Coordination Variables and Functions

The *coordination variable* provides the minimum amount of information required to effectively define the cooperation among the team of robots. Cooperation implies the computation of the decisions by each of the individual members of the system in order to reach the goal. In the modeled hose transportation

system, all control actions for all the virtual robot can be derived from the base-position of the formation $\xi$, therefore $\xi$ is the coordination variable. The following dynamics for $\xi$ are assumed:

$$\dot{\xi} = v,$$

where $v$ represents the desired velocity of the system transporting the hose along the path. In other words, the motion of the reference position corresponds to the motion of the entire system. Moreover, the knowledge of the coordination variable value determines the knowledge of the status of the entire system.

Because the knowledge of the coordination variable allows to compute everything, we can express the cooperation constraint as a function of the cooperation variable. Given that, by definition, $\|H\left(\psi_H(\xi, L, i) - H\left(\psi_H(\xi, L, {}_{i+1})\right)\right\| = L$, a new cooperation constraint can be defined as a function of the coordination variable:

$$\begin{aligned} J_{con} &= \sum_{i=0}^{n-2} \|H(s_i) - H(\psi_H(\xi, L, i))\| \\ &+ \|H(s_{i+1}) - H(\psi_H(\xi, L, {}_{i+1}))\|. \end{aligned}$$

If we want to derive local control policies for each robot, we need to decompose the global cooperation objective function into a combination of the local constraint functions that can be computed independently by each agent on the basis of its knowledge of the coordination variable. The cooperation objective is, therefore, expressed as a convex function of individual local objective functions:

$$J_{obj} = \sum_{i=0}^{n-1} J_{cf,i},$$

where

$$J_{cf,i} = \int_t^{\infty} \left(s_i\left(\tau\right) - \psi_H(\xi\left(\tau\right), L, i)\right)^2 d\tau. \tag{3.4}$$

Therefore, the local control decisions will be computed in order to optimize (minimize) the corresponding local objective function, according to the local knowledge of the coordination variable value.

### 3.3.3 Centralized Cooperation Control Scheme

A control scheme is derived as the minimization of the cooperation objective function subject to the cooperation constraint. The assumption of a central agent which possesses complete and uncertain knowledge about the system's state allows to specify the minimization problem. The central agent knows the exact actual position of the robots, their desired positions as a function of the coordination variable and is, thus, able to compute the corresponding error functions. The optimization problem is stated as the search for the optimal position of the rear robot minimizing the joint position error of the robots, subject to the constraint of constant distance between robots. Formally:

$$\xi = \arg\min \left\{ \lim_{t\to\infty} \sum_{i=0}^{n-1} J_{cf,i}(\xi, x_i) \right\},$$

subject to

$$\lim_{t\to\infty} J_{con}\left(\xi, X\right).$$

Because the central agent has complete uncertain knowledge of the system's state, and the global cooperation objective function is linear function of the local cooperation objective functions it is possible to decouple the minimization cooperation objective function into the minimization of its constituent local cooperation objective functions. Defining the position error of the $i$-th virtual robot along the path as a function of the coordination variable as follows:

$$e_i^{ref} = \psi_H\left(\xi, L, i\right) - s_i,$$

a Proportional Integral controller can be defined to solve this optimization problem as

$$u_i = v + K_p e_i^{ref} + K_i \int_0^t e_i^{ref} dt,$$

where $u_i$ is the velocity applied to the $i$-th virtual position reference and $K_p$ and $K_i$ are the proportional and integral constants used for the controller to minimize the local cooperation objective function given in equation (3.4).

### 3.3.4 Coordination variable estimation through Consensus

If we consider that the robots are autonomous agents, performing control decisions on the basis of local available knowledge of the system's state, then we need to provide some mechanism for asynchronous determination of the value of the global state variables, namely the coordination variable, whose knowledge allows the computation of the control decisions. The distributed asynchronous algorithm to obtain estimations of the non-local variables is known as a *Consensus* algorithm because it uses all available information from the other robots to build a consensual estimation.

We denote $\xi_i$ as the local estimation of $\xi$ at the $i$-th robot unit. The Consensus Algorithm is expressed as:

$$\dot{\xi} = -\sum_{i=0}^{n-1} a_{ij}(t) * (\xi_i - \xi_j) + v, \tag{3.5}$$

where $a_{ij}(t)$ represent the weight applied by the $i$-th Consensus Algorithm to the $j$-th local estimation communicated at instant $t$. These consensus coefficients can be set in many ways representing weightings on the knowledge of local estimations. They model the confidence that we have in one source of information and thus embody a model of the system communication reliability. The lack of *a priori* knowledge is represented as equal weights to all information sources, computing in fact the average of the local estimations.

The local control law implemented in each vehicle to achieve a cooperative solution to the distributed control problem is a Proportional-Integral controller whose terms are estimations of the error based on the estimation of the coordination variable value.

$$\hat{e}_i^{ref} = \psi_H(\xi_i, L, i) - s_i,$$

Therefore, the local control Cooperation Algorithm is given by :

$$u_i = v + K_p \hat{e}_i^{ref} + K_i \int_0^t \hat{e}_i^{ref} dt. \tag{3.6}$$

The Consensus Algorithm depends on the quality of the communication between the robots. There are two sources of trouble, the noise in the communication channel and the delays in communication introduced by the communication protocols and the noise. To assess the Consensus Algorithm robustness, the communications between five robots moving along a line were simulated

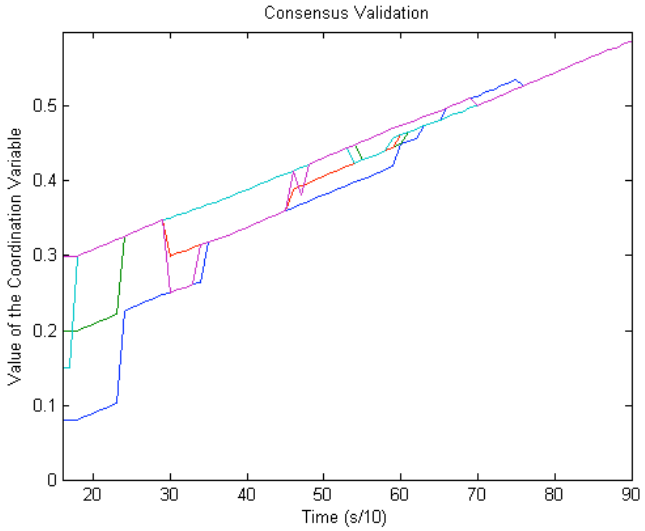



Figure 3.2: Convergence of the different local estimations of the Coordination Variable to a common estimation in the event of 95% error in communication between robots.

with a random error rate of 95% and different values for all the instances of the coordination variable, this is, only 5% percent of the messages sent were correctly received. Even under this hard error rate, all the local estimations of the coordination variable converged rather fast as shown in fig.3.2. In this figure, each line corresponds to a local estimation of the coordination variable $\xi$. The lack of initial knowledge makes initial estimations widely different, however as times evolves the distributed consensus approaches a unified estimation. Fig.3.3 shows the succesfully sent message count over time. It can be appreciated that in many time instants the number of successful communications is zero, however the system is robust enough to achieve convergence.

### 3.3.5 Distributed Control Scheme

The local control of one of the robots following this distributed asynchronous control scheme is shown in figure 3.4. The $i$-th Consensus Module receives all the local estimations of $\xi$ available to the $i$-th vehicle and updates its local estimation according to the Consensus Algorithm (equation 3.5). This updated estimation $\xi_i$ is fed back to the Communication Network, so the remaining robot units can update their own estimations, and fed it as the input to the

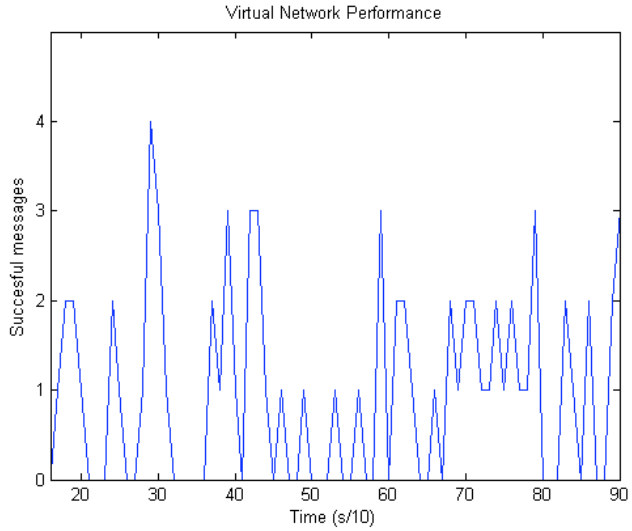



Figure 3.3: Successful messages transmitted between robots at each time instant in the event of 95% error in communications.

$i$-th Cooperation Module, which runs a PI controller, previously presented in equation 3.6). This controller's output is the velocity over $s$ which is the input to the $i$-th Path Tracking Module, which updates the $i$-th virtual robot's position along the path according to that velocity, and feeds back this new position to the $i$-th Cooperation Module. In the last step, the $i$-th Reference Tracker calculates the error in $x$ and $y$ axis and runs a PI controller which tries to minimize the error. The force output of the controller is used to calculate the next position for the $i$-th physical robot using the dynamic model described previously.

## 3.4 Simulation Experiments

The following simulation experiments are designed to demonstrate the dynamical effect introduced by the physical link modeled by a nominal clamped spring force ($K^s = 40\,Nm$). The absence of the hose is modeled by removing the spring ($K^s = 0\,Nm$). The traction of each simulated physical robot is bounded by a maximum force output $F_i^{max}$ and its output force vector was then clamped so that the individual performance could be individually affected. Formally:

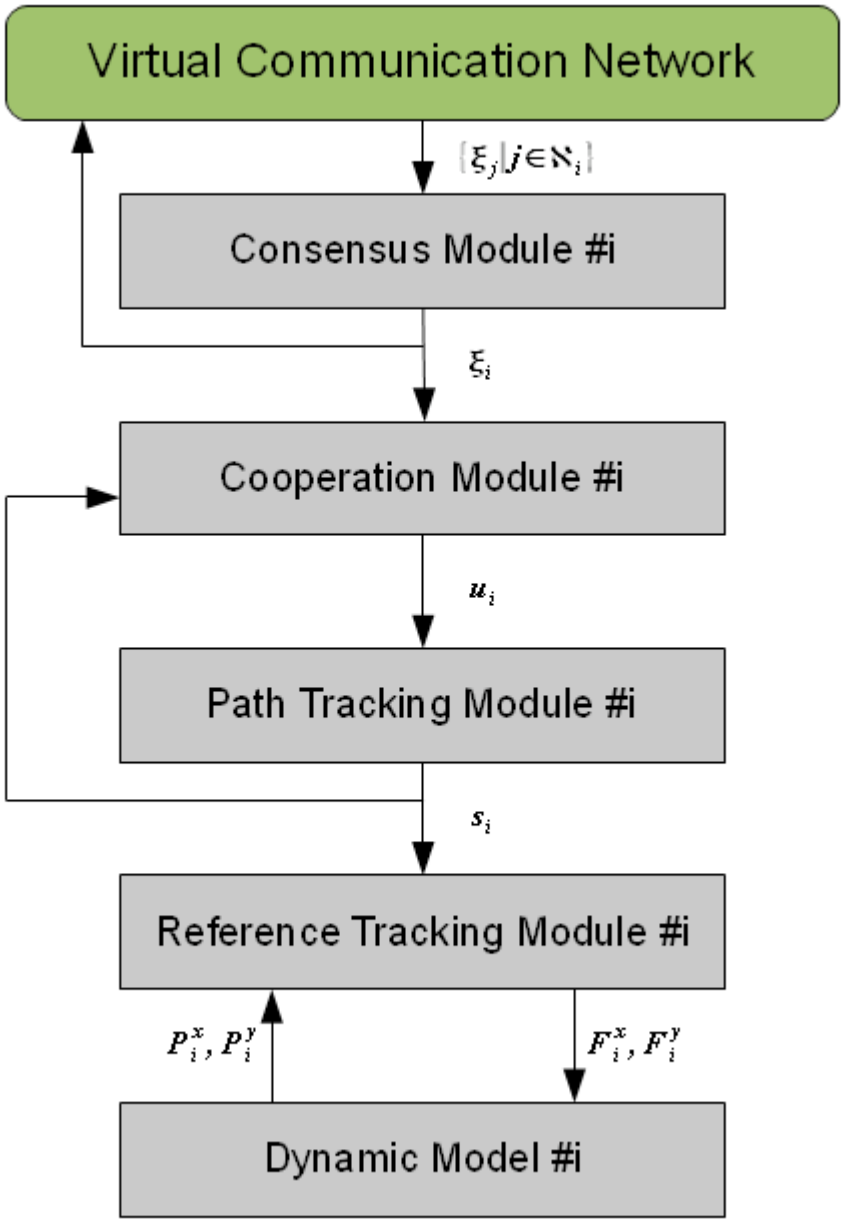



Figure 3.4: Local control in a distributed asynchoronous control scheme.

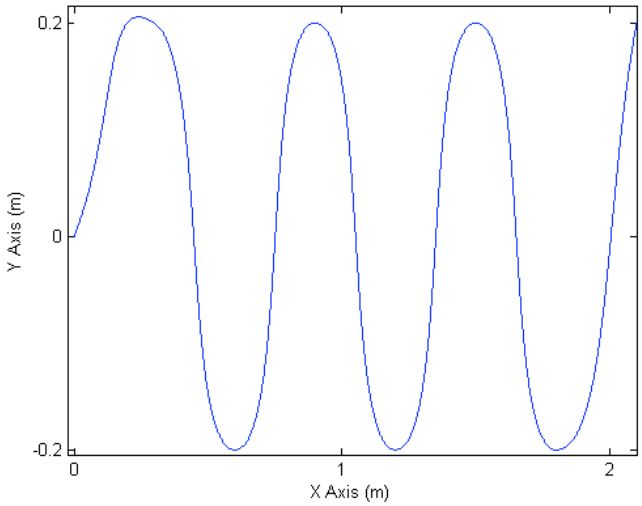



Figure 3.5: Path followed by robot units in experiments A and B

$$\mathbf{F}_i^{'} = \frac{\mathbf{F}_i}{\max\{1, \|\mathbf{F}_i\| / F_i^{max}\}},$$

where $\mathbf{F}_i$ is derived from equation (3.3) in order to minimize the distance to the reference virtual position.

Two experiments were conducted simulating 5 robots traveling along the path represented in figure 3.5 keeping a fixed separation of $L = 0.2m$ between every consecutive robot pairs. In both experiments, the system was simulated including the hose dynamic model ($K^s = 40Nm$) and without it ($K^s = 0Nm$), so the performance impact due to the physical link could be observed and quantified.

### 3.4.1 Experiment A

The last robot was set to be the weakest: $F_0^{max} = 2.5N$ and $F_1^{max} = F_2^{max} = F_3^{max} = F_4^{max} = 5N$. Results are shown in table **??** and figures 3.6 to 3.9. Table **??** rows contain the performance measures for the individual robots. Las rows correspond to the global performance. The existence of the hose is indicated by the value of $K^s$, when it is zero the hose is absent. Table **??** shows that the poor individual performance of the last robot ($i = 0$) doesn't affect the

individual performance of the remaining robot units when there is no hose in the system, therefore no elastic forces are introduced in the sytem ($K^s = 0$). In this case, the performance measures $e_i^{dis}$ and $e_i^{pos}$ remains constant $e_i^{dis} = 0.000$ and $e_i^{pos} \simeq 0.002$ for all remaining robots $i = 1, ..., 4$.

When the hose-related elastic forces are introduced in the system ($K^s = 40$), the rear robot is pulled by the other robots by the effect of the hose elasticity, then $e_0^{dis}$ drops to zero as could be expected because the model avoids separations between robots bigger than $L$. However, the last robot moves slower than the rest and the elastic forces trying to maintain the distance between robots below $L$, force the other robot units to go slower, making them unable to follow their own virtual robot references as accurately as in the uncoupled case. The global system error $e^{pos}$ grows from 0.0111 to 0.0216, which implies a 95% error growth due to the hose dynamics. Therefore, the hose has a definite impact on the systems's behavior.

For a qualitative assessment of the experiment results we refer to figures 3.6 to 3.9. Figures 3.6 and 3.8 represent the time evolution of the Reference-Position error ($|P_i - P_{i+1}|$) for $K^s = 0$ and $K^s = 40$ respectively. The traction effect due to the inclusion of the hose can be clearly appreciated if both figures are compared. Figures 3.6 shows that the error of the last robot unit does not influence the behavior of the remaining robot units, figure 3.8 shows how the error between the virtual robot references and the physical robots is propagated across the system. The rear robot drags the other robots making their lags relative to the virtual reference increase following the same pattern as the rear robot. Notice also that this error propagation is damped through the system, the front robot error is much less than the nearest robot to the rear. The Reference-Position distance oscillates as the virtual robots go faster than the physical robots at the curves to compensate the change in the growth of the euclidean distance between robots, and oscillations get closer to zero as time goes on, due to the Integrative component of the Proportional-Integrative controller used. Observing figure 3.7 the distance between robots is around nominal for the healthy robots, increasing for the unhealthy robot, when there is no hose in the system. Introducing the hose, in figure 3.9 we observe that the distance between robots reaches its nominal value for each pair, though at the beginning the lead robot is strongly dragged by the rear robots.

