Jorge da Silva Perdigão

Collaborative-Control based Navigation of Mobile Human-Centered Robots

Dissertation submitted in partial fulfillment of the
requirements for the degree of Master of Science
in Electrical and Computer Engineering
July, 2014

· U C ·

Universidade de Coimbra

University of Coimbra
Faculty of sciences and technology
Department of Electrical and Computer Engineering

Collaborative-Control based Navigation of Mobile Human-Centered Robots

Jorge da Silva Perdigão

Coimbra, 2014

Collaborative-Control based Navigation of Mobile Human-Centered Robots

Advisor: Prof. Dr. Urbano José Carreira Nunes
Advisor: Dr. Ana Cristina Barata Pires Lopes

Jury:
Prof. Dr. Manuel Marques Crisóstomo
Prof. Dr. Rui Pedro Duarte Cortesão
Prof. Dr. Gabriel Pereira Pires
Prof. Dr Urbano José Carreira Nunes

Jorge da Silva Perdigão

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical and Computer Engineering

Department of Electrical and Computer Engineering

Faculty of Sciences and Technology, University of Coimbra

July, 2014

"Not all those who wander are lost."

-J. R. R. Tolkien.

Acknowledgements

First of all I would like to express gratitude to my advisors Professor Dr. Urbano Nunes and Dr. Ana Lopes for their guidance, insightful observations, patience and always the best of advice. This dissertation would be impossible without their dedication.

Writing a dissertation is certainly not an individual task, and I must thank all members of the Autonomous Vehicles and Human-Centered Robotics team for their support, friendship and assistance. I owe a special thanks to André Lopes, Cristiano Premebida, Daniel Almeida, Diogo Gonçalves, Fábio Faria, João Sousa and Luis Garrote.

I would also like to thank ISR for hosting me, providing the best of resources, working environment and personnel that allowed me to accomplish all the goals I worked for. This work has been supported by Fundação para a Ciência e Tecnologia (FCT), under the project of "Assisted Mobility Supported by shared control and advanced Human Machine Interfaces" (AMS-HMI12 - RECI/EEI-AUT/0181/2012).

I thank my family for always being there for me. I must express special gratitude to my parents for being so comprehensive and supportive through all my life, specially during my university years.

To my girlfriend, Sara, I leave my greatest gratitude. Her patience and ever-present cheering attitude made the way brighter even in my darkest days.

Resumo

O envelhecimento da população nos países ocidentais tem pressionado a indústria de cuidados e de assistência a idosos a apresentar soluções eficientes para responder à crescente procura aos seus serviços. Isto leva a crer que a robótica será uma das soluções mais procuradas, ao aliviar a necessidade de apoio humano constante. Responder a esta procura é uma tarefa que deve começar a ser tida em conta agora. Esta dissertação é um trabalho na área da Robótica de assistência. Tem como principais objetivos a análise e a implementação prática de um sistema de navegação de assistência para a RobChair, uma cadeira de rodas inteligente desenvolvida no ISR-UC. É um estudo nas áreas de Navegação Autónoma, SLAM (*Simultaneous Localization and Mapping*) e de Controlo Colaborativo. Define como objetivo criar um sistema inteligente capaz de melhorar a mobilidade de doentes com graves disfunções motoras, melhorando a sua qualidade de vida. Começa por analisar o conceito de controlo colaborativo e o estado da arte em interação Humano-Robô. Analisa também soluções de SLAM sem odometria, mais concretamente o Hector SLAM, e de navegação autónoma. Para atingir os objetivos propostos, este trabalho apresenta uma arquitetura de um Sistema de Navegação Assistiva (ANS), e particularmente um desenho para um Controlador Colaborativo. Esse sistema é implementado em ROS (*Robot Operating System*) e os seus módulos são explanados. São também apresentadas alterações no hardware da RobChair tendo em vista facilitar o seu uso enquanto robô experimental. O sistema foi testado experimentalmente, validando o funcionamento de alguns dos seus componentes. Estes resultados representam os primeiros passos na implementação do ANS completo na parte de navegação e perceção.

Palavras-chave: Robótica Centrada no Humano; Controlo Colaborativo; SLAM; Navegação Assistida; Cadeira de Rodas Robótica, RobChair, ISR-UC, ROS.

Abstract

Population aging in western countries has been pressing the industry of elderly care and assistance to present more efficient solutions in order to meet the growing demand for its services. This suggests that robotics shall be one of the most sought solutions, relieving the constant necessary human support. Responding to this demand is a task that must be taken into consideration as soon as possible. This dissertation is a work on the field of Assistive Robotics, whose main goals are the analysis and implementation of an assistive navigation system for RobChair, an intelligent wheelchair developed at ISR-UC. It is a study on the fields of Autonomous Navigation, SLAM (Simultaneous Localization and Mapping) and Collaborative Control. It defines the objective of creating an intelligent system able to improve the mobility of patients with severe motor dysfunctions, improving their quality of living. It starts by analyzing the concept of Collaborative Control and the state of the art in Human-Robot interaction. It also analyzes the SLAM solutions without odometry, more specifically, the Hector SLAM, and autonomous navigation. To attain the proposed objectives, this work presents an architecture for the Assistive Navigation System (ANS) and particularly a scheme for the Collaborative Controller. Such system is implemented in ROS and its modules explained in detail. It is also presented the modifications performed in the RobChair hardware in order to facilitate its use as an experimental robot. The system was tested experimentally, validating the operation of some of its components. The results achieved represent the first steps in implementing the complete ANS, considering the navigation and perception modules.

Key words: Human-Centered Robotics; Collaborative Control; SLAM; Assisted Navigation; Robotic Wheelchair; RobChair; ISR-UC; ROS.

Contents

List of Figures

List of Tables

Nomenclature

3D	Three Dimensional
ALS	Amyotrophic Lateral Sclerosis
ANS	Assistive Navigation System
BCI	Brain-Computer Interface
CC	Collaborative Controller
CP	Cerebral Palsy
DOF	Degrees Of Freedom
DWA	Dynamic Window Approach
EEG	Electroencephalography
ERP	Event-Related Potentials
FCT	Fundação para a Ciência e Tecnologia
Hector	Heterogeneous Cooperating Teams of Robots
HMI	Human-Machine Interaction
HRI	Human-Robot Interaction
IM	Intent Matcher
ISR	Institute of Systems and Robotics
LIDAR	Light Detection and Ranging
LRF	Laser Rage Finder
RC	Remote Controller
RGB	Red, Green, Blue
RGB-D	Red, Green, Blue and Depth
RobChair	Robotic Wheelchair (referring to the ISR Robotic Wheelchair)
ROS	Robot Operating System

RRT	Rapidly-exploring Randomizing Tree
RW	Robotic Wheelchair
SLAM	Simultaneous Localization and Mapping
TCP/IP	Transmission Control Protocol/Internet Protocol
UARS	User-Aid-Required Situations
UC	Universidade de Coimbra
UID	User Interface Design
URDF	Unified Robot Description Format
USAR	Urban Search and Rescue
VC	Virtual Constraint

Chapter 1

Introduction

This chapter unveils the views and motivations that led to this work, as well as its main goals and key contributions. It summarizes the course intended for this dissertation and the contents of each chapter.