Table 3.1: Performances obtained in Experiment A: last robot reduced force

| | Individual performance | | | | | |
|---|---|---|---|---|---|---|
| $K^s$ | $i$ | 0 | 1 | 2 | 3 | 4 |
| 0 | $e_i^{dis}(*10^3)$ | 10.2 | 0.0 | 0.0 | 0.0 | |
| | $e_i^{pos}(*10^3)$ | 10.3 | 0.2 | 0.2 | 0.1 | 0.2 |
| 40 | $e_i^{dis}(*10^3)$ | 0.0 | 0.0 | 0.0 | 0.0 | |
| | $e_i^{pos}(*10^3)$ | 10.0 | 6.9 | 2.5 | 1.5 | 0.7 |

| $K^s$ | System performance | | | |
|---|---|---|---|---|
| 0 | $e^{dis}$ | 0.0104 | $e^{pos}$ | 0.0111 |
| 40 | $e^{dis}$ | 0.0000 | $e^{pos}$ | 0.0216 |

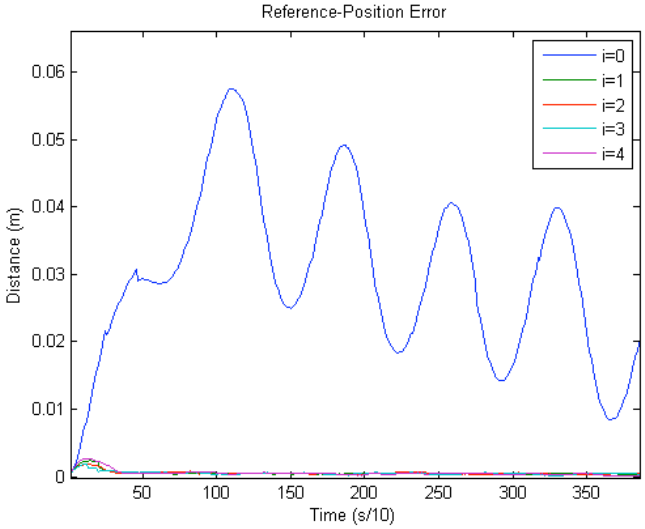



Figure 3.6: Reference-Position error with no physical link.

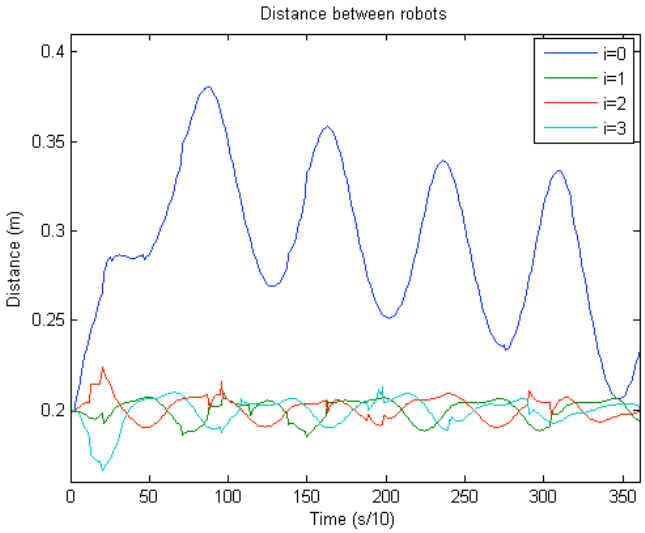



Figure 3.7: Distance between robots with no physical link.

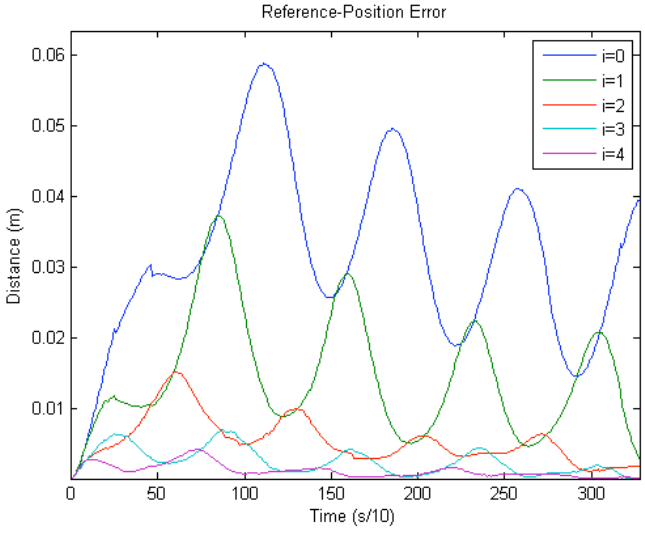



Figure 3.8: Reference-Position error with a physical link ($K = 40\ N * m$).

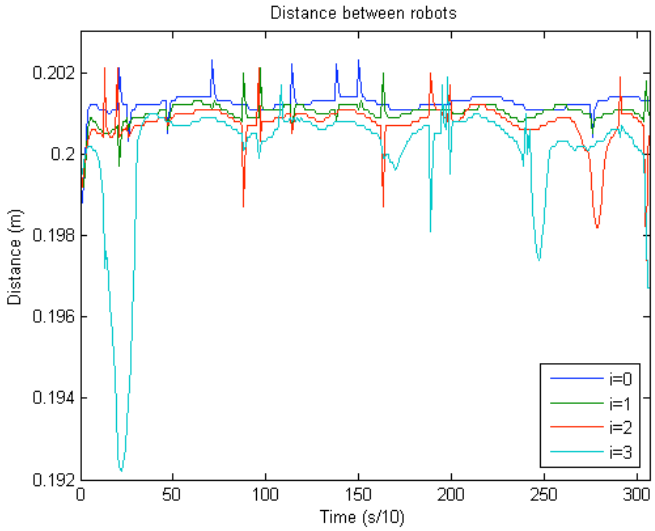



Figure 3.9: Distance between robots with a physical link ($K = 40\ N * m$).

### 3.4.2 Experiment B

The $3^{rd}$ robot was set to be the weakest: $F_2^{max} = 2.5N$ and $F_0^{max} = F_1^{max} = F_3^{max} = F_4^{max} = 5N$. Results are presented in table **??** and figures 3.10 to 3.13. We can see in Table **??** that, because the $3^{\text{rd}}$ $(i = 2)$ robot is expected to be the slowest, the distance errors $e_1^{dis}$ and $e_2^{dis}$ show a greater error than the rest. Fig. 3.11 gives further insight: the $3^{\text{rd}}$ can't go fast enough to follow his virtual robot reference so the distance from the robot ahead $(e_2^{dis})$ grows and, for that same reason, the distance from the $2^{\text{nd}}$ $(e_1^{dis})$ gets smaller. This same error $e_1^{dis}$ suggests that collisions between $2^{nd}$ and $3^{rd}$ robot could occur in real world when it gets close to zero.

The dynamic model inclusion spreads the error among neighbors, and makes $e_0^{dis}$, $e_1^{dis}$, $e_3^{dis}$ and $e_4^{dis}$ grow from an average 0.0002 error to 0.0010, 0.0011, 0.0054 and 0.0018 respectively, which means an average growth of nearly $10^3\%$ on the individual distance error. System performance error $e^{pos}$ grows from 0.0090 to 0.0177, which represents a 96.7% growth, similar to the growth observed in Experiment A. Confirming the statistics presented on Table **??**, Fig. 3.12 shows once again the Reference-Position spreading among neighbors.

Table 3.2: Results of the Experiment B

| | Individual error | | | | | |
|---|---|---|---|---|---|---|
| $K^s$ | $i$ | 0 | 1 | 2 | 3 | 4 |
| 0 | $e_i^{dis}(*10^3)$ | 0.0 | 17.4 | 9.5 | 0.0 | |
| | $e_i^{pos}(*10^3)$ | 0.2 | 0.2 | 8.0 | 0.2 | 0.3 |
| 40 | $e_i^{dis}(*10^3)$ | 0.0 | 16.1 | 0.0 | 0.0 | |
| | $e_i^{pos}(*10^3)$ | 1.0 | 1.1 | 8.4 | 5.4 | 1.8 |

| $K^s$ | System performance | | | |
|---|---|---|---|---|
| 0 | $e^{dis}$ | 0.0270 | $e^{pos}$ | 0.0090 |
| 40 | $e^{dis}$ | 0.0161 | $e^{pos}$ | 0.0177 |

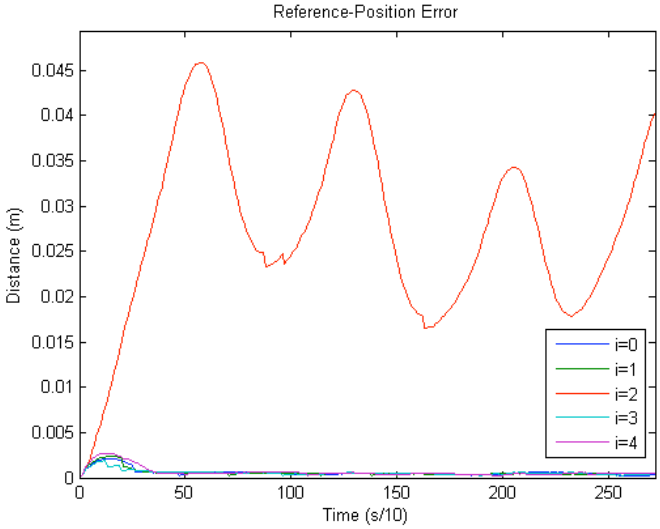



Figure 3.10: Reference-Position error with no physical link.

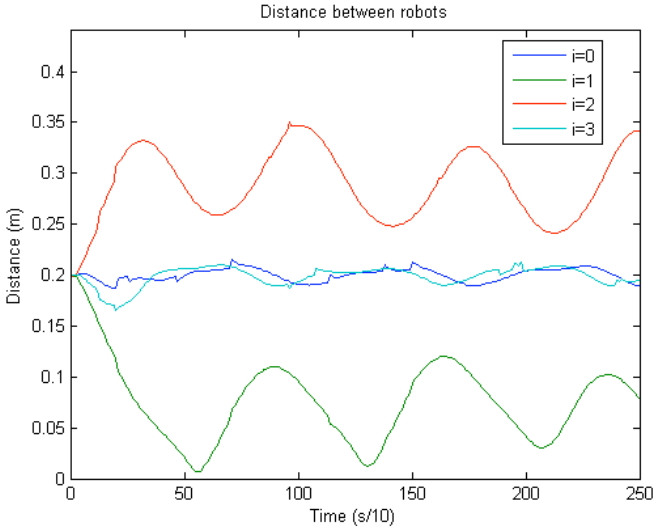



Figure 3.11: Distance between robots with no physical link.

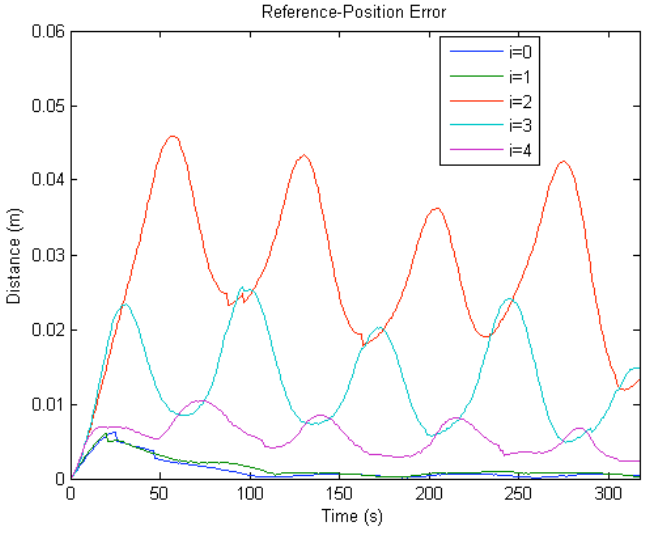



Figure 3.12: Reference-Position error with a physical link ($K = 40\ N * m$).

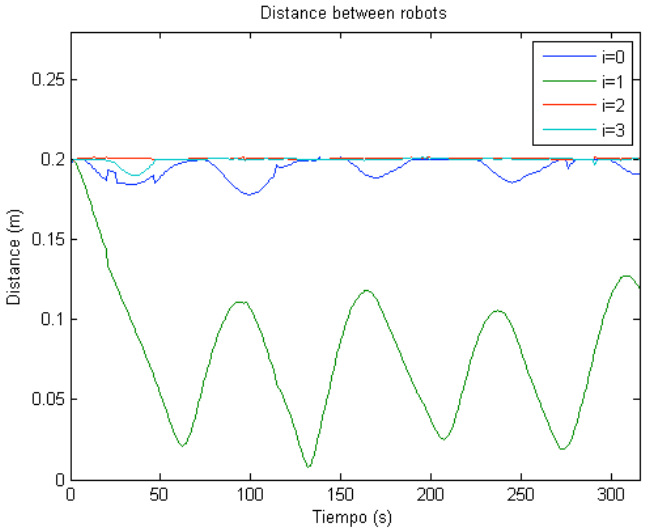



Figure 3.13: Distance between robots with a physical link ($K = 40\ N * m$).

## 3.5 Conclusions

In this chapter we have explored a way to justify the assertion that the non-rigid link between robot units in the L-MCRS introduces non-linear dynamic behaviors, that can not easily be accounted for by conventional control schemes. We have introduced a simplified model of the hose, allowing only elastic forces when the hose is stretched beyond its nominal length. We have defined a distributed asynchronous control system based on the consensus algorithm for the estimation of the coordination variable given by the position of the rear robot. Local control modules are Proportional Integral control algorithms relative to the reference position along the path. In the experiments, we have shown that the introduction of the hose has a definite effect on the resulting system dynamics. Slow robots that would be ignored by the control system if there is no linking element, drag the other robots and the control algorithm is unable to compensate for these deviations. The effect of the hose can be viewed in a positive perspective as adding robustness to the system, because all the robots are behaving in an homogeneous way despite the lack of performance of some of them.

# Chapter 4

# Reinforcement Learning

This chapter is devoted to the definition of Reinforcement Learning (RL) algorithms which we have applied to the autonomous learning of control strategies for the hose transportation system. The main advantage of RL approaches is that we do not need a precise knowledge of how a task must be performed in order to teach the system to do it. We only need to know when the task has been correctly performed, and the goal achieved. In this sense, the RL approach alleviates the burden from the teacher, which does not neet to propose model solutions, but requires longer exploration times for the system has to discover these solutions. In this chapter we review some fundamentals of RL, with an emphasis in the notorious Q-Learning approach which will applied in our experimental works.

The chapter is structured as follows: Section 4.1 provides an introduction. Section (4.2) comments some of the issues of applying RL to multicomponent robotic systems. Section 4.3 recalls basic defintinos of RL. Section 4.4 recalls the definition of Q-learning. Section 4.5 gives the definition of the Dyna-Q learning algorithm. Section 4.6 gives the definition of the TRQ-learning algorithm. Finally, Section 4.7gives a structural comparison of the algorithms on the basis of their flow diagram description.

## 4.1 Introduction

The paradigm of L-MCRS [9] exemplified by the task of carrying a hose from the origin point to a predefined destination using a collection of autonomous robots

attached to the hose has been introduced in Chapters 2 and 3. We have discussed the inherent problems of the system dynamics and the difficulties of developing control systems for them. A simplified spring-like model of the hose has been used in Chapter 3 to illustrate the problems that a control system designed for disconnected teams of robots would face when applied to a linked system [13]. As part of the on-going work of our research group, some efforts have been based on analytical detailed models [15, 10]. Knowing an accurate system model and the precise task goal, i.e. the exact path required of the hose, it is possible to derive iteratively the control commands minimizing the positioning error. However, real life implementation of this approach has strong demands: we need to have good sensory systems able to provide system localization with required accuracy, precise and accurate models of the system, and the planning problem must be exhaustively solved. All these demands have driven us to search for more flexible, adaptive ways to define control susystems for these L-MCRS. As a prototypical case we consider in the Chapter 5 the hose transportation problem consisting of moving the tip of the hose to a desired position while the other extreme of the hose is attached to a fixed point, a source. This is a simple formulation of the problem, and the results can serve as a starting point for further generalization.

Reinforcement Learning (RL) [38] is a category of computational learning methods that enable an agent to learn optimal policies from experience without being teach the desired response. Although there are many variants, there are certain shared elements: a *policy*, a *reward function*, a *value function* and the *model* of the environment. The *policy* describes the way an agent reacts to the perceived states deciding the action to be taken from the available ones. The *reward function* is the inmediate agent's perception of the response of the environment to its actions. It inherently sets the goals for the agent activity, which consists in obtaining the best environment's response. The *value function* is the long-term version of the reward function (*return*), that is, the total amount of rewards expected by the agent from a given perceived environment state. The *model* predicts the next environment state as a result of the action taken by agent. Model free methods, such as Q-learning, do not use an environment model. Learning processes providing experience to the agent can be continuous or episodic, that is, separate finite episodes.

In general, the specification of a RL to learn a given task requires the following concepts:

- State: the state has to capture the reality of the scenario in which the problem solution is being carried out. Its definition needs an equilibrium between the fidelity of the representation of the world and the quantity of the information that we have to deal with. The definition of the learning state may involve elements of the problem, as well as the dynamics of the system (i.e. the working space). In control processes it may include the control goal.

- Actions: they are the set of actions that the agent can perform in the world. Actions produce effects in the environment, inducing state changes that in return affect the agent.

- Reward system: it specifies the immediate reward that the agent perceives of the environment after doing any possible action. To completely specify a reinforcement system we have to establish the immediate rewards for different scenarios:

  - The goal is reached, the system state corresponds to the completion of the required task, i.e. the extreme of the hose attached to the mobile robot reaches the destination point. Usually, the reward value is positive.
  - A failure or forbidden situation occurs, the system state does not allow further processing. The system is stuck in an undesired state i.e. the mobile robot collides with the hose. Usually, the reward value is negative.
  - Other transitory states that are neither goal nor failures, usually the reward is null in this case.

There are three main families of RL algorithms: Dynamic Programming (DP), Monte-Carlo (MC) methods and Time Difference (TD) learning, each of them having its own strengths and weaknesses. The DP algorithms are mathematically well founded, but they are computationally expensive and require an accurate model of the environment, which is not always available. DP-based RL algorithms require complete knowledge of probability distributions of all possible state transitions, therefore this requirement limits their applicability to complex real environments. The MC methods and TD learning algorithms do not require a model of the environment, furthermore both can learn from experience, even from simulation of a simple model that allows to generate a sample

of the possible transitions among states. MC methods learn on an episode basis, that is, they need to know the actual final return of a sequence of actions in order to do some learning process. On the other hand, TD algorithms are able to deal with on-line learning tasks, that is, they do not need to know the actual final return of the episode, but only the estimated value one time step ahead, so that the state value update is made based on the prediction made one step ahead. Both MC and TD methods are known to converge to optimal control policies.

Q-Learning [?, ?, ?, 38] is a model free TD learning algorithm able to learn from on-line experience without requiring accurate knowledge of the environment. RL methods can be applied both in the real environment or in a simulated environment. As RL requires a huge amount of attempts to teach the system, whenever possible, it is better to use simulation to learn the system parameters. Simulation avoids tearing down the physical system and it is faster.

RL has been succesfully applied to develop control policies for robotic systems in the recent past. Equating agents to robots, training of decentralized control of MCRS can be viewed as an instance of Multi-Agent Reinforcement Learning (MARL) systems. We will consider cooperative Multi-Agent systems designed to maximize the collective utility of the system as a whole, fulfilling the task of hose transport in our application of intrests. We are not interested in competitive systems designed such that each agent only intends to maximize its individual utility[?, ?]. We do not envisage any way in which a pure competitive system may fulfill the hose transport task.

A RL approach is applied in [16] to guide a single robot in an environment with obstacles, using a model based on emotions to influence perception and provide the reinforcement/reward function. In [30] RL and learning through time techniques are used to achieve an obstacle-free path for a simple robot. A RL learned solution to the problem of model-free intelligent attitude control of aerospace launch vehicles is presented in [26]. They use RL due to the lack of a precise physical model. In [20] authors are given a robotic map that represents the world of the robot, where there are dynamical obstacles, to solve the path planning problem using an evolutionary algorithm which improves through experience. However, they need an *a priori* given detailed world model. Finally, in [21] authors deal with sophisticated tasks such as object manipulation, assembly tasks, or cooperative tasks with human workers. However, although there is no precise model, the objects mentioned in that paper are of fixed dimensions and neither allow deformations, nor exercise influence on the manipulators.

Further, [28] applies a RL method to the path-finding problem, [5] applies a quantum computation-inspired variant of Q-Learning to indoor robot navigation, [6] fuses a fuzzy inference system and a Q-Learning algorithm to derive a fuzzy control system, which yields in efficiency and adaptability. Q-Learning has been even applied to cooperative navigation in [27], but has never been applied in the presence of physical-links.

We have proposed Q-Learning [14] as the basic approach to learn L-MCRS controllers from experience. We have tested several ways of computing this reward, giving different values to the final state when the system ends in a state that can not be labeled as a failure or as having reached the goal. Results given in Chapter 5 were computed on the accurate simulation of L-MCRS developed[1] by our group [15, 10] based on the Geometrically Exact Dynamic Splines (GEDS) [40] approach to build dynamical model of uni-dimensional objects.

## 4.2 Issues of Reinforcement Learning for MCRS control

The main advantage of adopting the RL framework for the development of MCRS control algorithms is that it provides a systematic way to deal with the problem. It is sometimes easier to build the definition of the MDP modeling the system than the ad-hoc design and development of a control algorithm (even using supervised learning methods). However, applying RL algorithms to MCRS becomes a Multi-Agent RL problem, raising several strong issues. Coordination-related issues are specific to RL algorithms, others are inherited from the basic single-agent RL methods, which only may get worse in multi-agent configurations because of the added complexity of the system.

- *Resource scalability:* The intractable growth of memory requirements is the most serious limitation of the tabular representation of Q-matrices. In single-agent problems the size order of the table is $O(\mid S \times A \mid)$ and this gets even worse in most MARL algorithms, growing exponentially as the number of agents increase: $\mid S \times A^n \mid$. This problem is known as the *curse of dimensionality* and is the most serious limit to scale up the single agent Q-Learning. Besides, communication resources needed for RL also scale up combinatorially with the number of robots/agents. Hierarchical

[1] available at http://www.ehu.es/ccwintco/index.php/GIC-source-code-free-libre

solutions [?] can be considered to face this problem. Single-agent RL requires chosen actions to be transmitted to all agents at each time-step and multi-agent explicit-coordination mechanisms require even a bigger communication bandwidth for the agents to agree on a joint action.

- *Action Heterogeneity:* In standard RL formulation all actions span the same fixed amount of time. This is not a very realistic assumption in MCRS. Action are abstractions of operations performed with different electronic devices which usually require different amounts of time to perform equivalent actions. For example, if an action consists in the motion across a length of space, heterogeneous robotic units could require different amounts of time to complete the action. Furthermore, different actions may require wide differences in time in the same robotic unit, i.e. moving versus switching on/off a LED. Dealing with this time dimension means adding the complexity of synchronization on top of coordination.

- *Decentralized control:* A major issue towards achieving multi-agent coordination through RL is that the environment becomes non-stationary from the individual agent point of view because other agents' policies will change during the learning process. This is likely to produce oscillations and unexpected behaviors. This problem has been extensively studied as an *Stochastic Game*, leading to the concept of *Nash Equilibrium* [?]. If each agent follows an optimal policy relative to other agents' optimal policies, then the system is said to have reached Nash Equilibrium. However, there may exist more than one optimal policy achieving Nash equilibrium.

- *Control delay:* All on-line RL algorithms follow the same iterative pattern: observe the state, select an action and then issue the appropriate command to the actuators. This is completely safe in an ideal scenario where acquiring the state, executing the action selected and transmitting the command introduces no time delay, but in real life observation, communication and decision consume time and add complexity to the synchronization issue. I.e. coordination algorithms [?, ?] introduce complex communication protocols to agree on a joint action to be taken.

- *Robustness to partial and noisy sensor data:* Most approaches assume omniscient agents aware of all the sensed information, but this approach is unrealistic in complex environments facing serious limitations , i.e. physically limited and error-prone communications, sensor physical limitations

and/or obstacles. Furthermore, noisy measurements are likely to be perceived as different states in multi-agent environments. Therefore, algorithms that maximize the success possibilities in the presence of incomplete and noisy data are desired.

- *Convergence time:* Before the MCRS can be effectively controlled, the RL algorithm must explore the state-action space. The time required for this learning process can be unaffordable in real applications with large state-action spaces and thus, methods for a faster on-line learning are desirable. MARL systems may require even greater learning time because of the coordination requirements introduced.

## 4.3 Reinforcement Learning

Reinforcement Learning (RL) deals with the discovery of the optimal policy from the interaction between the agent and its environment by means of the rewards that the agent receives because of its actions. The RL approaches assume that the environment-agent system can be modelled as a Markov Decision Process (MDP) [2, 41] which is a discrete time stochastic process defined by the tuple $< S, A, T, R >$.

- $S$. The state space is the set of states in which the system can be found. Each $s \in S$ represents a different configuration of the entire system as perceived by the learning agent. The state space $S$ is defined by the cartesian of product of the range of values of the state variables $X$. State variables model agent internal states (e.g. its position), the task (e.g. the goal position), an environment condition, or the agent perception of the environment. Nevertheless, we can not assume that the set $S$ is completely known.

- $A$. The action repertoire is the set of feasible actions. Each $a \in A$ represents a different action that the learning agent can execute. $A_s$ represents the actions that can be executed in state $s \in S$.

- $T : S \times A_s \times S \rightarrow \mathbb{R}$. The state transition function provides the probability that action $a \in A_s$ taken in a given state $s \in S$ at time $t$ will lead to state $s' \in S$ in time $t + 1$:

$$T(s, a, s') = P\left(s_{t+1} = s' \,|s_t = s, a_t = a\right). \tag{4.1}$$

- $R: S \times A_s \rightarrow \mathbb{R}$. The immediate reward function. The environment gives immediate reward $r$ to the learning agent upon execution of action $a \in A_s$ on a given a state $s \in S$.

The MDP complies with the Markov Property:

$$T\left(s_{t+1}=s', r_{t+1}=r \,|s_t, a_t\right) = T\left(s_{t+1}=s', r_{t+1}=r \,|s_t, a_t, r_t, s_{t-1}, a_{t-1}, ..., r_1, s_0, a_0\right),$$

meaning that the next system state and corresponding reward are only dependent on the last system state and on the last taken action.

A policy $\pi: S \rightarrow A_s$ is a function implementing the probabilistic decision of executing action $a \in A_s$ in state $s \in S$. Time-Difference RL algorithms estimate the *value* of state $s$ (alternatively it can be viewed as the value of taking action $a$ in state $s$), as the expected accumulated rewards obtained from that state following a policy $\pi$. This value estimation, denoted $V^{\pi}(s)$, can be expressed as

$$V^{\pi}(s) = E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \mid s_t = s\right\},$$

where $\gamma \in [0,1]$ is a damping parameter, $r_t$ and $s_t$ are, respectively, the reward and state observed at time $t$, and $E_{\pi}$ represents the expectation conditioned to the agent following policy $\pi$. A policy $\pi$ is better than policy $\pi'$ if $V^{\pi}(s) > V^{\pi'}(s)$ for all $s \in S$ . There always exists [38] at least one optimal policy $\pi^*$ maximizing the state value $V^*$ satisfying

$$V^*(s) = \max_{a \in A_s}\left\{\sum_{s'} P(s,a,s')\left[R(s') + \gamma V^*(s')\right]\right\}.$$

The goal of RL is to learn an optimal policy for a given MDP or FMDP from experience in an autonomous process. Similarly, the value of taking an action $a$ in state $s$ is usually estimated using the state-action value function $Q^{\pi}(s,a)$, which can be written as

$$Q^{\pi}(s,a) = E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right\},$$

where $a_t$ represents the action taken at time-step $t$. The optimal state-action pair is, therefore,

$$Q^*(s,a) = \sum_{s'} P(s,a,s') \left[ R(s) + \gamma \max_{a'} Q^*(s',a') \right].$$

Regarding the policy, once we know the optimal policy $\pi^*$after learning, the obvious optimal course of action is to apply it. However, during the learning process we have only a guess about the optimal policy, embodied in the state-action value table $Q$. The learning process needs to continue the exploration of the state space. Applying the *greedy* policy consisting in selecting the action with the highest Q-value does not allow state space exploration, because the system is driven towards already known situations, found optimal in previous learning steps. This is the dilemma between exploration and exploitation. Exploration is needed to discover new optimal policies, exploitation refines the values of the already visited state-action pairs. The compromise between *exploration* and *exploitation* is solved using either a $\epsilon - greedy$ algorithm (a random action is selected with probability $\epsilon$ while the best action is chosen with probability $1-\epsilon$) or a Soft Max action selection based on a Boltzmann distribution:

$$\pi(s,a) = \frac{e^{Q(s,a)/\tau}}{\sum_{a' \in A} e^{Q(s,a')/\tau}}, \tag{4.2}$$

where $\tau$ is a positive *temperature* parameter, low values of $\tau$ increase the probability of taking actions with high Q-values, high-temperatures yield random action selections.

## 4.4 Q-Learning

In its simplest form, Q-Learning discovered by Watkins [38, 44, 45] is defined by the following iteration:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha \left[ r_{t+1} + \gamma \cdot \max_a Q(s_{t+1},a) - Q(s_t,a_t) \right], \tag{4.3}$$

where $a_t$ is the action taken at time $t$, $s_t$ is the state assumed at time $t$, $Q(s_t,a_t)$ represents the learned action-value discrete map at time $t$ and state $s_t$, $\alpha \in [0,1]$ is a step-size parameter that determines how new and old information are averaged, $r_{t+1}$ is the reward at time $t+1$, $a_t$ is the action taken at time $t$, and $\gamma \in [0,1]$ is a discount-rate parameter that indicates the importance of future rewards. The arrow in equation (4.3) means that the entry of the Q-table

**Algorithm 4.1** Q-learning algorithm

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize $s$
  Repeat (for each step of episode):
    Choose $a$ from $s$ using policy derived from $Q$
    Take action $a$, observe reward $r$ and new state $s'$
    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[ r_{t+1} + \gamma \underset{a}{max} Q\left(s_{t+1}, a\right) - Q\left(s_t, a_t\right) \right]$
    $s \leftarrow s'$
  until $s$ is terminal

corresponding to state-action pair $(s_t, a_t)$ is updated. Algorithm 4.1 represents the basic form of the learning algorithm. The learning process is composed of a succession of "episodes", each episode is a complete realization of the behavior of the system, that is, its evolution from an initial state until either the equilibrium state is reached or a stopping condition is met. Time variable $t$ denotes the time during an episode. The whole learning process is an iteration over the whole matrix $Q$ which evolves along the episodes. We avoid indexing it for notational simplicity.

Q-Learning requires setting specific parameters such as state space and action discretization: as the relationship between state and action is a discrete map, the resolution in the discretization of the state space and the actions is critical to obtain efficient and accurate realizations. Low resolution may allow fast realizations, losing accuracy. Conversely, high resolution state space discretization may hinder the realization of practical experiments. In [45] convergence of Q-learning in a stationary environment to an optimal policy with probability 1 is proved, as long as all state-actions pairs keep being updated (all actions are infinitely executed in all states), and $\alpha$ smoothly decreases during the learning process complying with the requirements of the stochastic gradient convergence. The main drawback of Q-Learning is that it has to execute many times all available actions in all possible states. Either when learning from real physical systems or from accurately simulated systems, this strategy is very expensive in terms of time, energy and wear of materials. This is a theoretically sensible condition, but hard to meet in practice because an agent may not be able to explore sufficient space to guarantee convergence. To relax this condition, Greedy in the Limit with Infinite Exploration (GLIE) policies were proposed **[?]**.

Initial approaches considered learning in a MCRS with $n$ units as a unique

learning process (a single agent) that had access to all environment variables and could control all robots applying simultaneous joint actions $A^n \equiv \{a_1, \ldots, a_n\}$, where $a_i \in A$ denotes the action applied by the $i^{th}$ robot. This kind of learning systems are known as *team learning* [?] and are not scalable to big robot teams for obvious reasons: the size needed to store the Q-table grows exponentially with the number of agents, even if we consider that the state space does not grow, because the action-state space size order is $O(\mid S \times A^n \mid)$. Besides, centralized control is less fault-tolerant than distributed control. *Concurrent learning* considers the presence of multiple *agents* implying that each of them is entitled to select its own actions and learn for itself how to maximize its local reward function. We have then an instance of Multi-Agent RL (MARL) [?]. In the cooperative MARL, a shared *global reward* is used as a quality assessment of the whole system behavior.

## 4.5 Dyna-Q algorithm

Dyna-Q is an unsupervised RL [38] algorithm, in fact, an enhanced Q-Learning algorithm which, besides using the environment's response to improve the value function and policy, builds and improves its own model of the world (model learning). It improves its current estimation of the state-action value function and the model of the environment in a specific bootstrapping phase, carried out in an internal loop executed $N$ times. The Dyna-Q process is described in Algorithm 4.2, where $\pi_\epsilon(s, Q)$ denotes a $\epsilon$-greedy policy. The algorithm builds a model of the system response storing the reward and next state observed after executing action $a$ in state $s$, denoted $Model(s,\ a)$ in the algorithm. After each real life (or simulation) state change, the Dyna-Q algorithm performs a number of repeating iterations of te Q-table updating based on the stored model values. These bootstraping iterations are intended to enhance the accuracy of the state-action value pairs without the burden of repeating actual real life episodes. Random pairs of already seen states and actions are drawn, their corresponding state transition and reward are obtained from the stored model and used to update the Q-table.