1.1 Motivation and context

For the past few years, several studies in Europe and other western countries have showed a trend in demographics of population aging and depopulation. This increasing in elderly population will inevitably pressure the senior care industry to innovate and find new and robust solutions to provide full-time care to these individuals [de Jouvenel, 1989, Bloom et al., 2011]. Such trends seem to lead to a world where robots and intelligent systems will have an evermore important role in integrating and assisting impaired individuals. On another perspective, there are several non age-related disorders where robots may assist. Severe cases of mobility disorder, such as Cerebral Palsy (CP) and Amyotrophic Lateral Sclerosis (ALS) [Hoffmann et al., 2008, Nijboer et al., 2008] are a notorious example.

This work focuses on the development of human-centered robotics, and specifically in the semi-autonomous control of robotic wheelchairs for the motor impaired, whose life standards are normally reduced due to their motor conditions, and so are ones where assistive technology can help first. The deployment of human-centered robots, such as robotic wheelchairs, may contribute to help motor-impaired people to reach a better level of mobility, thus improving their life standards. Furthermore, increasing the mobility levels of people with motor disabilities can also ultimately contribute to improve their social inclusion.

Robotic wheelchairs have in fact been one of the first taken routes towards assistive robotics by researchers worldwide. In the Institute for Systems and Robotics (ISR), the RobChair (Robotic WheelChair) project is being developed [Pires and Nunes, 2002, Lopes et al., 2011, Lopes et al., 2012, Lopes et al., 2013] since the mid-90's, with particular focus on BCI (Brain Computer Interfaces) and collaborative control. Since then, difficulties related with pose estimation, efficient navigation and other mobile robotics issues have been successfully improved [Grisetti et al., 2007, Marder-Eppstein et al., 2010]. Some of this progress is now more easily accessed with software sharing communities like ROS, and with increased published work on the subject. On the other hand, BCI technologies have been evolving consistently and now

produce results that increasingly encourage its use on assistive technologies.

Growing success in both technologies encourages further efforts on implementing assistive systems like the one proposed in this work. With the achievements reached so far, with better solutions and algorithms, the development of systems that put them together contributes to more robust and fully functioning systems.

1.2 Goals

In 2001, when RobChair Project was still at its beginning, four key goals were established in [Pires, 2001]. The defined goals were:

1. To ensure the safety and integrity of RobChair users;

2. To minimize user's effort in driving the wheelchair, by adding growing levels of functionality:

   (a) Shared control between user and wheelchair;

   (b) Local planning of local trajectories for obstacle avoidance;

   (c) Autonomous execution of difficult tasks such as door crossing and wall following;

3. To allow users incapable of using standard HMIs, such as the usual joystick, to command the wheelchair.

4. To integrate the wheelchair within a network thus providing communication with the outside.

In 2001, these were goals difficult to attain. Since then, substantial achievements have been made in the field of mobile robotics and great advances in computing technologies helped scientists and engineers tackle these problems. Even so, pursuing some of the goals proposed in 2001 remain a challenge today, not only because they are complex in nature, but also because the tasks needed to achieve such goals depend greatly on the environment the wheelchair is on, as well as the level of disability of the user controlling it.

This work aims to provide a new take on Assistive Navigation System (ANS), integrating it with the new software and hardware architecture developed for RobChair in [Gonçalves, 2013], upgrading the hardware, software and developing new algorithms.

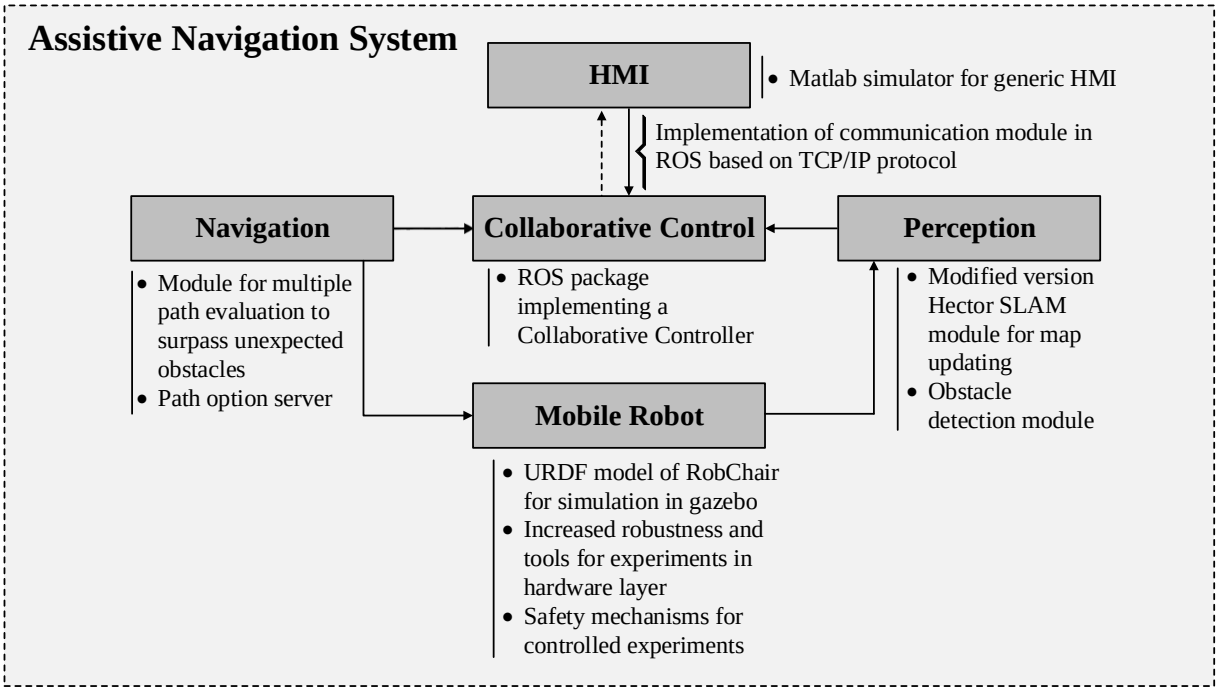


Figure 1.1: ANS main modules and key contributions

1.3 Implementations and key contributions

The key contribution of this dissertation are the first steps towards the development and implementation of an Assistive Navigation System on ROS framework, for RobChair. To implement the ANS, several works shared by other authors were studied and analyzed in this system. In order to build a functioning system, some modules were fully implemented, modified from a distributed version or added after careful analysis. Figure 1.1 summarizes the implementations and contributions described in this dissertation.

Assistive Navigation System (ANS) (Chapter 3):

- Overview of the current ANS architecture, where contributions were given in this dissertation. All modules are briefly described and interaction between them is analyzed.

Collaborative Controller (Chapter 4 and 5):

- Collaborative Controller: A collaborative control architecture [Lopes, 2012] was redesigned and implemented in ROS environment.

Perception (Chapter 3 and 5):

- Hector mapping for RobChair: A ROS SLAM method (*Hector* SLAM) was adapted for the proposed ANS and specifically for RobChair, allowing for an *a priori* map and its subsequent updates.
- Obstacle Detection: A method for detection of unexpected obstacles was implemented.

Human-Machine-Interface (Chapter 3 and 5):

- A simulator for a discrete HMI, with a predefined set of commands, was implemented in Matlab. It defines a protocol used throughout the ANS.
- A communication protocol was also established for data exchange between HMIs and the ANS.

Navigation (Chapter 3 and 5):

- Path option server: A module to distribute information about path options available to the user was implemented. In this dissertation, three situations were considered: obstacle detection, path forking and deadlocks.

Mobile Robot (Chapter 5):

- RobChair Simulation: A URDF model of RobChair was created for simulation in gazebo simulator.
- Hardware Update: Safety mechanisms and physical supports were added to RobChair in order to provide robustness and enable controlled experiments.

In Chapter 6, several experiments for testing and validating the system are described and its results discussed.

Chapter 2

Background and literature

This chapter introduces the fundamental topics required to understand the work presented in this dissertation. In the first section a brief discussion on robotic wheelchairs is carried out. The subsequent sections review important background theory and state of the art on the specific topics of robotics that are involved in the design and development of this work; by order of appearance, the subtopics are: Localization and Mapping, Path Planning, Semi-autonomous Control and Human-Machine interfaces.

2.1 Robotic Wheelchairs

A Robotic Wheelchair (RW) is a type of mobile robot. Mobile robotics is a broad engineering field, and an important branch of robotics. Mobile robots are defined as robots capable of locomotion, meaning they can move more or less freely in their environment; this includes various types of locomotion, such as wheeled, legged, underwater and airborne motion [Siegwart et al., 2011]. This field studies the kinematic models, motion control, perception (sensory information processing), planning and navigation for such robots.

Robotic wheelchairs are a particular form of wheeled robots. They are an upgrade from standard motorized wheelchairs with the general goal of simplifying and assist the users in navigation tasks. Research concerning RWs has grown consistently over the last few years and varied intelligent wheelchair platforms have been developed. One of these platforms is RobChair, the robotic wheelchair developed at the Institute of Systems and Robotics at University of Coimbra [Pires and Nunes, 2002, Lopes et al., 2011, Lopes et al., 2012, Lopes et al., 2013]. Most of the RWs aim to provide some type of semi-autonomous control, aiding the patients by easing the hard task of maneuvering the wheelchair. To achieve such goal, an HMI is required to collect the user intent. On the other hand, all the RWs require some kind of environment analysis, decision making modules for navigation purposes and a control scheme that connects user intent and navigation reference. A list of some recent Robotic Wheelchairs are presented in Table 2.1, summarizing the main technologies implemented.

Table 2.1: Recent wheelchair platforms [Gonçalves, 2013]

Institution	Main Technologies	Control scheme	HMI
University of technology of Sydney [Patel et al., 2012]	Montecarlo localization Topological mapping.	Hierarchical Hidden Markov Model framework that predicts both the short term (local) and long term (navigational) goals of the user.	Joystick.
VAHM (LASC, University Paul Verlaine-Metz) [Grasse et al., 2010]	Particle filtering approach to implement the recognition of the most frequent paths according to an offline-constructed topological map.	Provides assistance to the user during navigation by proposing the direction to be taken when a path has been recognized.	Joystick.
SHARIOTO (Katholieke Universiteit Leuven) [Vanhooydonck et al., 2010]	Dynamic window approach for obstacle avoidance.	Shared-control with user intention prediction based on a Bayesian network.	Joystick.
INRIA [Rios-Martinez et al., 2011, Escobedo et al., 2012]	Motion planner based on the RiskRRT; Map of the environment is built using a LIDAR mounted on the top of the wheelchair; Important goals in the map are set by hand.	A Bayesian network is used to estimate the user intent. Generation of human friendly paths based on the application of a social filter, which includes constraints inspired by social conventions.	Face tracking and voice recognition.
LURCH Politecnico di Milano [Bonarini et al., 2012]	Localization based on odometry. Odometry correction is performed based in the detection of passive markers placed in the environment using vision. Trajectory planning based on the fast planner SPIKE (Spike plans In Known Environments).	Control module based on a fuzzy behavior management system, where a set of reactive behaviors, which will be carried out by the robot, are implemented as a set of fuzzy rules. Two set of rules were established: one implementing trajectory following, and the another one implementing obstacle avoidance.	Joystick, touch-screen, electro miographic interface, and BCI.
University of Michigan [Park et al., 2012]	A static occupancy grid map obtained via SLAM. Global topological map. Position and velocity estimation of new obstacles in the environment based on a Kalman filter.	Model Predictive Equilibrium Point Control (MPEPC) framework, which allows the navigation of a wheelchair in dynamic, uncertain, structured scenarios with multiple pedestrians.	Joystick.
RobChair Institute for Systems and Robotics, University of Coimbra [Lopes et al., 2011, Lopes et al., 2012, ?]	A priori metric map; Markov localization based on laser scan matching; Global planner based on the A*algorithm and local planning for obstacle avoidance.	Two-layer collaborative controller that depends on the user's ability steering the wheelchair with the BCI (the user selects among a set of discrete steering commands); Intent matching algorithm that matches user intents with machine steering proposals.	Synchronous BCI; scanning interface with sin-gle/multiple switch.
University of Zaragoza [Iturrate et al., 2009]	A binary occupancy grid map is used to model the static obstacles and free space. A set of extended Kalman filters was chosen to track moving objects around the robot. The final motion of the vehicle was computed using the nearness diagram (ND) technique.	Control of real wheelchair and simulated wheelchair in virtual environment (selection of local surrounding points).	Synchronous BCI.

At ISR, the RobChair has been a research platform for integrating Brain-Computer Interface (BCI) and other HMIs, and to develop navigation and obstacle detection algorithms, with the goal of improving mobility and life quality of impaired people [Lopes et al., 2012, Lopes et al., 2011, Pires and Nunes, 2002, Lopes et al., 2013].

2.2 Localization and Mapping

In Mobile Robotics, localization is the problem of estimating the robot's pose relatively to its environment. This is crucial to most basic tasks in mobile robotics. Localization is a broad term comprising a group of variations from the core pose estimation problem. As [Thrun et al., 2005] puts it, localization can be global (initial pose is unknown, the robot can be anywhere) or local (initial pose is known simplifying the problem to position tracking); it can be set in a dynamic or in a static environment, it can be passive or active and it can be a problem for a single robot or a team of cooperating robots. Important solutions to the localization problem in mobile robotics are the Monte Carlo and Markov-based solutions [Thrun et al., 2005].

The localization problem by itself usually assumes that the map of the environment is known. However, in many cases, because the environment is unknown or because updating the map is required, the localization task must run together with the mapping task. This problem, where the geometric relation between the robot and a set of landmarks (that compose a map of the environment) must be estimated is known as Simultaneous Localization and Mapping (SLAM). As [Durrant-Whyte and Bailey, 2006] recalls, while describing the history of this dual problem, after several years tackling the probabilistic mapping problem with solutions exhibiting unbounded error growth, it finally became clear that if the combined mapping and localization problem was formulated as a single estimation problem, it would became convergent [Durrant-Whyte et al., 1996]. Realizing this was a milestone for the problem and since then it has been called SLAM, reinforcing the idea of a single approach to achieve the two tasks.

2.2.1 Definition of the SLAM problem

The SLAM problem is approached in many different and diverse ways. Nevertheless, a general definition of the problem can be established, if some simplifications are allowed.

Assume a robot driving through an unknown, static environment, with uncertain motion. The robot is equipped with sensors that enable estimation of geometric relationships between the robot and world landmarks. The SLAM problem consists in estimating the position of each landmark relatively to the initial robot position and between them, i.e., in estimating a map of the world landmarks. Furthermore, this task also requires an estimate of the robot position in the map, since both map and robot pose estimates are interconnected [Siciliano and Khatib, 2008]. Since SLAM is an estimation problem, a success mathematical approach to it is done in probabilistic terms.