**Algorithm 4.2** Dyna-Q algorithm

Initialize $Q(s,\, a)$ and $Model(s,\, a)$ for all $s \in S$ and $a \in A(s)$ arbitrarily
Do forever:
  $s \leftarrow$current (non terminal) state
  $a \leftarrow \pi_\epsilon(s,\, Q)$
  Execute action $a$, observe resultant state $s'$ and reward $r$
  $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
  $Model(s,\, a) \leftarrow s',\, r$ (assuming deterministic environment)
  Repeat $N$ times:
    $s \leftarrow$random previously observed state
    $a \leftarrow$random action previously taken in $s$
    $s',\, r \leftarrow Model(s,\, a)$
    $Q(s,\, a) \leftarrow Q(s,\, a) + \alpha\left[r + \gamma \max_{a'} Q(s',\, a') - Q(s,\, a)\right]$

**Algorithm 4.3** TRQ-Learning algorithm

Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
  Initialize $s$
  Repeat (for each step of episode):
    Choose $a$ from $s$ using policy derived from $Q$
    If $T(s,a) \neq \emptyset$
      $s' \leftarrow T(s,a)$
      $r \leftarrow R(s,a)$
    else
      Take action $a$, observe reward $r$ and new state $s'$
      $T(s,a) \leftarrow s'$
      $R(s,a) \leftarrow r$
    $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma max_a Q(s',a) - Q(s,a)\right]$
    $s \leftarrow s'$
  until $s$ is terminal

## 4.6 TRQ-Learning algorithm

The TRQ-Learning algorithm enhances Q-Learning and Dyna-Q algorithms. Its process is described in Algorithm 4.3. The main idea is to maintain separate lookup tables for the state transitions and rewards obtained when executing action $a$ in each state $s$ which are filled as the learning proceeds. Assuming a deterministic environment, these tables may be used to recall the reached state and reward once the corresponding entry has been filled, without repeating the execution of the action in the real system or in accurate simulations, in the subsequent times that the same action $a$ is taken in the same state $s$.

There are three specific data structures in the TRQ-Learning algorithm:

- $Q(s, a)$: the state-action value function.
- $T(s, a)$: the state transition function of the FMDP, assuming a deterministic environment, stores the state $s'$ reached after executing the action $a$ in the state $s$.
- $R(s, a)$: the reward function of the FMDP, assuming a deterministic environment, stores reward $r$ obtained after executing the action $a$ in the state $s$. Often, we can benefit from a direct functional dependency of the state, such that $R(s, a) = f(T(s, a))$. In these circumstances we can avoid direct storage of the lookup table.

We can rewrite the updating rule of equation (4.3) as follows:

$$Q(s_t,\, a_t) \leftarrow Q(s_t,\, a_t) + \alpha \left[ R(s_t,\, a_t) + \gamma \cdot \max_a Q(T(s_t,\, a),\, a) - Q(s_t,\, a_t) \right]. \tag{4.4}$$

The convergence conditions for TRQ-Learning are the same as for Q-Learning, however, great time savings can be expected by the substitution of the action execution by querying a lookup table. The computational memory space required is at worst three times the cost of Q-Learning, and equal to Dyna-Q. Therefore, TRQ-Learning does not impose additional extreme conditions for its implementation.

## 4.7 A structural comparison of algorithms

We make a structural comparison of the algorithms over their block diagram representation of each state transition in an episode. Block diagrams must be

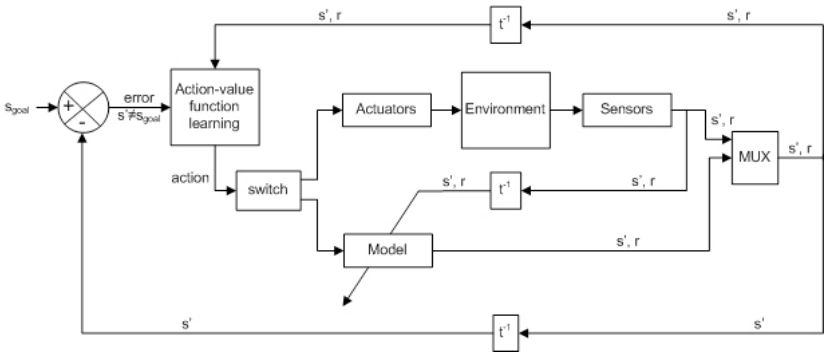



Figure 4.1: TRQ-Learning block diagram

read from left to right. Figure 4.1 shows the block diagram of the TRQ-Learning algorithm. The "action-value function learning" block representing the update rule of matrix $Q(s,a)$ activates a "switch" that either recalls the stored state transition $s'$ and reward $r$ through the "model" block or invokes the real system (or its simulation), through the "actuators" and subsequent blocks, to obtain $s'$, $r$, which, besides being returned to the "action-value function learning" block, are also feed to the model building block. The real system is invoked only when the actual response to the chosen action in the actual state is unknown. The TRQ-learning assumes that the system is deterministic.

Figure 4.2 contains the block diagram of the Q-Learning algorithm. Here the "action-value" block directly invokes the real system/simulation sequence of blocks to obtain the state transition and reward. The block diagram highlights the process of model building added to the TRQ-Learning algorithm. Time and real/computational costs are expected to be much smaller in the "model" branch. Moreover, it is expected that the process will go through the "model" branch with increasing frequency as the learning process and more state-action pairs are explored. Experience will increasingly reduce cost and time in TRQ-Learning relative to Q-Learning.

Figure 4.3 contains the block diagram of the Dyna-Q algorithm. A direct comparison with the TRQ-Learning block diagram in figure 4.1 highlights their differences. The Dyna-Q algorithm always invokes the real system/simulator to estimate the state transition and reward, thus it does not reduce the computational cost relative to Q-Learning. In fact, in some settings, the model bootstrapping, represented by the loop at the right end of the diagram between the model and the planner, can be an additional computational burden. For

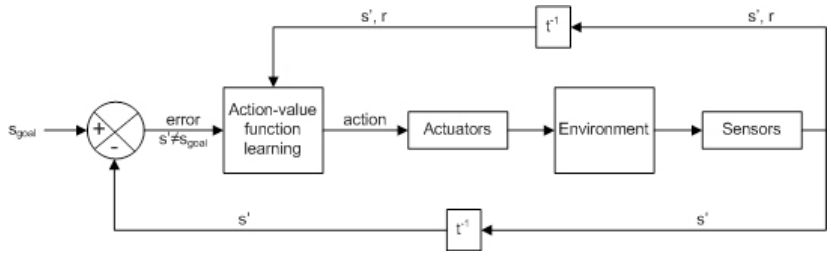



Figure 4.2: Q-Learning block diagram

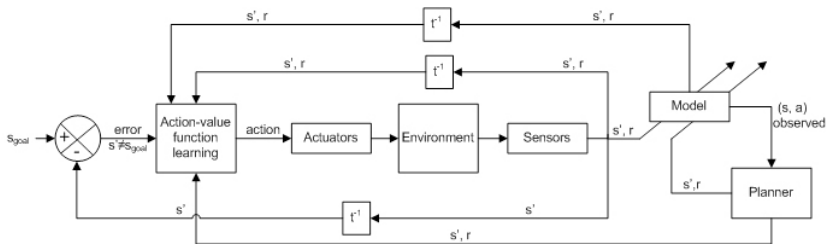



Figure 4.3: Dyna-Q block diagram

instance, in the case of large state space systems many bootstrapping cycles will be useless due to the sparsity of the model and the lack of information. In this situation, TRQ-Learning will improve steadily since the beginning of the learning process, while Dyna-Q wastes computational resources. Computational experiments have demonstrated that some elementary operations such as table updating require non-negligible time for large state spaces. For simulated systems, increased computational time devoted to simulation results in increased knowledge incorporation into the model.

# Chapter 5

# Experiments of hose transportation control with Reinforcement Learning

This chapter reports the experimental results of the application of Reinforcement Learning (RL) approaches to learn the control of a hose transportation system in the special case of moving the tip of the hose to a desired position while the other end of the hose is attached to a source in a fixed position. We describe the formulation of the Q-Learning system for the case of one and two robots attached to the hose. RL is based on the accurate system simulation based on the GEDS model described in Chapter 2.

The contents of the chapter are as follows: Section 5.1 gives an introduction. Section 5.2 gives the algorithm for initial state generation. Section 5.3 gives the first results on RL over a simple instance of the problem. Section 5.4 explores the accuracy sensitivity due to the changes in the definition of the state variables, the discretization resolution and the reward system. Section 5.5 studies the improvement introduced by adding some variables that embody predictions on action effects. Section 5.6 introduces the TRQ-learning in the single robot case. Section 5.7 summarizes the results for single robot system experimentation. Section 5.8 gives the specification of the learning for the two robot system. Section 5.9 gives the experimental results for two robot system. Finally, Section 5.10 gives some conclusions of the chapter.

## 5.1 Introduction

As a first step towards the general application of RL techniques to L-MCRS control we have chosen the hose transportation problem for a single robot: from an arbitrary initial configuration of the hose, a robot fixed to the hose tip is expected to carry it to a given position. Simulations have been carried out to provide the information for the the Q-Learning algorithm to avoid the main drawback of the experience-based learning algorithms: the huge amount of time required to realize the experiments in the real world. We have used an accurate simulation of the hose dynamics based on a Geometrically Exact Dynamic Splines (GEDS) model [10] as a substitute for the physical system realization.

Besides previous works reported by our research group [10, 13, 14], we have not found references in the literature to autonomous learning of the control of a system equivalent to the L-MCRS that we are dealing with in this chapter. Differential features of the system are:

- The lack of *a priori* model of the world. The model described in Chapter 2 is not included in the learned control system, it is used as a surrogate of the real physical system through simulation.
- The hose can be considered as a complex dynamical obstacle because it changes its position, size and shape while the robot carrying it is moving.
- The hose is not a purely passive object because there are stretching and bending forces on it due to its physical characteristics. So it exerts a force on the robot and limits its movements, thus the robot and the hose are interdependent subsystems.

We have also tested the TRQ-Learning algorithm introduced in Chapter 4. This algorithm has boosted convergence to an optimal policy through learning empirical models of both state probability transition and rewards. TRQ-Learning improves the results reported in [14] with the introduction of new state variables corresponding to feasible sensory information of the autonomous robot moving the tip of the hose.

The results on RL applied to learn the control of a hose transportation system reported in this chapter have been partially published previously. First, in [14] only one state model and one reward function have been used. Later, in [?] the same approximation is followed, but testing three different state models, and the reward function is modeled by nine different reward systems, with the

aim of studying the effect of these elements in the performance of the learning algorithm. This chapter provides additional results obtained after the ones reported in [14] and in [?], experimenting with a new state model that contributes to improve the results that have been obtained in these works .

## 5.2 Generation of initial hose configurations

The generation of initial states is a basic element of all the experiments reported in this chapter. It has been done trying to avoid extreme hose configuration from which evolution is unfeasible. Algorithm 5.1 contains a pseudo-code description of the procedure applied to generate an arbitrarily large set of initial hose configurations. The parameters of the algorithm 5.1 are:

- $\#Hose\,Points$: It is the number of corners of the hose. A typical value is 3.
- $\#Hoses$: It is the number of hose shapes that will be generated.
- $States\,per\,Hose$ : It is the number of different states that will be generated for each shape of hose.

All the $\#Hose$ hoses are attached to a fixed point which is the origin $(0, 0)$ of the working space. The total number of different states that will be generated is $\#Hoses \times States\,per\,Hose$. Some predicates that appear in the algorithm need some explanation.

1. $Smooth\,Angle$ is true if the angle between the $Actual\ Segment$ and $Previous\ Segment$ is in a given interval, meaning that there is no sharp bend of the hose.
2. $Is\,Cross$ is true if the $Actual\ Segment$ crosses any of the already generated $Hose\ Segments$.
3. $Collisionrisk\,(A, B)$ is true if some hose points fall in the rectangle whose diagonal vertices are points $A, B$

Figure 5.1 illustrates an initial configuration of the entire system, where $P_{ri}$ is the initial position of the robot and $P_d$ is the desired or goal position of the robot.

**Algorithm 5.1** Algorithm for the generation of learning initial states.

1. $Radius \leftarrow \frac{L}{\#Hose\,Points}$
2. repeat (until $\#Hoses$ hose patterns generated)
   (a) $PP \leftarrow (0,0)$
   (b) $Hose\,Segments \leftarrow \{\}$
   (c) $Hose\,Points \leftarrow \{(0,\,0)\}$
   (d) repeat (until $\#Hose\,Points$ points generated)
      i. $AP \leftarrow$ random point $(x,\,y)$ on the circumference $c(PP, Radius)$
      ii. $AS \leftarrow \overline{PP, AP}$ segment that links $PP$ and $AP$
      iii. if $(PP \neq (0,\,0))$ compute $Smooth\,Angle$ and $Is\,Cross$
      iv. if $(PP = (0,\,0) \vee (Smooth\,Angle \wedge \neg Is\,Cross))$
         A. $Hose\,Points \leftarrow Hose\,Points \cup \{AP\}$
         B. $PP \leftarrow AP$
         C. $Hose\,Segments \leftarrow Hose\,Segments \cup \{AS\}$
         D. $PS \leftarrow AS$
   (e) repeat (until $States\,per\,Hose$ distinct states are recorded)
      i. $P_d \leftarrow$ random point $(x,\,y)$ inside the $c\left((0,0),L\right)$
      ii. Compose the state as:
         - $P_r \leftarrow AP$ . The robot will be placed in the last $Actual\,Point$
         - $P_d$: the desired destination point
         - $i \leftarrow \overline{P_r P_d} \cap \overline{S} \neq \emptyset; \forall \overline{S} \in Hose\,Segments$
         - $c \leftarrow Collision\,risk\,(P_d, P_r)$
         - $v$ : a vector with a flag for each available action, indicating if collision happens in the case of the robot moving in that direction
      iii. if this state has not been recorded before (the same tuple $(P_r,\,P_d,\,i,\,c,\,v)$)
      - Record the state $(P_r,\,P_d,\,i,\,c,\,v)$
      - Record $Hose\,Points$
      - Record $Hose\,Segments$

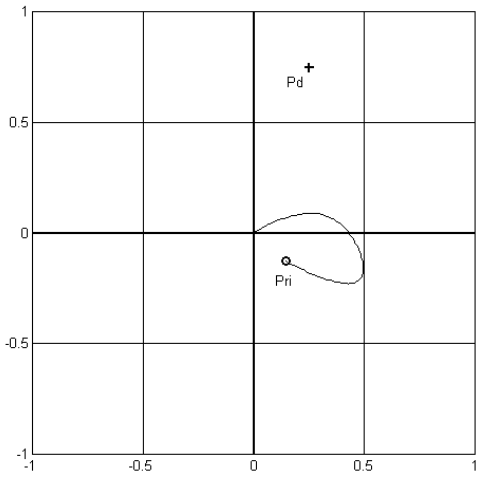



Figure 5.1: An instance of initial system configuration. $P_{ri}$ is the initial position of the robot driving the tip of the hose. $P_d$ is the goal position.

## 5.3 Initial Experimental design and results (NNW)

The system is composed of one hose segment attached to a fixed end (the source) and whose other end (the tip) is transported by a mobile robot attached to it. It is a single robot system. Figure 5.2 exemplifies several configurations of this system. The source position lies in the middle of the configuration space. The task for the robot is to bring the tip of the hose to a destination position. The working space where the tip-of-the-hose robot moves is a square of size $2 \times 2\text{m}^2$. The initial Q-learning experiment specific definitions are the following:

- State: we have defined the state varaiables as $X = (P_r, P_d, i)$, where
  - $P_r = (x_r, y_r)$ is the actual position of the tip-of-the-hose robot,
  - $P_d = (x_d, y_d)$ is the desired position of the tip-of-the-hose robot,
  - $i$ is a binary variable that indicates if the line $\overline{P_r P_d}$ intersects the hose. $i = 1$ means that there is such an intersection. This variable summarizes the perception of the system's state computed by the robot or some outside control system.

- $S$. The state space is partitioned into disjoint subsets $S = S^G \cup S^F \cup S^I$, where:

  - $S^G$ is the subset of states where the goal has been reached, i.e., when the mobile extreme of the hose has reached the desired position. They are absorbing states and a training episode finishes when any of these states is reached.
  - $S^F$ is the subset of the states that represent that a failure or a forbidden situation occurs, i.e., when any part of the unidimensional object has left the working area, or the mobile robot collides with any part of the hose. They are absorbing states ending training episodes.
  - $S^I$ is the subset of the inconclusive states, and they represent any other intermediate scenarios.

- Working space discretization: in order to follow with the simplest formulation of the problem we have considered a discretization step of $0,5$m. This discretization determines the cardinality of the universe of states that we are working with, and it determines the precision of the coordinates of the point $P_r$ and $P_d$ too. Our working space is, thus, partitioned into 16 boxes. These boxes are the minimum resolution for the placement of a robot. As the robot point $P_r$ can be in any of these 16 boxes, and the destination point $P_d$ can be in any of these 16 boxes too, there are 256 combinations. Also, the state has another boolean component called $i$, so there could be a maximum cardinality of 512 possible states.

- Actions: In our problem we can only interact with the scenario using the mobile robot to change the position of the tip-of-the-hose, so the actions are the possible motion directions of the robot. We have chosen a small set of only four actions: $A = \{\, North,\, South,\, East,\, West \,\}$, meaning that the robot will move in this direction for a length equivalent to the size of the resolution box.

- Reward system: We have used a simple reward system, that gives a positive value to the agent when it reaches goal, a negative when the agent fails to reach the goal, and zero value when decision is postponed:

$$r \leftarrow \begin{cases} +1 & if \text{ goal is reached} \\ -1 & if \text{ failure occurs} \\ 0 & else \end{cases} .$$

  The condition reaching the goal is equivalent to "reaching the same box

where the goal is located". As the motion of the tip-of-the-hose robot is of fixed step-size, it is in general impossible to meet a predefied goal point with arbitrary precision.