Allow x_t to be the robot pose in time frame t; u_t the command given to the robot that drives it from x_{t-1} to x_t and z_t the sensor measurements at time t . Let $x_{0:t}$ be a vector comprising all robot poses since the initial time frame; and $u_{1:t}$ and $z_{1:t}$ the sets of controls given to the robot and the sensor measurements from the robot in all subsequent time frames, respectively. Figure 2.1 represents the causal relationship between these variables. Lightly shaded circles represent variables not directly observable, and the ones SLAM tries to estimate from the directly observable information, represented by the darker circles. We proceed by distinguish two possible takes on the SLAM problem:

- One approach seeks to estimate the current robot pose in map m, i.e., calculate the posterior $p\left(x_t, m \mid z_{1:t}, u_{1:t}\right)$. This approach is called *online* SLAM;

- Another approach seeks to estimate the whole vector $x_{1:t}$ relatively to the world map, by calculating the posterior $p\left(x_{1:t}, m \mid z_{1:t}, u_{1:t}\right)$. This approach is called *full* SLAM;

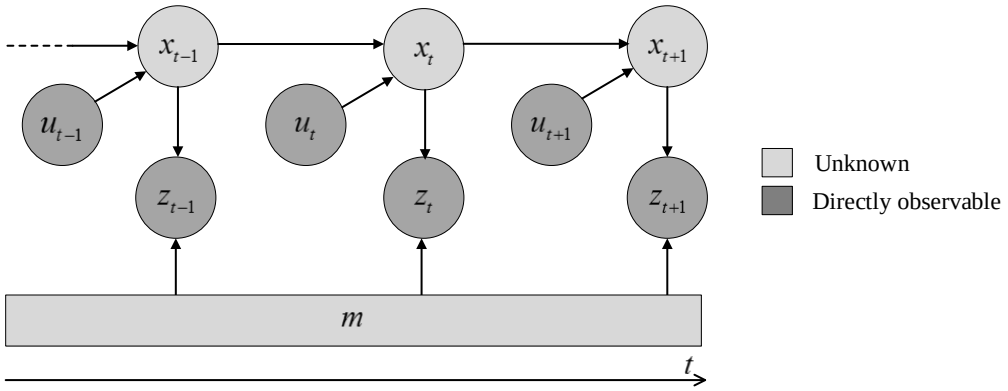


Figure 2.1: Graphical model of the SLAM problem (adapted from [Siciliano and Khatib, 2008])

For most applications, only the online problem will be feasible, and practical approaches that tackle full SLAM must rely on important approximations.

To solve either definition of the problem, two probabilistic observation models must be considered, one that links sensor measurements z_t and the world m and other that relates control measurements u_t and robot pose x_t. These models may be considered as probability distributions: $p\left(z_t|x_t, m\right)$ for the first model, and $p\left(x_t|x_{t-1}, u_t\right)$ for the second.

Further detailed solutions to the SLAM problem depend greatly on assumptions of the environment and models. SLAM can be volumetric (by sampling the map) or feature-based (by extracting sparse information about the world); topological (defining a pose graph) or metric; with known or unknown correspondence of the map landmarks in different time frames; with a static or dynamic assumption of the world; considering small or large uncertainty; active or passive (considering exploration) or done by a single robot or by a team of robots.

2.2.2 Darmstadt's team Hector approach to SLAM

The team Hector (Heterogeneous Cooperating Team of Robots) is a research group based in Technische Universität Darmstadt. Among other topics, they have been researching mobile robotics for Urban Search and Rescue (USAR) [Darmstadt, 2012]. For several applications in USAR scenarios, these robots require localization and map of the environment. Such scenarios may have harsh (i.e., irregular and not planar) terrain, making odometry unreliable and full 6-DOF pose estimation a requirement. A 2D SLAM solution able to function with 6-DOF pose estimation and elevation mapping was developed, fulfilling the design requirements. An overview of the solution is depicted in Figure 2.2, displaying the data flow between sensors, 2D SLAM (also known as Hector Mapping) and 3D pose estimation subsystems. More details of the complete system can be found in [Kohlbrecher et al., 2011, Kohlbrecher et al., 2012].

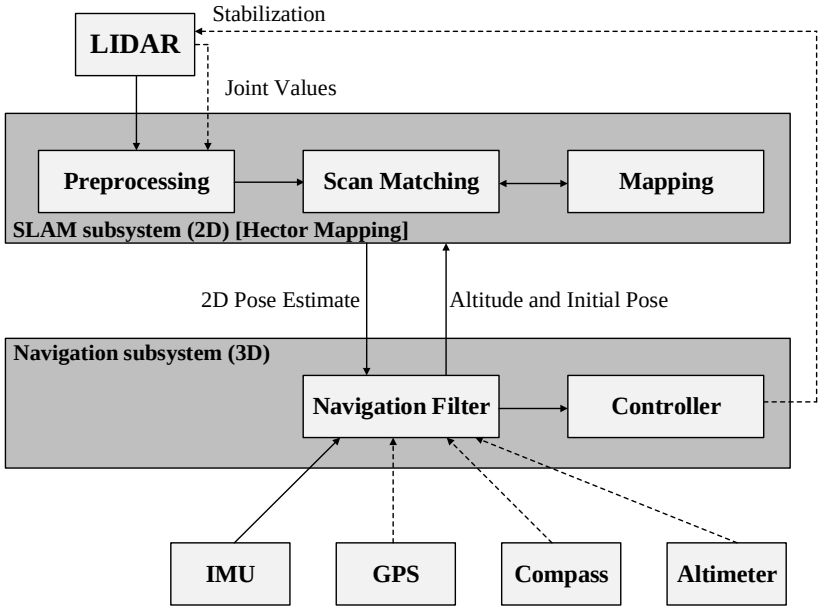


Figure 2.2: Hector SLAM overview [Kohlbrecher et al., 2011]

As Figure 2.2 illustrates, Hector Mapping solves SLAM using only measurements from a Light Detection And Ranging (LIDAR) device. It contrasts with most used SLAM approaches as it does not require odometry.

It must be noted that even though the Hector SLAM system makes arrangements to deal with non-planar, 6-DOF robot movement, the Hector Mapping subsystem can be used in simpler, 3-DOF, 2D environments by ignoring altitude considerations. Hector Mapping subsystem will be analyzed further, briefly describing its operation.

2.2.2.1 Preprocessing

The preprocessing module acts as a filter for the laser data. It filters the laser data based on a given z value of the 3D pose estimate, to ensure the map is produced based on the intended

scan plane. For a planar robot, the value of z remains constant, and so unused in this filter.

This module also filters the laser data based on the sensors maximum and minimum readings, to ensure the gathering of correct readings.

2.2.2.2 Scan Matching

Scan Matching is a process that outputs an estimate of the geometric transformation between the map and a new set of points (laser points). This implementation is based on the Gauss-Newton approach, commonly used in computer vision to solve non-linear least squares problems [Lucas and Kanade, 1981].

The map used in this module is an occupancy grid map, more specifically an array of occupation probability values ($M(p)\epsilon\,[0,1]$) that correspond to the information known about the world. As this module compares laser 2D points and the grid map, an interpolation scheme is considered, to approximate occupancy values in continuous map coordinates. For a continuous map coordinate $P_m = (x, y)$, the occupancy value $M(P_m)$ is approximated by using the four closest grid cells $P_{00..11}$ as depicted in Fig 2.3 and defined in equation 2.1.

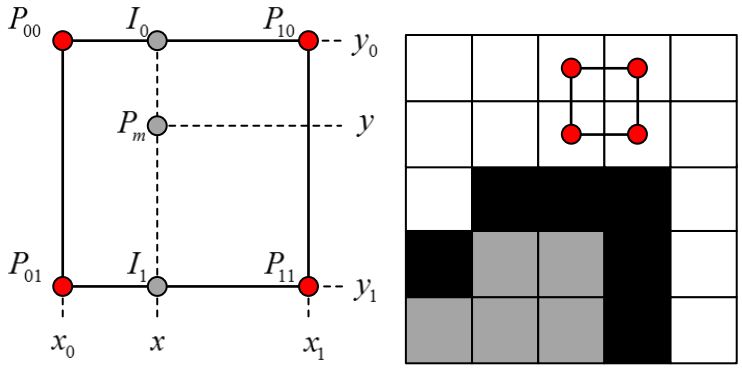


Figure 2.3: Interpolation scheme for the discrete gridmap [Kohlbrecher et al., 2011]

$$\begin{aligned} M(P_m) \approx\ & \frac{y-y_0}{y_1-y_0}\left(\frac{x-x_0}{x_1-x_0}M(P_{11})+\frac{x_1-x}{x_1-x_0}M(P_{01})\right) \\ & +\frac{y_1-y}{y_1-y_0}\left(\frac{x-x_0}{x_1-x_0}M(P_{10})+\frac{x_1-x}{x_1-x_0}M(P_{00})\right) \end{aligned} \tag{2.1}$$

Also, an approximation of the partial derivatives of $M(P_m)$, $\frac{\partial M}{\partial x}(P_m)$ and $\frac{\partial M}{\partial y}(P_m)$, may be calculated in a similar manner, as equations 2.2 and 2.3 define, respectively.

$$\frac{\partial M}{\partial x}(P_m) \approx \frac{y-y_0}{y_1-y_0}\left(M(P_{11})-M(P_{01})\right)+\frac{y_1-y}{y_1-y_0}\left(M(P_{10})-M(P_{00})\right) \tag{2.2}$$

$$\frac{\partial M}{\partial y}(P_m) \approx \frac{x - x_0}{x_1 - x_0}(M(P_{11}) - M(P_{10})) + \frac{x_1 - x}{x_1 - x_0}(M(P_{01}) - M(P_{00})) \tag{2.3}$$

The scan matching module must align the laser scan with the map. In mathematical terms, this problem consists in finding a rigid transformation $\xi = (p_x, p_y, \psi)^T$ that better fits a set of scan endpoints $s_i = (s_{i,x}, s_{i,y})^T$ with the map M. Let $S_i(\xi)$ be the scan point s_i transformed by ξ defined by:

$$S_i(\xi) = \begin{bmatrix} cos(\psi) & -sin(\psi) \\ sin(\psi) & cos(\psi) \end{bmatrix} s_i + \begin{bmatrix} p_x \\ p_y \end{bmatrix} \tag{2.4}$$

Given n scan endpoints, the scan matcher algorithm seeks to find a rigid transformation that minimizes function ξ^*, defined in equation 2.5.

$$\xi^* = \arg\min_{\xi} \sum_{i=1}^{n} [1 - M(S_i(\xi))]^2 \tag{2.5}$$

This function reaches a minimum value for a ξ for which the grid cells corresponding to the transformed scan endpoints are mostly occupied (with $M(S_i)$ closest to 1). Given an initial estimate ξ, the problem is to find $\Delta\xi$ that optimizes the error measure:

$$\sum_{i=1}^{n} [1 - M(S_i(\xi + \Delta\xi))]^2 \tag{2.6}$$

To solve this, it is considered the first order Taylor expansion of $M(S_i(\xi + \Delta\xi))$:

$$M(S_i(\xi + \Delta\xi)) \approx M(S_i(\xi)) + \nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \Delta\xi \tag{2.7}$$

Using equation 2.4, the definition of $\frac{\partial S_i(\xi)}{\partial \xi}$ is:

$$\frac{\partial S_i(\xi)}{\partial \xi} = \begin{bmatrix} 1 & 0 & -\sin(\psi) s_{i,x} - \cos(\psi) s_{i,y} \\ 0 & 1 & \cos(\psi) s_{i,x} - \sin(\psi) s_{i,y} \end{bmatrix} \tag{2.8}$$

Using the defined Taylor expansion, and setting its partial derivative with respect to $\Delta\xi$ to zero, the equation to solve becomes:

$$2\sum_{i=1}^{n} \left[\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi}\right]^T \left[1 - M(S_i(\xi)) - \nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \Delta\xi\right] = 0 \tag{2.9}$$

Solving equation 2.9 for $\Delta\xi$ gives the Gauss-Newton equation for the minimization problem:

$$\Delta\xi = \mathbf{H}^{-1} \sum_{i=1}^{n} \left[\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi}\right]^T [1 - M(S_i(\xi))] \tag{2.10}$$

with Hessian matrix:

$$\mathbf{H} = \left[\nabla M\left(S_i\left(\xi\right)\right)\frac{\partial S_i\left(\xi\right)}{\partial \xi}\right]^T \left[\nabla M\left(S_i\left(\xi\right)\right)\frac{\partial S_i\left(\xi\right)}{\partial \xi}\right] \tag{2.11}$$

Evaluation of this Gauss-Newton equation can be done considering the approximation defined in equations 2.2 and 2.3, and equation 2.8, computing each step towards zero.

The Gauss-Newton algorithm, just like other gradient-based approaches can find solutions in local minima. To avoid such risk, the authors of Hector SLAM use multiple maps with decreasingly lower resolutions that are simultaneously updated using the pose estimates generated by the scan matching. The scan alignment process starts with the lowest definition map level, with that estimate being used for the alignment of the next map level. Figure 2.4 illustrates three map levels.

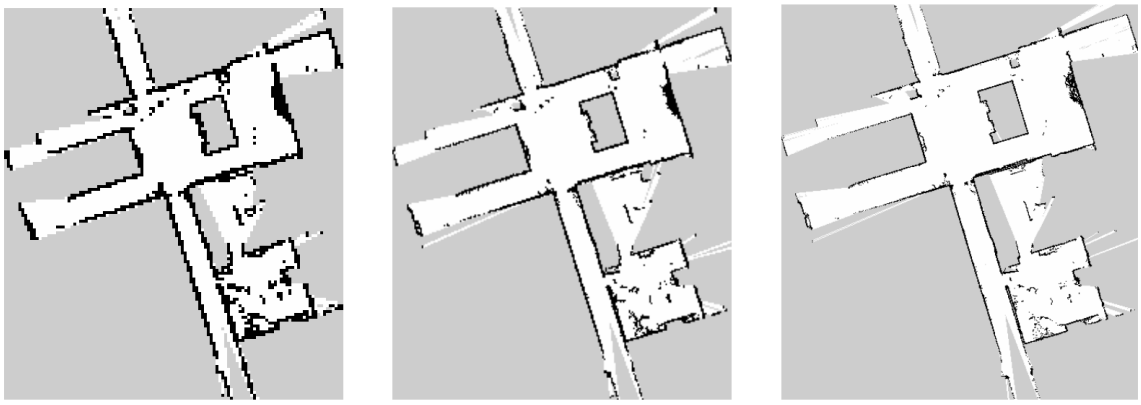


Figure 2.4: Multiple definition map levels [Kohlbrecher et al., 2011]

2.2.2.3 Qualitative comparison with other SLAM implementations

Hector Team SLAM's approach compares fairly well with other well-known approaches.