- $\alpha$: $[0 < \alpha \leq 1]$, as we suppose that we are working in a deterministic environment we can assume that the value of this parameter is 1, so the Q-table update expression simplifies to the following expression:

$$Q\left(s_t, a_t\right) \leftarrow r_{t+1} + \gamma \cdot \max_a Q\left(s_{t+1}, a\right)$$

.

- $\gamma$: $[0 < \gamma \leq 1]$, we have set this value to 0.9.

- Action selection: we apply an $\epsilon$-greedy policy. This policy is based on the existence of a parameter $\epsilon$ that establishes the equilibrium between the use of the known information (exploitation) and the discovery of new information (exploration), and we have set this value as 0.2. This means that in each step of each episode, with the system being in the state $s$, we choose the action $a$ with this criterion:

$$a \leftarrow \begin{cases} \max_{a'} Q\left(s,\, a'\right) & \text{with probability } (1-\epsilon) \\ \text{any } a' \in A & \text{with probability } \epsilon \end{cases}.$$

- Generation of the initial state: it amounts to the problem of generating a feasible configuration of the hose. To that end, we generate the positions of the spline control points in order from the working space origin (the source) outwards. We generate 10 control points, ensuring that the resulting GEDS will not have excesive bending or streching. Each episode starts from randomly generated configuration of the hose.

This initial experiment consisted of 76e+6 episodes, and performance was measured applying the learned state-action value $Q$-table to fresh unseen 1.000 episodes. The test episodes are independent of the episodes used for training of the system, avoiding circularity issues in the training-validation process. The success rate, i.e. the percentage of episodes where the robot reaches the goal in the test episodes, is 73%. The 0.7% of the test episodes concluded because the maximum allowed step count was reached. Finally, 26.3% of the test episodes

failed either because the robot collided with the hose or because the whole system reached a non-feasible position.

In order to illustrate the behavior achieved by Q-learning in this initial experiment, we have chosen a difficult initial configuration in which the hose is placed between $P_r$ and $P_d$ . Figures 5.2 and 5.3 show two instances of successful episodes, where the succesive $P_r$ positions of the robot moving the tip-of-the-hose after each of the actions taken during the episode. The initial hose conguration corresponds to the continuous line. All the intermediate hose configurations, until the robot reached the desired cell $P_d$, are shown as dotted lines. It can be easily appreciated how the robot avoids colliding with the hose by taking an initially suboptimal strategy (i.e. going away from the goal position). Figure 5.4 shows the plot of the evolution of the test episodes along the RL process. Each 1000 training episodes, the learning was stopped and 100 test episodes are executed. The plot shows the ratio of successful episodes (green), failed episodes (red) and the inconcluside episodes (blue).

## 5.4 Effect of the state and reward definition

In this section we report results on the sensitivity of the RL process to changes in the definition of the MDP state and in the reward function. All remaining components of the model are the same of previous section. The figure 5.5 shows the order in which we have considered the experimental parameters to perform the experimental design. More detailed design is presented in the tree shown in figure 5.6, following a path from the root to a leaf of the tree gives an experiment parameter setting. The specific new definitions of the Q-learning experiment realized are the following:

- State: we have defined the state using three alternative models: $X = (P_r, P_d, i)$, $X = (P_r, P_d, i, c)$ and $X = (P_r, P_d, i, P_1, P_2)$, where
  - $P_r = (x_r, y_r)$ is the actual position of the tip-of-the-hose robot.
  - $P_d = (x_d, y_d)$ is the desired position of the tip-of-the-hose robot, the goal.
  - $i$ is a binary variable that indicates if the line $\overline{P_r P_d}$ intersects the hose. $i = 1$ means that there is an intersection. This variable models a perceptual process that allows to decide if the hose is an obstacle to accomplish the task.

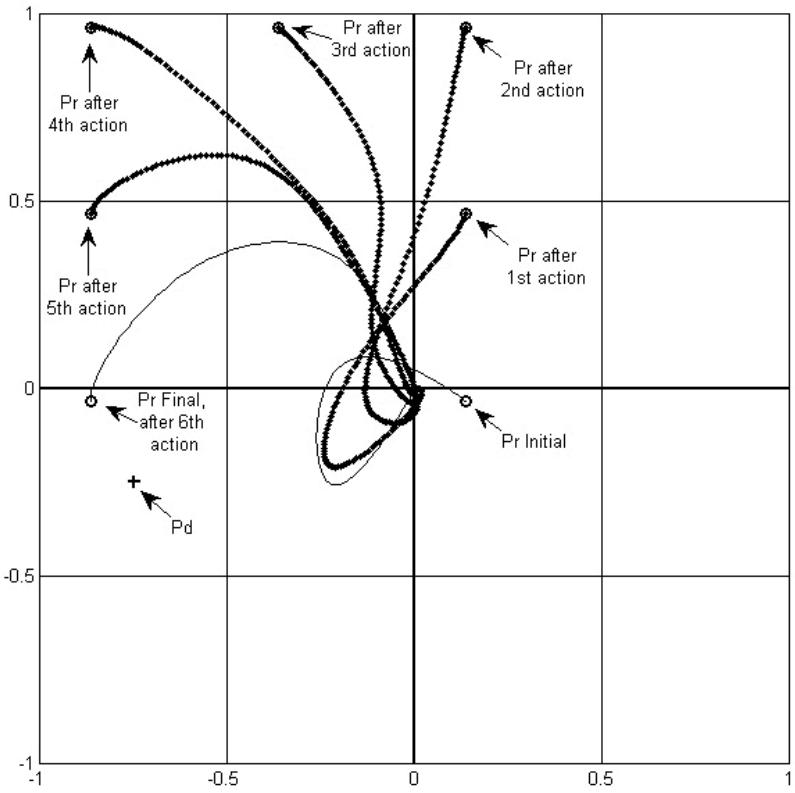



Figure 5.2: Example of a successful test episode in the initial experimentation with RL for the single robot hose deployment system.

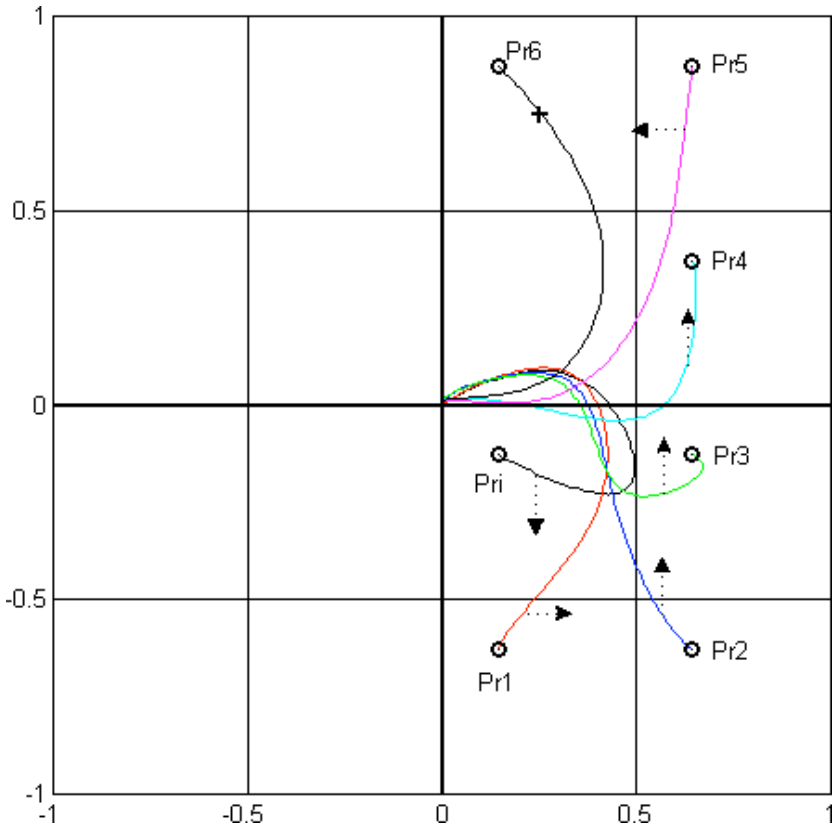



Figure 5.3: The evolution of the hose in a successful episode where the tip reaches the goal. Arrows are used to indicate the motion of the hose.

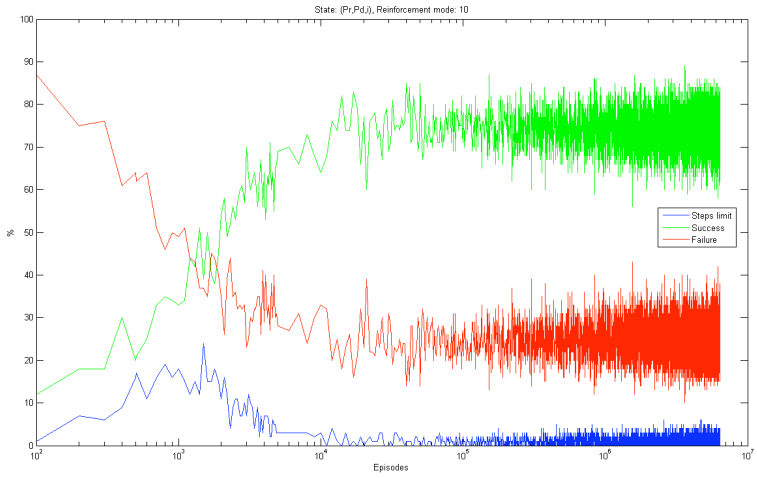



Figure 5.4: Evolution of the RL results for the basic experiment settings

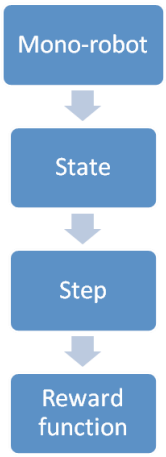



Figure 5.5: Hierarchy of the experimental design. We select the state definition, discretization step and type of reward function in order to obtain the experiment parameter combination.

| reward system code: 10 | reward system code: 60 |
|---|---|
| $r \leftarrow \begin{cases} +1 & if\ s \in S^G \\ -1 & if\ s \in S^F \\ 0 & if\ s \in S^I \end{cases}$ | $r \leftarrow \begin{cases} +100 & if\ s \in S^G \\ -100 & if\ s \in S^F \\ f^{60}(\mathbf{p}_r, \mathbf{p}_g) & if\ s \in S^I \end{cases}$ |
| reward system code: 20 | reward system code: 70 |
| $r \leftarrow \begin{cases} +100 & if\ s \in S^G \\ -100 & if\ s \in S^F \\ 0 & if\ s \in S^I \end{cases}$ | $r \leftarrow \begin{cases} +100 & if\ s \in S^G \\ -1000 & if\ s \in S^F \\ f^{70}(\mathbf{p}_r, \mathbf{p}_g, i, c) & if\ s \in S^I \end{cases}$ |
| reward system code: 30 | reward system code: 80 |
| $r \leftarrow \begin{cases} +100 & if\ s \in S^G \\ -1000 & if\ s \in S^F \\ f^{30}(\mathbf{p}_r, \mathbf{p}_g, i, c) & if\ s \in S^I \end{cases}$ | $r \leftarrow \begin{cases} +100 & if\ s \in S^G \\ -100 & if\ s \in S^F \\ f^{80}(\mathbf{p}_r, \mathbf{p}_g, i, c) & if\ s \in S^I \end{cases}$ |
| reward system code: 40 | reward system code: 90 |
| $r \leftarrow \begin{cases} +100 & if\ s \in S^G \\ -100 & if\ s \in S^F \\ f^{40}(\mathbf{p}_r, \mathbf{p}_g, i, c) & if\ s \in S^I \end{cases}$ | $r \leftarrow \begin{cases} -1000 & if\ s \in S^G \\ +1000 & if\ s \in S^F \\ f^{90}(\mathbf{p}_r, \mathbf{p}_g, i) & if\ s \in S^I \end{cases}$ |
| reward system code: 50 | |
| $r \leftarrow \begin{cases} +1 & if\ s \in S^G \\ 0 & if\ s \in S^F \\ 0 & if\ s \in S^I \end{cases}$ | |
| $f^{30}(\mathbf{p}_r, \mathbf{p}_g, i, c) = -\left(\frac{d(\mathbf{p}_r, \mathbf{p}_g)}{10} + 10i + 10c\right)$ | |
| $f^{40}(\mathbf{p}_r, \mathbf{p}_g, i, c) = \left(100 - \frac{d(\mathbf{p}_r, \mathbf{p}_g)}{2}\right) - 10i - 10c$ | |
| $f^{60}(\mathbf{p}_r, \mathbf{p}_g) = \left(100 - \frac{d(\mathbf{p}_r, \mathbf{p}_g)}{2}\right)$ | |
| $f^{70}(\mathbf{p}_r, \mathbf{p}_g, i, c) = \left(100 - \frac{d(\mathbf{p}_r, \mathbf{p}_g)}{2}\right) - 10i - 10c$ | |
| $f^{80}(\mathbf{p}_r, \mathbf{p}_g, i, c) = -\left(\frac{d(\mathbf{p}_r, \mathbf{p}_g)}{10} + 10i + 10c\right)$ | |
| $f^{90}(\mathbf{p}_r, \mathbf{p}_g, i) = d(\mathbf{p}_r, \mathbf{p}_g) + 100i$ | |

Table 5.1: Reward systems for single robot systems for hose transportation. Bottom rows have the definition of specific distance functions.

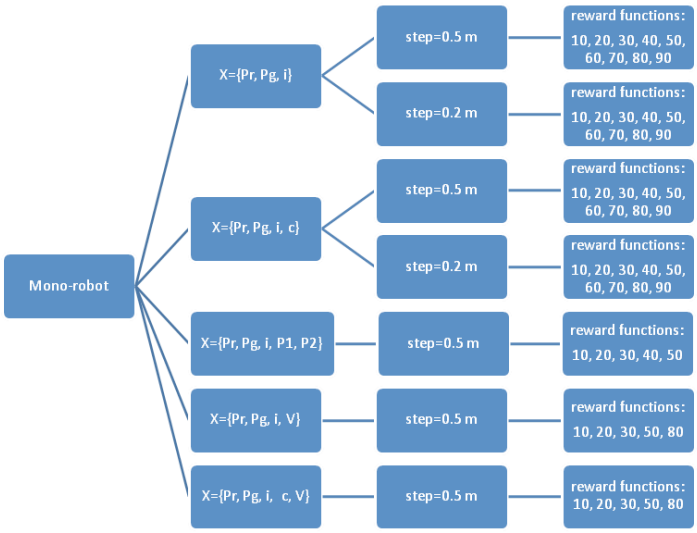
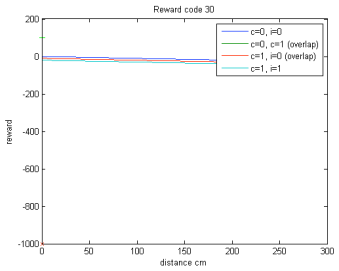


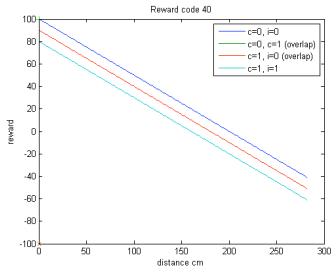

Figure 5.6: The tree of the experimental parameter value combinations tested

- $c$ is a binary variable that indicates if the box with corners $P_r$ and $P_d$ intersects the hose. $c = 1$ means that there is an intersection. This is a more sophisticated perceptual process, that implies a more detailed perception of the hose deployment than the one needed to compute $i$.
- $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ are two points of the hose that are uniformly distributed from one end to the other end.

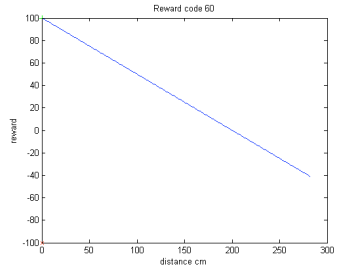
- Reward system: We have used several reward systems. In table 5.1 we present the formalization of all the reward systems that we have used. In figure 5.7 we presente the graphical representation of these rewards as a function of the distance to the goal position.
  - Reward systems 10 and 20: both give a positive reward when reaching the goal, negative when failing and nothing if the end state is inconclusive.
  - Reward system 50: only gives positive reward when reaching the goal.
  - The remaining reward systems give positive reward when reaching the goal, negative when failing and for the inconclusive states, a function of the actual distance between the hose tip and the goal. In some



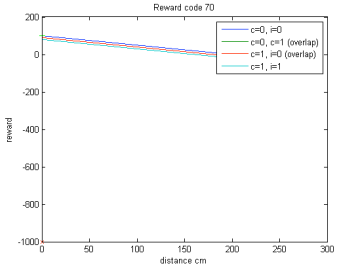

(a) Reward function 30



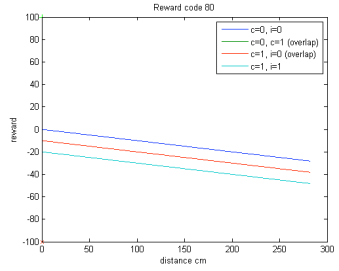

(b) Reward function 40



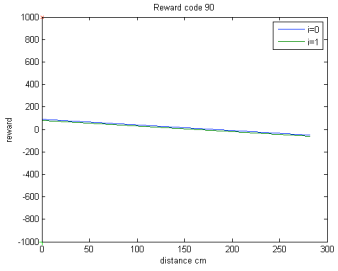

(c) Reward function 60




(d) Reward function 70




(e) Reward function 80




(f) Reward function 90

Figure 5.7: Plots of the reward response of reward systems as a function of distance to the goal position.

cases the reward function is also function of the binary variables $c$ and $i$.