Hector approach to SLAM differs from Gmapping's (an implementation of the algorithm proposed in [Grisetti et al., 2005]) in sensor requirements, since odometry is needed by the latter. According to [Kohlbrecher et al., 2012], his implementation performs better as Gmapping particle filter because Hector's approach is designed to leverage higher scan rates that modern LIDAR's own. Moreover, for many mobile robots the accuracy obtained by scan matching gives a much more precise displacement estimate than odometry [Kohlbrecher et al., 2011].

Visual SLAM is also an important approach in the state of the art. Compared to algorithms like Gmapping and Hector's, they still lack robustness or require too much processing capacity [Thrun et al., 2005].

Finally, SLAM for robots with six degrees of freedom (DOF) and for three dimensional (3D) worlds are gaining momentum in the SLAM communities. However, they too can be computationally expensive and still need further development [Nuchter, 2010].

2.3 Path Planning

The path planning problem is defined as, given a certain mobile robot, a map, a initial position and a goal, the task of planning a collision-free path that is valid, feasible and preferably a geometrically optimum path that the given robot can track from the initial position to the goal. This task is usually divided in two sub-tasks: global path planning and local path planning. The global sub-task should generate a rough, high-level path between initial position and goal; as for the local path planning problem, the sub task must produce a low-level, online path that solves a segment of the global path, and avoids close sensed objects.

Global path planning solutions are usually based on graph and tree search algorithms such as A* [Kala et al., 2010], Dijkstra's [Gerkey and Konolige, 2008], etc. These algorithms are computationally efficient and provide the theoretical optimal solution for a given cost map. There are also implementations using D*, a algorithm similar to A* with the key difference of allowing dynamic cost in the search tree [Stentz, 1995], allowing better execution in dynamic or unknown environments.

Local path planning is a sensor-based, reactive system with collision avoidance as its main objective. Most common solutions are dynamic window approaches [Fox et al., 1997, Brock and Khatib, 1999, Simmons, 1996], Trajectory Rollout [Gerkey and Konolige, 2008], Nearness diagram [Minguez and Montano, 2004], force (or potential) fields [Pradhan et al., 2006]. In this work the [Fox et al., 1997] dynamic window approach method is used, which is described in a following subsection. To avoid high computational load and to tackle more difficult problems like non-rigid obstacle avoidance, random methods have been proposed; e.g. the rapidly-exploring randomizing tree (RRT) [Lavalle, 1998].

2.3.1 Dijkstra's graph search algorithm

Dijkstra's algorithm is a graph search algorithm that solves the shortest path problem by computing a shortest path tree. Among other applications, it can be used for robot path planning, if a map of the world (be it in grid or graph form) is available. The algorithm is as follows:

- Let there be an initial node A that defines where search tree starts, and let the distance of node N be the cost of the path from node A to node N.
- Let there be also a ternary value for each node. This value defines if a node was not yet visited, if it was already visited or if it is the node being evaluated. Define a structure that enables the listing of nodes with certain characteristics in a set. Define a function that returns a set of the nodes immediately linked to a node.

1. Define the distance of A as zero and the rest of the nodes as infinity;

2. Mark all nodes as unvisited, and set node A to current. Create a set listing all unvisited nodes;

3. Get, for the current node, the set of neighbors that haven't yet been visited. Calculate the distances for the nodes in this set. Compare the distance of the current node with the distance of each one of the nodes in the set and detect the smallest difference.

4. Mark the node of smallest difference as the current node and change its visited value. If the node of smallest difference was already visited then terminate the algorithm as there are no possible solutions. If the selected node is the node N, terminate the algorithm - a solution was found. In other case, continue the algorithm by returning to point 3.

2.3.2 Dynamic Window Approach

The Dynamic Window Approach (DWA) to obstacle avoidance [Fox et al., 1997] is a method that solves obstacle avoidance by discretely sampling the control space (v and ω speeds). Figure 2.5 illustrates the control space sampling that will be described next.

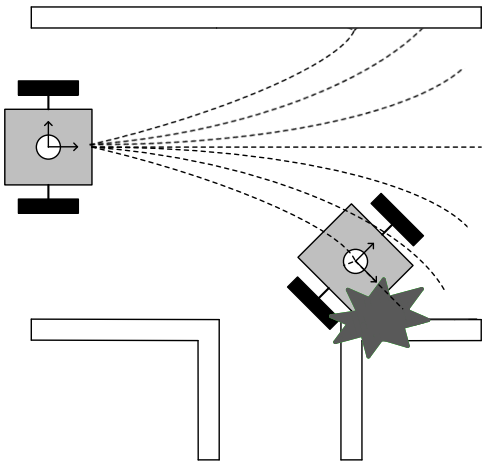


Figure 2.5: Sampling of the control space (adapted from [Marder-Eppstein and Perko, 2014])

Let this control space be U defined in equation 2.12.

$$U = \{(v \in [v_{min}, v_{max}], \omega \in [\omega_{min}, \omega_{max}])\} \tag{2.12}$$

Let $a_{v_{min}}$ and $a_{v_{max}}$ be the minimum and maximum accelerations, respectively, for linear velocity and let $a_{\omega_{min}}$ and $a_{\omega_{max}}$ be the minimum and maximum accelerations, respectively, for angular velocity. For a given control loop period T and current robot speeds (v_t, ω_t), there is a control space U_R that defines the reachable velocities during the duration of a control loop. U_R is defined in equation 2.13.

$$U_R = \{(v \in [v_t + a_{v_{min}}T, v_t + a_{v_{max}}T], \omega \in [\omega_t + a_{\omega_{min}}T, \omega_t + a_{\omega_{max}}T])\} \tag{2.13}$$

Finally, for each v and ω let it be known the distances from an obstacle: d_{obs} and θ_{obs}. Using this knowledge, a control space U_A that defines the set of admissible controls can be defined (see equation 2.14).

$$U_A = \left\{ \left(v \leq \sqrt{2 d_{obs} a_{v_{min}}}, \omega \leq \sqrt{2 \theta_{obs} a_{\omega_{min}}} \right) \right\} \tag{2.14}$$

Candidate set of controls is defined by the control space $U_C = U \bigcap U_A \bigcap U_R$. Given this control space, the next step is to select a control pair u in the control space that maximizes an objective function. Given cost functions $TowardsGoal(u)$ that favors controls approaching goal, $AvoidObstacle(u)$ that favors controls moving far from obstacles and function $BestVelocity(u)$ that favors a set of speeds, a control pair u is chosen so that it maximizes the objective function in equation 2.15.

$$G(u) = \alpha_1 \cdot TowardsGoal(u) + \alpha_2 \cdot AvoidObstacle(u) + \alpha_3 \cdot BestVelocity(u) \tag{2.15}$$

2.4 Semi-autonomous Control

According to [Antsaklis et al., 1991], "Autonomous controllers have the power and ability for self governance in the performance of control functions. (...) There are several degrees of autonomy. A fully autonomous controller should perhaps have the ability to even perform hardware repair, if one of its components fails". A semi-autonomous control system, on the other hand, is a system that explicitly requires some kind of outside assistance, be it from a human or another system.