### 5.4.1 Experimental results

The computational results give a systematic exploration of the combinations of state definition, reward system and discretization step. Besides, we have obtained numerical values of the results with different training time (expressed in terms of training episodes) in order to compare the learning of the same systems varying this parameter. The number of total episodes that we have carried out with each combination in the training phase is in table 5.2. In this way we have obtained the results that are shown in figure 5.8 and figure 5.9.

In these figures we show, for each combination of reward system and state model (with each different discretization step), the percentage of episodes where the robot reaches the goal in figure 5.8, and the percentage of episodes concluded because the maximum allowed step count was reached in figure 5.9. For each combination we show the results obtained in the test phase with 100 different initial configurations.

The best results correspond to the reward system code 20 with the state defined as $X = (P_r, P_d, i, P_1, P_2)$, the success rate, i.e. the percentage of episodes where the robot reaches the goal, is 77% of the test episodes. The 2% of the test episodes concluded because the maximum allowed step count was reached. Finally, 21% of the test episodes failed either because the robot collided with the hose or because the whole system reached a non-feasible position. The worst results come from the reward strategies 40, 60, 70 and 90 that give some combination of the distance to the goal and the binary variables as the reward function in the inconclusive state. They have the worst accuracy results and the higher number of simulations ended because they reached the step limit. The reward systems that gave a null reward or pure negative reward in terms of distance to the goal in the inconclusive final states were the ones with better results, regardless of the definition of the state, which is indicative of the robustness of the approach.

In general, we can see that increasing the training episodes the result in the test phase improve slightly in two ways: firstly, the number of episodes that finish with success have experienced a slight increase. On the other hand, the number of episodes that fail decrease to increase the number of those that finish because the maximum allowed step count was reached.

| | state variables | | | | |
|---|---|---|---|---|---|
| | $X = (P_y, P_d, i)$ | | $X = (P_y, P_d, i, c)$ | | $X = (P_r, P_d, i, P_1, P_2)$ |
| reward system | $\triangle s$ $0'5\,m.$ | $\triangle s$ $0'2\,m.$ | $\triangle s$ $0'5\,m.$ | $\triangle s$ $0'2\,m.$ | $\triangle s$ $0'5\,m.$ |
| 10 | 6.410 | 1.740 | 5.920 | 1.410 | 9.830 |
| 20 | 6.410 | 460 | 6.490 | 370 | 10.260 |
| 30 | 6.070 | 1.900 | 6.700 | 1.510 | 12.360 |
| 40 | 4.530 | 780 | 4.440 | 650 | 12.480 |
| 50 | 7.330 | 2.260 | 7.540 | 1.660 | 22.620 |
| 60 | 22.650 | 1.010 | 20.470 | 750 | |
| 70 | 23.290 | 1.090 | 20.450 | 620 | |
| 80 | 34.490 | 2.350 | 31.270 | 1.940 | |
| 90 | 37.970 | 2.810 | 36.430 | 1.980 | |

Table 5.2: Total episodes of the training phase (thousands of episodes)

Figure 5.10 shows the plot of the test episodes during the system training (training stopped each 1000 episodes for a 100 test episodes). The system reaches good results after $10^4$ episodes, after that the learning oscillates around the average values. Inconclusive final states are almost null. However the number of failed final states does not go to zero.

- Another question arises about whether the discretization step is relevant or not for the learning process of the agent. We have carried out experiments with the two discretization steps of $\triangle s = 0'5\,m.$ and $\triangle s = 0'2\,m.$ with the two state models of $X = \{\mathbf{p}_r, \mathbf{p}_g, i\}$ and $X = \{\mathbf{p}_r, \mathbf{p}_g, i, c\}$. These results can be seen in figure 5.11 and figure 5.12, and we have realized that using the same state model, the same reward system and a learning process of similar duration, these conclusions can be extracted:

- For the $X = \{\mathbf{p}_r, \mathbf{p}_g, i\}$ and the reward system $r3$, the discretization step $\triangle s = 0'2\,m.$ obtains the best results in terms of $\%\,goals$.

  - For the $X = \{\mathbf{p}_r, \mathbf{p}_g, i, c\}$ and the reward systems $r3$ and $r5$, the discretization step $\triangle s = 0'2\,m.$ obtains the best results in terms of $\%\,goals$.

  - For the remaining cases, the the discretization step $\triangle s = 0'5\,m.$ obtains the best results in terms of $\%\,goals$.

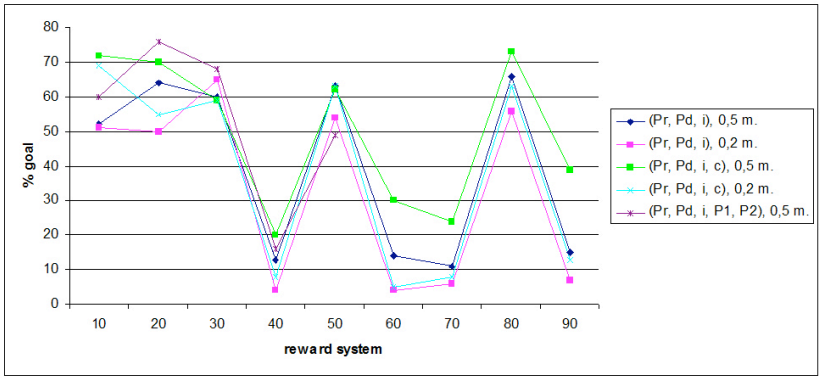



Figure 5.8: Percentage of successful runs (reaching goal) obtained in test phase with each reward system and state definition over a hundred simulations per combination of parameters.

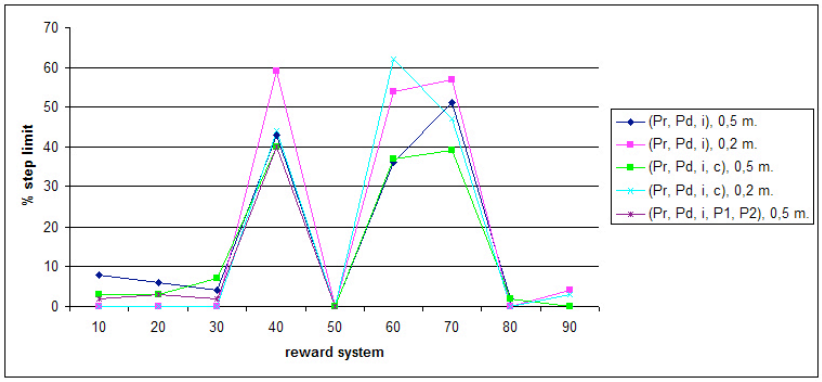



Figure 5.9: Percentage of runs terminated because they reached the limit number of steps obtained in test phase with each reward system and state definition over a hundred simulations per combination of parameters.

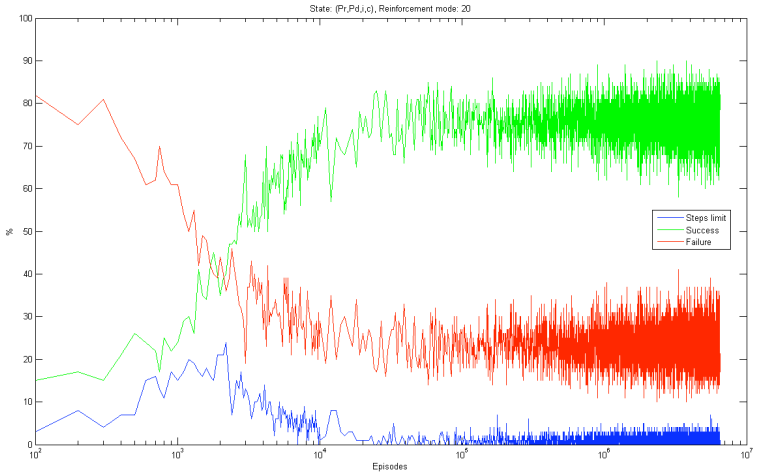



Figure 5.10: Evolution of the test episodes for the best combination of rewards and state definitions

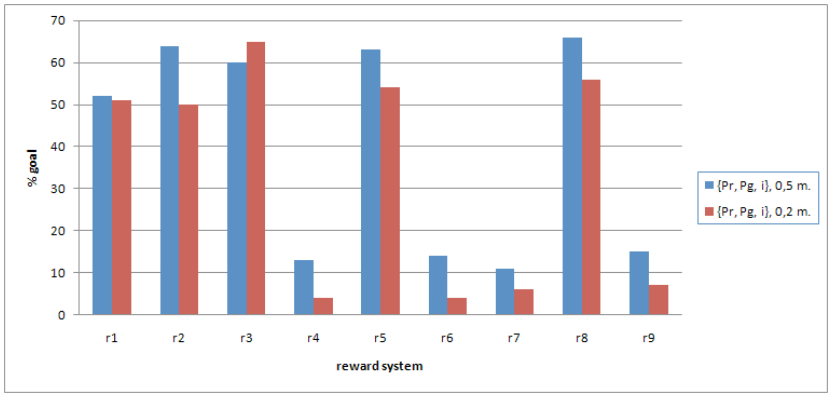



Figure 5.11: % goals obtained with the same state model $X = \{\mathbf{p}_r,\ \mathbf{p}_g,\ i\}$ using as discretization step $\triangle s = 0'5\,m.$ and $\triangle s = 0'2\,m.$, using the nine distinct reward systems $r1$ to $r9$. Except in one case, a certain superiority of the discretization step $\triangle s = 0'5\,m.$ can be appreciated.

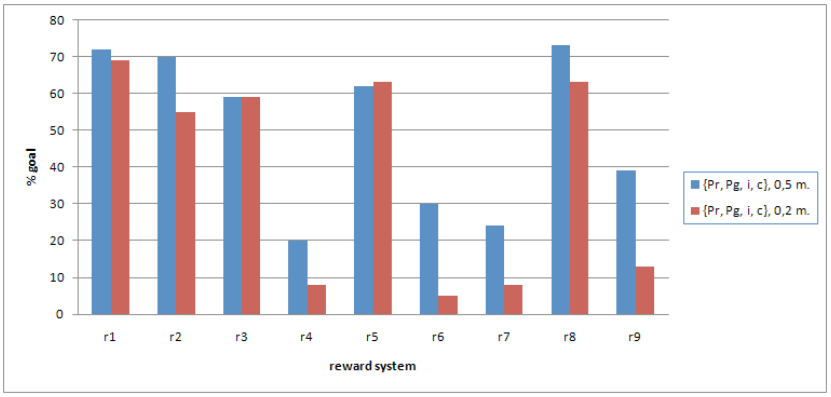



Figure 5.12: % goals obtained with the same state model $X = \{\mathbf{p}_r, \mathbf{p}_g, i, c\}$ using as discretization step $\triangle s = 0'5\,m.$ and $\triangle s = 0'2\,m.$, using the nine distinct reward systems $r1$ to $r9$. Except in two cases, a certain superiority of the discretization step $\triangle s = 0'5\,m.$ can be appreciated.

## 5.5 The value of prediction

In this section we consider an specific definition of the system variables which include some predictions of the effects of the actions. The new definition of the state variables is $X = (P_r, P_g, i, V)$, where $V$: it is a collection of logical variables, one for each feasible action, that are true if there will be collision of the mobile extreme of the hose with itself after performing the corresponding action. This variable involves some limited predictive ability of the robot, which allows it to detect the danger one step ahead.

The results obtained with the combination of the state model $X = \{\mathbf{p}_r, \mathbf{p}_g, i, \mathbf{V}_r\}$ and the reward system $r1$, improve the results that were reported in the previous section. In this case, the success rate, i.e. the percentage of episodes where the robot reaches the goal, is 79% of the test episodes. The 2% of the test episodes concluded because the maximum allowed step count was reached. Finally, 19% of the test episodes failed either because the robot collided with the hose or because the whole system reached a non-feasible position.

The question of the influence of the reward systems in the learning process is always present. In figure 5.13, we show the success rate measured in percentage of goals reached with four different state models, nine different reward systems and the discretization step $\triangle s = 0'5\,m.$ All the combinations have been not simulated due to the high computational burden involved. In general, the more complete is the definition of the state model, the better are the results. However,

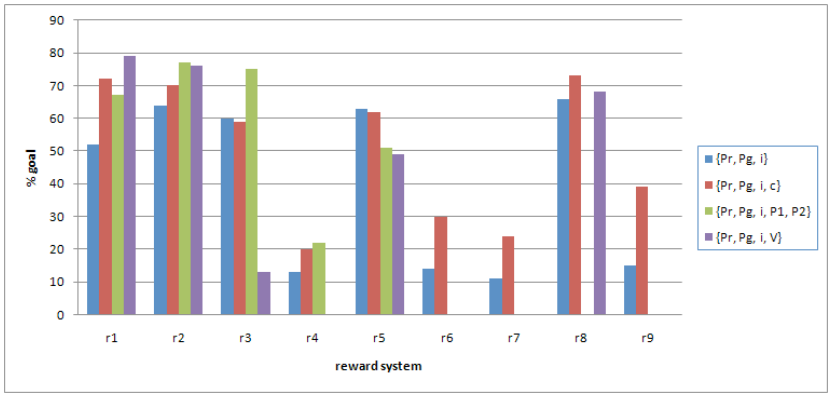



Figure 5.13: Percentage of succesful test episodes (%goal) obtained with four different state models, nine different reward systems and the discretization step $\triangle s = 0'5\, m$. No all the combinations have been simulated. In general, the more complete is the definition of the state model, the better are the results.

we can see that there are some cases where although the model is more complete, the result are no as goods as we could have expected. This happens in reward systems $r3$, $r4$ and $r8$. Likely, the reason for this is that the more complete the state model definition is, the wider is the search space. This is an open issue that needs more research.

## 5.6 Experimental results of TRQ-Learning

The system consists of an unidimensional object (a hose) that has one end attached to a fixed point (which is set as the middle point of the ground working space), and the other end (the tip) is transported by a mobile robot. The learning task for the learning agent (the robot) is to bring the tip of the unidimensional object to an arbitrarily designated destination point. The working space is a square of $2 \times 2$ m$^2$. We have used the procedure specified in the Section 5.2 to generate a number of different initial system configurations to be used as initial states in the TRQ-Learning algorithm. Besides, states found during the learning process are incorporated to this pool of initial states. The TRQ-Learning is applied to two definitions of the state variables $X^{(1)} = \{P_r,\, P_d,\, i,\, V\}$ and $X^{(2)} = \{P_r,\, P_d,\, i,\, c,\, V\}$. The working space has been discretized into squares of size $0.5 \times 0.5\ m^2$, therefore all spatial variables are given in the corresponding discrete coordinates.

For this specific experiment we have defined the immediate reward function according to the state that has been reached in each moment:

$$R(s,\, a) \leftarrow \begin{cases} r_g & \arg\max_{s'} \{T(s, a, s')\} \in S^G \\ r_f & \arg\max_{s'} \{T(s, a, s')\} \in S^F \\ r_i & \arg\max_{s'} \{T(s, a, s')\} \in S^I \end{cases}, \tag{5.1}$$

where $r_g = 1$, $r_f = -1$ and $r_i = 0$. The control system resulting from the TRQ-Learning process will be adaptative in the following sense. If there is an error performing any action (problems with the movement of the robot, steering, etc.) and the system reaches any state different from the predicted by the learned model $T(s,\, a)$, this new state will be already considered in the Q-table and the control algorithm will act consequently from that accidentally reached state applying an optimal policy. Specific parameters of TRQ-Learning algorithm are adjusted as follows:

- The action selection policy is an $\epsilon$-greedy policy, with $\epsilon = 0.2$.

$$\pi(s,\, a) \leftarrow \begin{cases} \max_{a'} Q(s,\, a') & \text{with probability } (1-\epsilon) \\ any\, a' \in A & \text{with probability } \epsilon \end{cases}. \tag{5.2}$$

- Episodes. The maximum length of an episode is set to 10 steps.

In table 5.3 we show the validation results. Validation has been performed as follows: after the TRQ-Learning process, the obtained state-action $Q$-table was used to solve the problem for 1000 test episodes starting from random initial configurations, applying the optimal action selection policy derived from the learned $Q$. In table 5.3 success corresponds to the percentage of successful validation episodes achieving the goal position, fail corresponds to the percentage of validation episodes ending in a fail state, and inconclusive corresponds to the percentage of test episodes that ended because the validation episode reached the limit number of steps. The second row gives the number of training episodes for each case, in thousands of episodes. The third row gives a measurement of the computational time in thousands of seconds. We denote $X^{(0)}$ the set of state variables considered in [14].