In [Rogers et al., 1996], semi-autonomous control is defined as an advancement from teleoperation by reducing the amount of supervision by the operator and as an approach to autonomous robots, letting the human retain high level operations such as task specification, but handing down routine and safe portions of tasks to be handled autonomously. Furthermore, [Ong et al., 2005] summarizes semi-autonomous control into two (and one extra) categories:

- a parallel or shared control type, where portions of a certain task are handled by a machine agent, and the other parts are handled by a human agent;

- a serial or traded control, where the controller decides over time if the control lies on the human side or the machine side completely;

- a mixture of the two previous types, where sharing and trading of task both occur (e.g. Collaborative Control).

The following three subsections briefly describe each one of these categories.

2.4.1 Shared Control

Shared control, classified as a parallel type of semi-autonomous control, is a paradigm of semi-autonomous control that shares the effort involved in the decision task between human and machine, so that some tasks are made by the user and others by the machine, side by side.

A simple example of its application is a military application robot gun that automatically chooses what to aim and track, and leaves to the human the control of the trigger, leaving each task to the agent that better performs it. Both agents (human and machine) work in a shared, parallel fashion. Figure 2.6 illustrates the example of a shared control loop.

As compared to manual control, shared control relieves the human of easily automated tasks while allowing him direct control of more hard-to-control activities such as manipulation of parts [Hirzinger, 1993, Lee, 1993] or navigation in cluttered areas [Bourhis and Agostini, 1998,Bruemmer et al., 2003]. It also may simply give the human more flexibility in a automated task.

Other examples of applications of shared control are vision-based guidance control [Papanikolopoulos and Khosla, 1992, Hoppenot and Colle, 2002], safeguarding control [Krotkov et al., 1996, Fong, 2001, Wasson and Gunderson, 2001] and behavioral control [Bourhis and Agostini, 1998, Bruemmer et al., 2003].

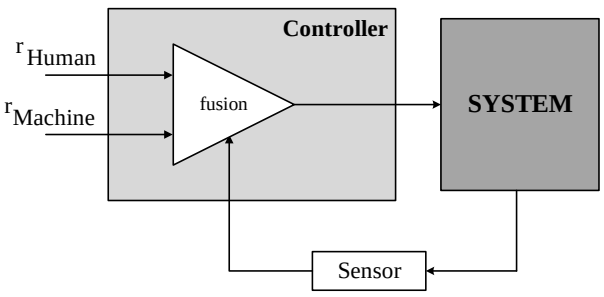


Figure 2.6: Shared control loop

2.4.2 Traded Control

As opposed to Shared Control, Traded Control is classified as a serial type of semi-autonomous control. It is based on a mutually exclusion mechanism for human and machine, exchanging control between the two. Simply put, a traded controller is an algorithm that decides if user commands should be put to action or not [Ong et al., 2005].

In the presence of multiple situations such as goal deviations, addition or deletion of goals, modifications in goals importance, incompetent performance of tasks/restrictions and to nullify potential harmful commands/situations; control based on the demand of the task is exchanged between the human and the robot. Briefly, there are two main conditions when control is traded between human and robot. First, when considering a navigation task [Bruemmer et al., 2003]

if the robot takes a wrong direction, the human may interpose and assume the control from the robot, giving it a new direction movement. The second condition occurs when the human issues commands that could cause damage to himself. In this situation the robot may revoke undesired commands and stop the movement. Given this perspective, this traded control allows both human and robot to support each other. Figure 2.7 illustrates a control loop applying the trading philosophy [Ong et al., 2005].

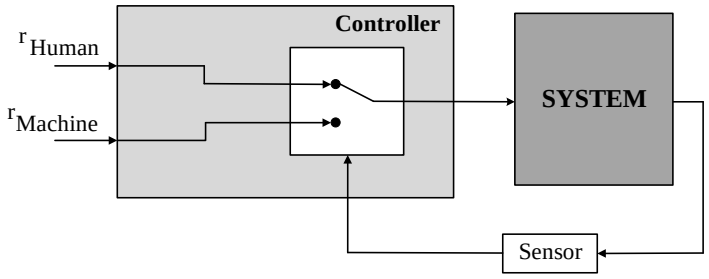


Figure 2.7: Traded control loop

2.4.3 Collaborative Control

In [Fong, 2001], a mixture of the previous two types of semi-autonomous control is explored, defining a new control model called Collaborative Control. This concept tries to break away from the concept of robot as a tool, and to approach the robot as a partner by developing a channel of communication between human and machine agents.

Considering this combined configuration, it is important to underline that both shared and traded control interact. This type of combined control is observed for example in the aircraft autopilot system [Billings, 1996], where during the cruise phase, the pilot exchanges the control over the controller, the autopilot is responsible for holding the altitude, whereas the pilot adjusts the heading, sharing control at the same time. Collaborative Control can adjust its method of operation taking into account the needs of the current situation, so it enables a accurate sharing and trading of control. Taking for example a situation where the robot is not able to perform accurately, it has the possibility to hand control to the human, enabling the work to be dynamically allocated to the human or the robot as the tasks take place [Ong et al., 2005].

In [Fong, 2001], Fong refers what are the main appeals in considering the Collaborative Control. This approach definitely enables humans and robots to work better together, being this an important feature when success strongly depends on joint task performance. Also, robots can use humans as a resource for non-command functions such as perception. Taking into account the superior human capabilities in some points like perception, it seems to be beneficial to take advantage of them, considering that those same capabilities are poorer in robots. Fong also refers that this type of combined control can better accommodate multiple users and adjust its autonomy.

In order to build a successful Collaborative Controller (CC), one must satisfy both human and robot needs. Fong defines seven key design issues that should be addressed to construct a useful system of this kind:

1. Dialogue: A collaborative control system must have a good communication channel, i.e., the human and the robot should be able to exchange information effectively, allowing questions to be asked and to question each others responses.

2. Awareness: The robot should have the knowledge of what it can and cannot do, being aware of its abilities. On the other hand, it should also have knowledge of the human limitations.

3. Self-Reliance: The robot must be self-reliant to a certain degree, meaning that the robot should not rely on the human to always be available or always provide accurate information. It must be able to maintain its own safety.

4. Adaptiveness: By design, the collaborative controller should be equipped with tools to prepare it for certain changes in its working environment. With the same philosophy, it should be prepared for different characteristics of the human with whom it shares control: differences in experience, skill and training should be accounted for.

5. Human-Robot Interaction (HRI): With collaborative control, HRI is dynamic and broadly acceptive. The robot may ask for assistance but the human is not required to answer. He may answer, but he may as well not. For systems like this, humans and robots need to understand each other through interaction, rather than by design or inspection.

6. User Interface Design (UID): Since communication has a key role in the collaborative control, UIDs are a key design aspect. It may be interesting to allow the robot to have a mechanism to attract user's attention. Also, information to the user and from the user could be displayed in different forms, depending on the situation.

7. Control and Data Flow: As opposed to control of teleoperation systems, where the flow of control is clear (the operator retains ultimate authority), in collaborative control systems the control is allowed to be negotiated. Human commands may also be viewed as approximate and with error. With this being established, there must be rules to solve situations where robot and human do not agree. On the other hand, data handling should be flexible, to prevent misunderstanding between robot and human.

2.4.3.1 Applications of Collaborative Control in the literature

Since [Fong, 2001] was published, several other works have been done using this concept of human-robot collaboration. In [Fong et al., 2003] advances are made allowing the collaborative control of a multi-robot team. Later, social robot (Leonardo) presented in [Breazeal et al., 2004]

and [Hoffman and Breazeal, 2004], was developed to interact with people in a collaborative manner. Figure 2.8 shows Leonardo working together with a human on a button-pressing task.