From table 5.3 it can be appreciated that the new state variables definition improves the results of [14] with much less computation needed (in terms of

| | $X^{(0)}$ | $X^{(1)}$ | $X^{(2)}$ |
|---|---|---|---|
| #training episodes ($10^3$) | 76.000 | 17.390 | 14.260 |
| comp. time ($10^3 s$) | 623 | 21 | 18 |
| % success | 73 % | 79 % | 92 % |
| % fail | 26.3 % | 19 % | 8 % |
| % inconclusive | 0.7 % | 2 % | 0 % |

Table 5.3: Performance evaluation of the state variable sets: previous works $X^{(0)} = \{P_r, P_d, i\}$ with Q-Learning, new state variable sets $X^{(1)} = \{P_r, P_d, i, V\}$ and $X^{(2)} = \{P_r, P_d, i, c, V\}$ with TRQ-Learning.

episodes) for the training of the system. We obtain up to a 92% success if we use information about the effect of the actions in the state. Therefore, incorporating an elementary prediction of the potential for failure allowing to avoid action sequences leading to failing states is a major improvement of the control system. Besides, this prediction ability is learned by the agent while performing the TRQ-Learning process. This improvement is obtained with little increase of the computational requirements as the state space size is mostly dependent on the spatial discretization resolution, and the number of episodes needed to reach peak convergence are in fact reduced for the innovative state variable configuration. In addition, besides the fact that fewer episodes are necessary to reach better results, each episode needs less computational time because on average the result of every action is calculated in less time, by the use of the on-the-fly learned model.

## 5.7 Summarizing results for single robot

In this section we give summary results for the single robot case. Figure 5.14 contains the plots of the success rates (the percentage of goals reached in the test episodes) for the different reward systems, each curve corresponding to a state variable definition and working space discretization resolution. Regarding the reward systems, rewards #50 and #80 are consistently good at all the combinations of policies and discretization, while system #40 and #90 are a consistent disaster. Reward systems #10 and #20 have wide variability depending on the discretization resolution. The inclusion of the predictive variables $V$ gives the best results. Figure 5.15 gives the rate of failure (percentage of failing final states reached in the test episodes). Suprisingly, this failure rate is rather homogenous across reward systems, but for the #90 system which really bad.

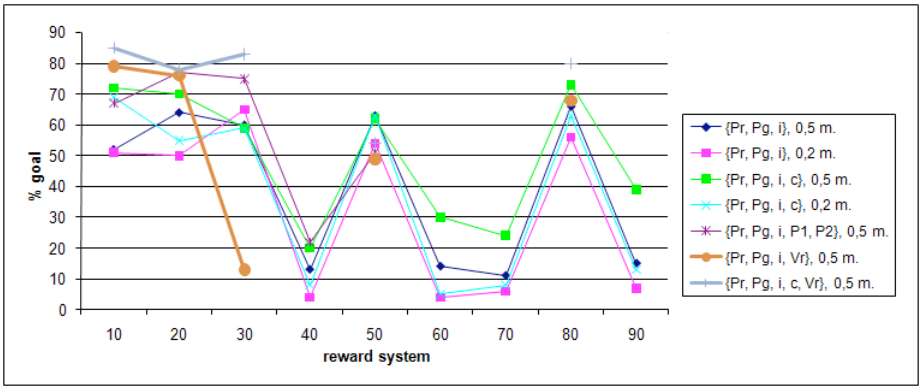



Figure 5.14: Summary success rate for all state variable definitions and discretization step relative to the reward system

Figure 5.16 contains the plots corresponding to the rate of inconclusive states reached at the end of the test episodes. It can be appreciated that most of the lack of success of reward systems #40, #60 and #70 is due to the lackof convergence, maning that the system does not arrive to any final state. On the other hand, sysem #90 is simply bad without excuse.

Trying to ascertain the effect of the state definition we plot the success rate of the state definition combined with the discretization step for the diverse rewards in figure 5.17. It can be appreciated that the reward system is the most influential factor of success. Some reward systems give the top curves in the plot. Figure 5.18 contain the failure rates, which again are highly consistent for the reward system. Finally, the same conclusion can be drawn from the plot of the rate of inconclusive states in figure 5.19.

## 5.8 Specification of learning on the two robot system

The system consists of an unidimensional object (a hose) that has one end attached to a fixed point (which is set as the middle point of the ground working space), and two robots, one attached to the central point of the hose, and the other robot attached to the other end (the tip of the hose). In this way, the hose is transported using the two mobile robots. The learning task for the learning agents (both robots) is to bring the tip of the unidimensional object to an arbitrarily designated destination point. Most definitions for the Q-learning

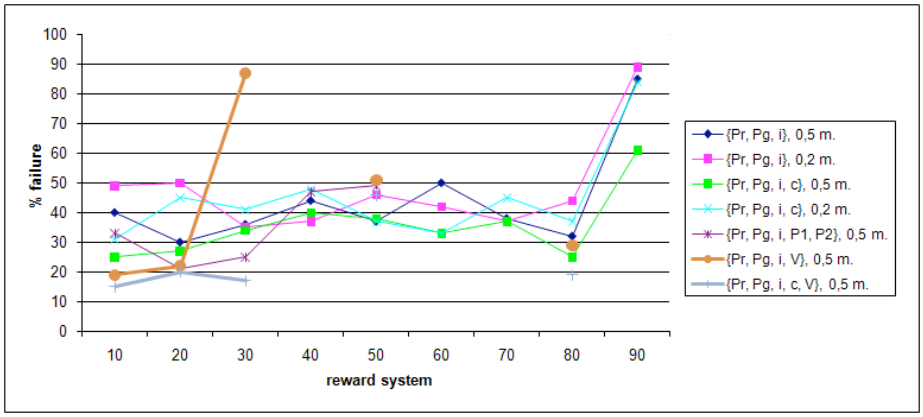



Figure 5.15: Summary failure rate for all state variable definitions and discretization step relative to the reward system

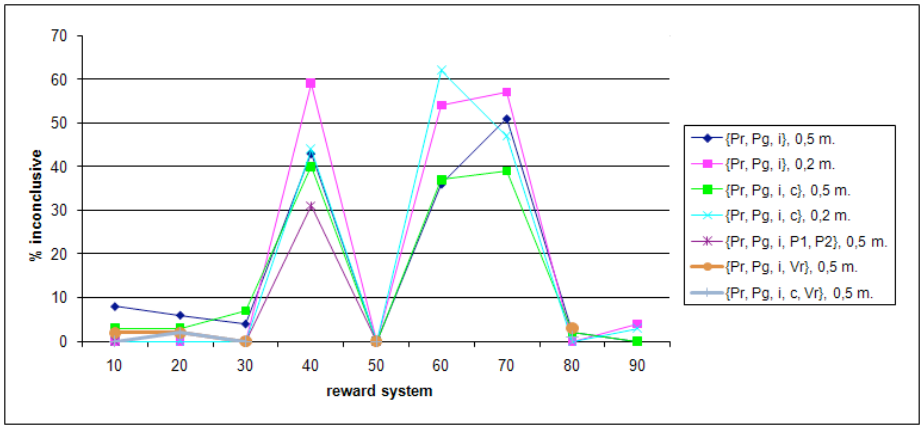



Figure 5.16: Summary inconclusive state rate for all state variable definitions and discretization step relative to the reward system

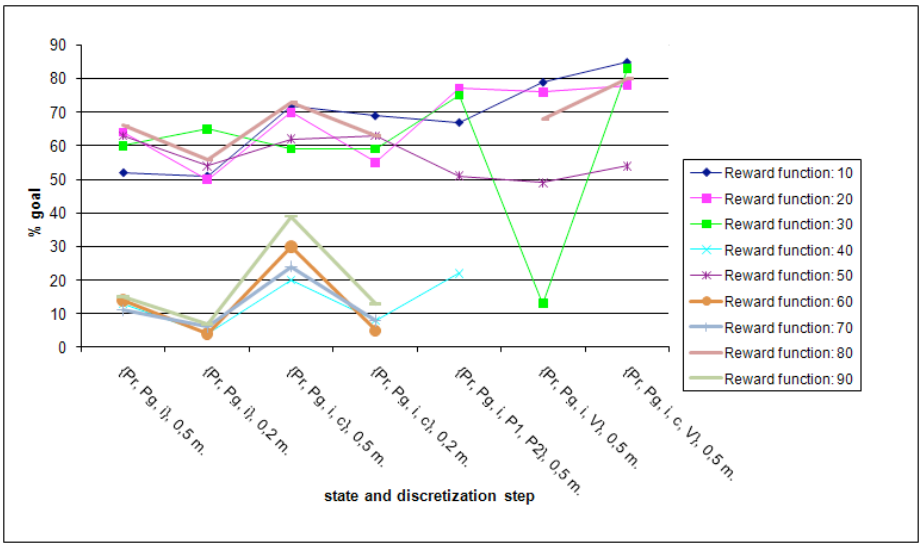



Figure 5.17: Summary success rate for all reward systems relative to the state variable definitions and discretization step.

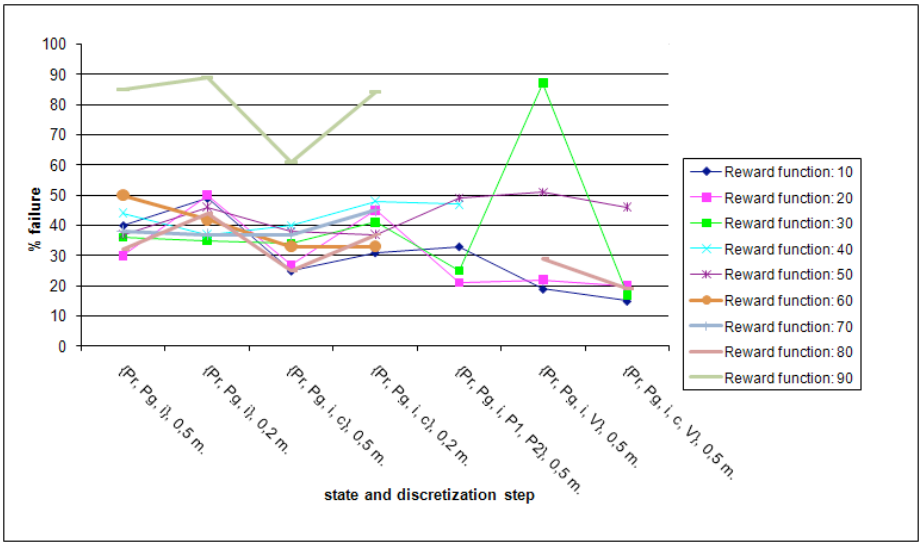



Figure 5.18: Summary failure rate for all reward systems relative to the state variable definitions and discretization step.

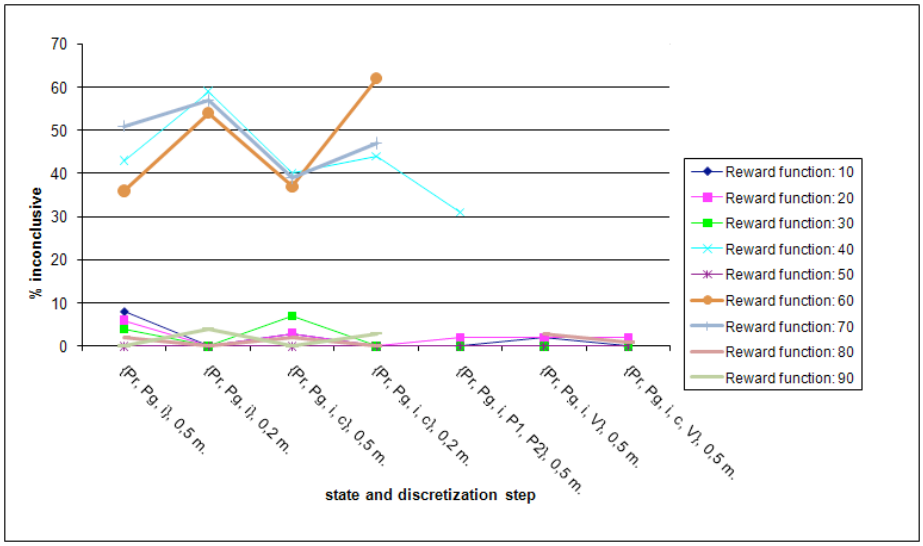



Figure 5.19: Summary inconclusive state rate for all reward systems relative to the state variable definitions and discretization step.

approach given in section 5.3 are the same in this setting, we will only specify the new elements of the definition.

- For the two robot system the state variables are $X = \{\mathbf{p}_{r_1}, i, c, \mathbf{V}_{r_1}, \mathbf{p}_{r_2}, \mathbf{V}_{r_2}, \mathbf{p}_g\}$. The meaning of these variables is as follows:
  - $\mathbf{p}_{r_1}$: it is the position $(x_{r_1}, y_{r_1})$ of the robot attached to the tip of the hose.
  - $i$: it is a boolean variable that is true if the line $\overline{\mathbf{p}_{r_1}\mathbf{p}_g}$ intersects the hose. This variable models a perceptual process that allows to decide if the hose is an obstacle to accomplish the task.
  - $c$: it is a boolean variable that is true if the box with diagonal corners $\mathbf{p}_{r_1}$ and $\mathbf{p}_g$ intersects the hose. This is a more sophisticated perceptual process, that implies a more detailed perception of the hose deployment than the one needed to compute $i$.
  - $\mathbf{V}_{r_1}$: it is a collection of boolean variables, one for each feasible action, that are true if there will be collision of the hose with itself after the robot $r_1$ performs the corresponding action. This variable involves some limited predictive ability of the robot $r_1$, which allows it to detect the danger one step ahead.

- $\mathbf{p}_{r_2}$: it is the position $(x_{r_2}, y_{r_2})$ of the robot attached to the tip of the hose.

- $\mathbf{V}_{r_2}$: it is a collection of boolean variables, one for each feasible action, that are true if there will be collision of the hose with itself after the robot $r_2$ performs the corresponding action. This variable involves some limited predictive ability of the robot $r_2$, which allows it to detect the danger one step ahead.

- $\mathbf{p}_g$: it is the goal position $(x_g, y_g)$ of the tip of the hose. It specifies the task to be accomplished. The distance between $\mathbf{p}_{r_1}$ and $\mathbf{p}_g$ is a measure of the degree of task accomplishment, but it is not explicitly used as a state variable.

- The available actions are the directions of motion of each robot carrying the hose plus the possibility of not making any movement, $A = \{North, South, East, West, None\}$. Each action of the overall system is taken as one action of one robot, because they take actions using a round robin algorithm. When a robot moves, it moves always one spatial unit corresponding to one discretization step thas has been taken in that experiment. The execution of an action involves the simulation of the hose dynamical model to evaluate the resulting hose spatial configuration.

- $R$. We have used several reward systems, taking into consideration only one robot and taking the two robots too. We have defined the immediate reward function according to the state that has been reached in each moment. So, rewards have three possible values as is explained in equation 5.3: $r_g$ when the system reaches the goal, $r_f$ when it fails to reach the goal, and $r_i$ when decision is postponed because the state is inconclusive. The condition "reaching the goal" is equivalent to "reaching the same box where the goal is located". As the motion of the robots that transport the hose is of fixed step-size, it is not possible to reach an specific goal point with arbitrary precision. In the table 5.1 we present all the reward systems that we have used taking into account only the robot that trasports the tip of the hose, while in the tables 5.4 and 5.5 we present the reward

systems that consider both robots.

$$R(s,\, a) \leftarrow \begin{cases} r_g & \arg\max_{s'} \{T(s,a,s')\} \in S^G \\ r_f & \arg\max_{s'} \{T(s,a,s')\} \in S^F \\ r_i & \arg\max_{s'} \{T(s,a,s')\} \in S^I \end{cases}, \tag{5.3}$$

- The maximum length of an episode is set to 30 steps when the discretization set is set to $0.5 \times 0.5\ m^2$ and to 100 steps when the discretization set is set to $0.2 \times 0.2\ m^2$.

## 5.9 Results for the two robot system

The reward systems already defined for the single robot case are applied here only to the robot carrying the tip of the hose. The other robot has no reward system, this equates to the idea that the lead robot is "responsible" for the task fulfilment. Reward systems described in the tables 5.4 and 5.5 are specific of the two robot system, giving specific rewards to the lead and middle robot. Figure 5.20 shows the plots of the success ratio (the % goals reached in the test episodes) for the different rewards changing the learning algorithm and the discretization step. The specific two robot reward system #210 gives the best results, though in some cases the single-robot reward system #30 is also very effective. The reward system #220 is rather catastrophic. . If we compare the Q-learning and TRQ-learning algorithms we TRQ-learning improves greatly on Q-learning for the lower discretization resolution. For the higher discretization resolution, Q-learning gives better results in some cases, but it is not consistently better. Overall the best performance is obtained with low resolution discretization, the TRQ-learning and the #210 reward system, reaching almost 90% successful episodes. Attending to the rate of failures in figure 5.21 we find that the reason for the poor performance of TRQ-learning on the high resolution discretization is that it reaches quite frequently an undesired blocking state. Q-learning suffers from a similar problem, but less pronunciated. Looking at the ratio of episode termination in inconclusive states inf figure 5.22 we find that TRQ-learnign falls below Q-learning, meaning that the TRQ-learning obtains policies that reach faster the terminal states, either positive or negative, while Q-learning seeem sto be procrastinating more. The relative improvement of Q-learning in some

reward system code: 200

$$r \leftarrow \begin{cases} +100 & if\ s \in S^G \\ -100 & if\ s \in S^F \\ f\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2},\, \mathbf{p}_g,\, i,\, c\right) & if\ s \in S^I \end{cases}$$

$$f\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2},\, \mathbf{p}_g,\, i,\, c\right) = -\left(\frac{d\left(\mathbf{p}_{r_1},\, \mathbf{p}_g\right)}{10} + 10i + 10c\right) + \frac{20}{g\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right)}$$

$$g\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right) = \max\left(\mathrm{abs}\left(r_1\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right)\right),\, abs\left(r_2\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right)\right)\right)$$

$$r_1\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right) = \begin{cases} \frac{d\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right)}{d\left(\mathbf{p}_{r_2},\, (\mathbf{0},\, \mathbf{0})\right)} & if\ d\left(\mathbf{p}_{r_2},\, (\mathbf{0},\, \mathbf{0})\right) \neq 0 \\ \infty & else \end{cases}$$

$$r_2\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right) = \begin{cases} \frac{d\left(\mathbf{p}_{r_2},\, (\mathbf{0},\, \mathbf{0})\right)}{d\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right)} & if\ d\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right) \neq 0 \\ \infty & else \end{cases}$$

reward system code: 210

$$r \leftarrow \begin{cases} +100 & if\ s \in S^G \\ -100 & if\ s \in S^F \\ f\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2},\, \mathbf{p}_g,\, i,\, c,\, \mathbf{V}_{r_1},\, \mathbf{V}_{r_2}\right) & if\ s \in S^I \end{cases}$$

$$f\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2},\, \mathbf{p}_g,\, i,\, c,\, \mathbf{V}_{r_1},\, \mathbf{V}_{r_2}\right) = -\left(\frac{d\left(\mathbf{p}_{r_1},\, \mathbf{p}_g\right)}{10} + 10i + 10c\right) + \frac{20}{g\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right)} + o\left(\mathbf{V}_{r_1},\, \mathbf{V}_{r_2}\right)$$

$$g\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right) = max\left(abs\left(r_1\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right)\right),\, abs\left(r_2\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right)\right)\right)$$

$$r_1\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right) = \begin{cases} \frac{d\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right)}{d\left(\mathbf{p}_{r_2},\, (\mathbf{0},\, \mathbf{0})\right)} & if\ d\left(\mathbf{p}_{r_2},\, (\mathbf{0},\, \mathbf{0})\right) \neq 0 \\ \infty & else \end{cases}$$

$$r_2\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right) = \begin{cases} \frac{d\left(\mathbf{p}_{r_2},\, (\mathbf{0},\, \mathbf{0})\right)}{d\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right)} & if\ d\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right) \neq 0 \\ \infty & else \end{cases}$$

$$o\left(\mathbf{V}_{r_1},\, \mathbf{V}_{r_2}\right) = 5\sum_i \mathbf{V}_{r_1,\, i} + 5\sum_i \mathbf{V}_{r_2,\, i}$$

Table 5.4: Reward systems taking into account two robots

reward system code: 220

$$r \leftarrow \begin{cases} +100 & if\ s \in S^G \\ -100 & if\ s \in S^F \\ f\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2},\, \mathbf{p}_g,\, i,\, c,\, \mathbf{V}_{r_1},\, \mathbf{V}_{r_2}\right) & if\ s \in S^I \end{cases}$$

$$f\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2},\, \mathbf{p}_g,\, i,\, c,\, \mathbf{V}_{r_1},\, \mathbf{V}_{r_2}\right) = -\left(\frac{d\left(\mathbf{p}_{r_1},\, \mathbf{p}_g\right)}{10} + 10i + 10c\right) + \frac{70}{g\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right)} + o\left(\mathbf{V}_{r_1},\, \mathbf{V}_{r_2}\right)$$

$$g\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right) = max\left(abs\left(r_1\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right)\right),\, abs\left(r_2\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right)\right)\right)$$

$$r_1\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right) = \begin{cases} \frac{d\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right)}{d\left(\mathbf{p}_{r_2},\, (\mathbf{0},\, \mathbf{0})\right)} & if\ d\left(\mathbf{p}_{r_2},\, (\mathbf{0},\, \mathbf{0})\right) \neq 0 \\ \infty & else \end{cases}$$

$$r_2\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right) = \begin{cases} \frac{d\left(\mathbf{p}_{r_2},\, (\mathbf{0},\, \mathbf{0})\right)}{d\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right)} & if\ d\left(\mathbf{p}_{r_1},\, \mathbf{p}_{r_2}\right) \neq 0 \\ \infty & else \end{cases}$$

$$o\left(\mathbf{V}_{r_1},\, \mathbf{V}_{r_2}\right) = 5\sum_i \mathbf{V}_{r_1,\, i} + 5\sum_i \mathbf{V}_{r_2,\, i}$$

Table 5.5: Reward systems taking into account two robots

instances is due to the higher rate of inconclusive terminations, biasing the success statistics towards Q-learning.

## 5.10 Conclusions

We have approached the hose transportation problem in a L-MCRS using Reinforcement Learning methods, more specifically, Q-learning and TRQ-learning defined in Chapter 4. We have worked on a single robot and a two robot configurations, performing extensive learning experiments using the accurate simulation of the model described in Chapter 2. Simulations are quite expensive in computation time, the effective time required for the training process is in order of hours or days. However, this simulation process allows reproducibility and increased accuracy of validation. We have tested several combinations of configuration space resolution, reward systems and definitions of the state variables modeling the process. Each state variable definition embodies some assumptions on the system and the knowledge available to the learning agent. We have found that some simple reward systems provide the best results. Also,

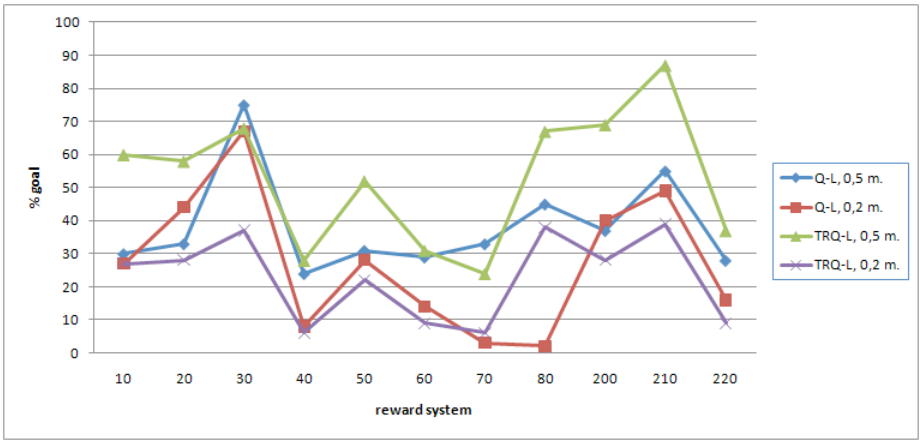



Figure 5.20: Success of the training methods on the two robot system relative to the reward system

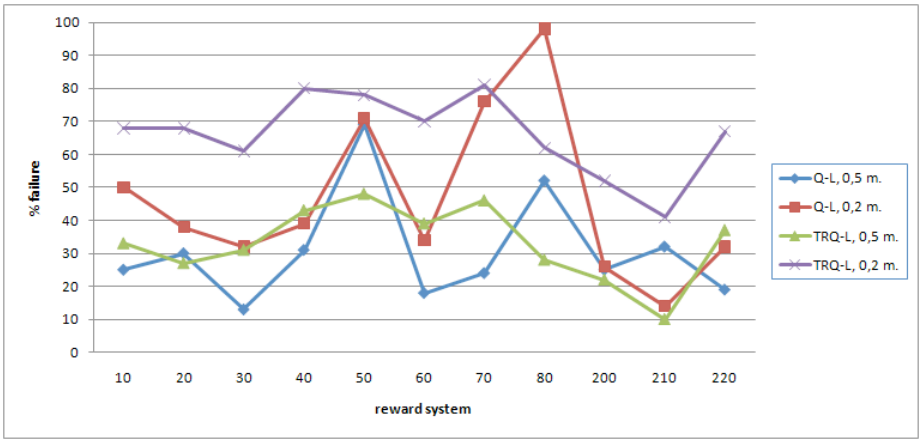



Figure 5.21: Failure of the training methods on the two robot system relative to the reward system

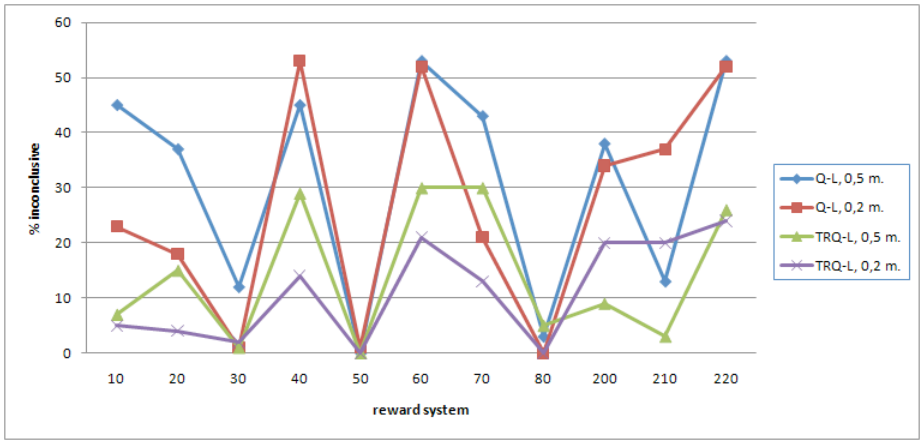



Figure 5.22: Rate of inconclusive states reached by the training methods on the two robot system relative to the reward system

the ability to perform first order predictions enhances the system performance. In the two robot system, we have found that applying the reward system only to the lead robot can provide good resuls in some cases.

# Bibliography


[1] S. S. Antman. *Nonlinear Problems of Elasticity*. Springer-Verlag, 1995.

[2] Richard Bellman. A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684, 1957.

[3] Carl De Boor. *A Practical Guide to Splines*. Springer, August 1994.

[4] Y.U. Cao, A.S. Fukunaga, and A. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–27, March 1997.

[5] C. Chen, P. Yang, X. Zhou, and D. Dong. A quantum-inspired q-learning algorithm for indoor robot navigation. In *Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference on*, pages 1599 –1603, 6-8 2008.

[6] Y. Duan and X. Hexu. Fuzzy reinforcement learning and its application in robot navigation. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 2, pages 899 –904 Vol. 2, 18-21 2005.

[7] G. Dudek, M. R. M. Jenkin, E. Milios, and D. Wilkes. A taxonomy for multi-agent robotics. *Autonomous Robots*, 3(4):375–397, December 1996.

[8] R. Duro, M. Graña, and J. de Lope. On the potential contributions of hybrid intelligent approaches to Multicomponent Robotic System development. *Information Sciences*, 180(14):2635–2648, 2010.

[9] R.J. Duro, M. Graña, and J. de Lope. On the potential contributions of hybrid intelligent approaches to multicomponen robotic system development. *Information Sciences*, 2010. in press.

[10] A. Echegoyen, I. Villaverde, R. Moreno, M. Graña, and A. d'Anjou. Linked multi-component mobile robots: modeling, simulation and control. *Robotics and Autonomous Systems*, in press, 2010.

[11] A. Farinelli, L. Iocchi, and D. Nardi. Multirobot systems: a classification focused on coordination. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(5):2015–2028, October 2004.

[12] Jacques Ferber. *Multi-agent systems: an introduction to distributed artificial intelligence*. Addison-Wesley, 1999.

[13] B. Fernandez-Gauna, J.M. Lopez-Guede, and E. Zulueta. Linked multicomponent robotic systems: Basic assessment of linking element dynamical effect. In Maite García-Sebastián Manuel Grana Romay, Emilio S. Corchado, editor, *Hybrid Artificial Intelligence Systems, Part I*, volume 6076, pages 73–79. Springer Verlag, 2010.

[14] B. Fernandez-Gauna, J.M. Lopez-Guede, E. Zulueta, and M. Graña. Learning hose transport control with q-learning. *Neural Network World*, 20(7):913–923, 2010.

[15] Zelmar Echegoyen Ferreira. *Contributions to Visual Servoing for Legged and Linked Multicomponent Robots*. PhD thesis, UPV/EHU, 2009.

[16] SC Gadanho and J Hallam. Emotion-triggered learning in autonomous robot control. *Cybernetics and Systems*, 32(5):531–559, Jul-Aug 2001. 5th International Conference on Simulation of Adaptive Behavior, Zurich, Switzerland, Aug 17-21, 1998.

[17] Mireille GrÃ©goire and Elmar SchÃ¶mer. Interactive simulation of one-dimensional flexible parts. *Computer-Aided Design*, 39(8):694–707, 2007.

[18] E. HergenrÃ¶ther and P. Dhne. Real-time virtual cables based on kinematic simulation. In *Proocedings of the WSCG*, 2000.

[19] L. Huang. Speed control of differentially driven wheeled mobile robots: Model-based adaptive approach. *Journal of Robotic Systems*, 22(6):323–332, 2005.

[20] Rahul Kala, Anupam Shukla, and Ritu Tiwari. Dynamic environment robot path planning using hierarchical evolutionary algorithms. *Cybernetics and Systems*, 41(6):435–454, 2010.

[21] K Kiguchi, K Watanabe, K Izumi, and T Fukuda. A humanlike grasping force planner for object manipulation by robot manipulators. *Cybernetics and Systems*, 34(8):645–662, Dec 2003.

[22] Gregor Klancar and Igor Skrjanc. Tracking-error model-based predictive control for mobile robots in real time. *Robotics and Autonomous Systems*, 55:460–469, 2007.

[23] K. C. Koh and H. S. Cho. A smooth path tracking algorithm for wheeled mobile robots with dynamic constraints. *Journal of Intelligent and Robotic Systems*, 24:367–385, 1999.

[24] Ning Liu. Intelligent path following method for nonholonomic robot using fuzzy control. In *Second International Conference on Intelligent Networks and Intelligent Systems*, 2009.

[25] J.M. Lopez Guede, M. Graña, E. Zulueta, and O. Barambones. Economical implementation of control loops for multi-robot systems. In *Advances in Neuro-Information Processing*, volume 5506/2009 of *Lecture Notes in Computer Science*, pages 1053–1059. Springer, 2009.

[26] Ali Reza Mehrabian, Caro Lucas, and Jafar Roshanian. Design of an aerospace launch vehicle autopilot based on optimized emotional learning algorithm. *Cybernetics and Systems*, 39(3):284–303, Apr 2008.

[27] F.S. Melo and M.I. Ribeiro. Reinforcement learning with function approximation for cooperative navigation tasks. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 3321 –3327, 19-23 2008.

[28] S. Miyata, H. Nakamura, A. Yanou, and S. Takehara. Automatic path search for roving robot using reinforcement learning. In *Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference on*, pages 169 –172, 7-9 2009.

[29] Iván Villaverde De La Nava. *On Computational Intelligence Tools for Vision Based Navigation of Mobile Robots*. PhD thesis, UPV/EHU, 2009.

[30] AMAF Nuseirat and R Abu-Zitar. Hybrid trajectory planning using reinforcement and backpropagation through time techniques. *Cybernetics and Systems*, 34(8):747–765, Dec 2003.

[31] G. Oriolo, A. De Luca, and M. Vendittelli. Wmr control via dynamic feedback linearization: Design, implementation, and experimental validation. *IEEE Transactions on Control Systems Technology*, 10(6):835–852, November 2002.

[32] Dinesh K. Pai. Strands: Interactive simulation of thin solids using cosserat models. *Computer Graphics Forum*, 21(3):347–352, 2002.

[33] Hong Qin and Demetri Terzopoulos. D-nurbs: A physics-based framework for geomatric design. Technical report, Los Alamitos, CA. USA, 1996.

[34] Francesco M. Raimondi and Maurizio Melluso. A new fuzzy robust dynamic controller for autonomous vehicles with nonholonomic constraints. *Robotics and Autonomous Systems*, 52:115–131, 2005.

[35] Wei Ren and Randal W. Beard. *Distributed Consensus in Multi-Vehicle Cooperative Control: Theory and Applications*. Springer Publishing Company, Incorporated, 2007.

[36] M.B. Rubin. *Cosserat Theories: Shells, Rods and Points*. Kluwer, 2000.

[37] L. Shi-Cai, T. Da-Long, and L. Guang-Jun. Formation control of mobile robots with active obstacle avoidance. *Acta Automatica Sinica*, 33(5):529–535, 2007.

[38] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[39] Robby T. Tan, Ko Nishino, and Katsushi Ikeuchi. Separating reflection components based on chromaticity and noise analysis. *IEEE Trans Pattern Anal Mach Intell*, 26(10):1373–1379, October 2004.

[40] A. Theetten, L. Grisoni, C. Andriot, and B. Barsky. Geometrically exact dynamic splines. *Computer-Aided Design*, 40(1):35–48, January 2008.

[41] Henk C. Tijms. *Discrete-Time Markov Decision Processes*, pages 233–277. John Wiley & Sons, Ltd, 2004.

[42] Randal W. Beard Timothy W. McLain. Coordination variables, coordination functions, and cooperative timing missions. *AIAA Journal of Guidance, Control, & Dynamics*, 28(1):150–161, 2005.

[43] I. Villaverde, Z. Echegoyen, R. Moreno, and M. Graña. Experiments on robotic multi-agent system for hose deployment and transportation. In Y. Demazeau et al., editor, *Trends in Practical Applications of Agents and Multiagent Systems*, volume 71 of *Advances in Intelligent and Soft Computing*, pages 573–580. Springer, 2010.

[44] C. Watkins. *Learning from Delayed Rewards*. PhD thesis, University of Cambridge,England, 1989.

[45] CJCH Watkins and P Dayan. Q-Learning. *Machine Learning*, 8(3-4):279–292, MAY 1992.