Figure 2.8: Leonardo collaborating in a button-pressing task [Breazeal et al., 2004]

In a different scope, several works [Monferrer and Bonyuet, 2002, Zeng et al., 2008, Montesano et al., 2010, Lopes et al., 2013] have been implementing collaborative control schemes towards intelligent wheelchairs. In particular, a Collaborative Controller designed to work with BCI commands has been developed at ISR [Lopes, 2012].

2.5 Human-Machine Interfaces

A Human-Machine interface is the module of a machine's system designed to handle human-machine interaction. Human-machine interaction, term popularized in [Card et al., 1980], is a multi-discipline field of investigation that focuses on interaction between human and machine, where interaction means a two way exchange of information. When HMI applies to robots or computers, some authors use the terms Human-Robot Interface (HRI) or Human-Computer Interface (HCI), respectively. In this work these terms are used interchangeably.

When interacting with a robotic wheelchair, several interfaces have been used, such as the conventional joystick, touch-screens, voice recognition and more (see Table 2.1). When people with certain disabilities or handicaps are the main target of a HMI, several adaptations must be considered to let those users be able to successfully operate those interfaces. For example, patients with ALS or CP disorders see their muscular activity diminished or absent, in the most advanced stages of the disease [Hoffmann et al., 2008, Nijboer et al., 2008]. In this cases, interfacing with joysticks or voice recognition becomes an arduous task, so different solutions have to be taken into account, such as the use of electroencephalography (EEG) signals, the detection of small hand movements or eye tracking. An interface that uses EEG brain signals to estimate user commands is called Brain-Computer Interface (BCI). This interface is one of the last options available for the patients characterized before, and so it has been an interface where research has been growing.

2.5.1 Brain Computer Interfaces

Work with monkeys in 1969 [Fetz, 1969] have already showed that signals from singled cortical neurons could be used to control a needle. Investigations relying in humans began in the 1970s and it was only in 1973 that Vidal [Vidal, 1973] asked if considering only the use of electrical brain signals it would be possible to carry information in human-machine communication for controlling devices such as prosthesis. The initial progress in Human BCI was slow and really limited principally by computer capabilities and even the few knowledge of brain physiology [Shih et al., 2012]. Since then, several developments have been made in both areas. In 1988, Farwell and Donchin [Farwell and Donchin, 1988] showed that P300 event-related potentials could be used such way that allowed healthy volunteers to interface a spelling device and successfully spell words on a computer screen. The developments observed both in brain understanding as in computer technologies culminate nowadays in more advanced applications as for example the implantation of microelectrodes arrays in the primary motor cortex of a young man with complete tetraplegia, in 2006 [Hochberg et al., 2006]. Using the signals obtained from the electrode array the patient could successfully open e-mail, operate a television and perform rudimentary actions with a robotic arm.

Nowadays, BCI research is growing in an extremely rapid rate, and different applications are being studied and implemented, such as control tasks for game interfaces or even navigation through the presentation of paradigms with direction cues [Hoffmann et al., 2011, Pires et al., 2011, Pires et al., 2012].

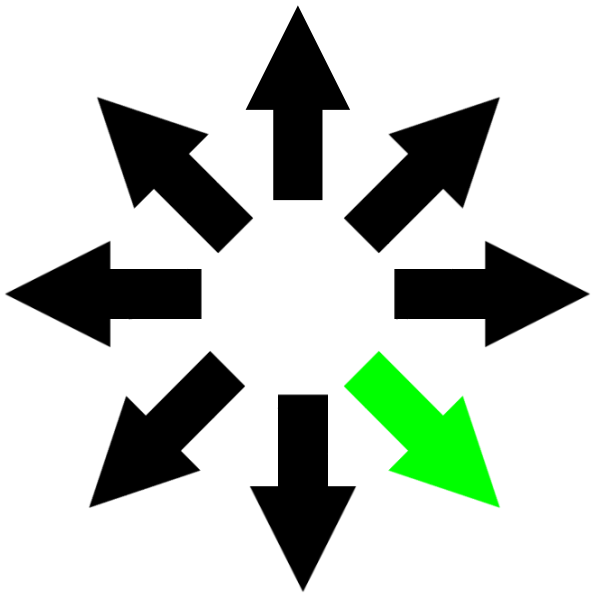


Figure 2.9: Example of an visual oddball paradigm (adapted from [Pires et al., 2008])

An oddball paradigm is usually used to evaluate the human intent. This type of paradigm is mainly used in evoked potentials analysis, where the stimuli presentation is used to assess the

brain reaction to unpredictable events. To do so, the stimuli events are presented in a random fashion. One example of an oddball visual paradigm is represented in Figure 2.9, where the target event is exhibited by a green flashing color.

One way to develop a BCI using EEG data is to detect the event-related potential (ERP) P300 elicited by sensory stimuli. ERP are EEG transient voltage shifts induced by internal events. In the majority of users, sensory stimuli presentation in an oddball paradigm produces a positive peak in the EEG signal, 300 ms after the onset of the stimulus, being this component named as P300. It is a common ERP component used to develop BCI and it consists in a positive voltage shift usually observed in response to target stimulus (see Figure 2.10).

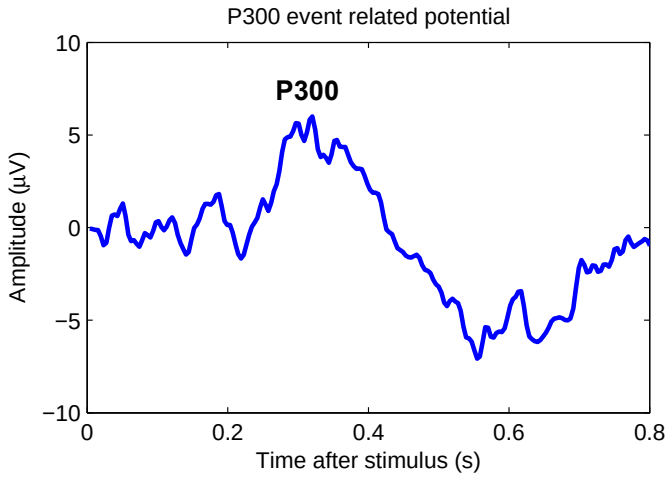


Figure 2.10: P300 event related potential (adapted from [Pires, 2011a])

Chapter 3

ANS Architecture

This chapter briefly describes the architecture of the Assistive Navigation System (ANS) that is the aim of this work. It is done at a conceptual level, trying to attain the predefined goals. To do so, these goals are reviewed, and the requirements needed to achieve them are examined. At this phase of development, implementation issues are not yet considered; such concerns will be tackled in Chapter 5.

3.1 System goals and requirements

The assistive navigation system is designed to relieve and support the user in the task of operating a robotic wheelchair. For the particular ANS developed in this work the main objectives taken into account were:

1. Allow users in different stages of disability to easily and comfortably communicate with the RW;

2. Create a control module that merges user and machine's abilities;

3. Incorporate modules of perception and navigation allowing a safe and efficient navigation;

4. Ensure that the physical layer is capable of supporting such system.

To achieve the defined objectives several requirements have been identified. To allow users in different stages of disability to communicate with the wheelchair, a generic Human-Machine Interaction protocol was developed so that different Human-Machine Interfaces can be accepted, such as Brain-Computer Interfaces and others. For a control module capable of merging user and machine's inputs, a Collaborative Controller was developed, that is further described in Chapter 4. Taking into account that a safe and efficient navigation is our aim, a perception module for obstacle detection and positioning in the environment that provides information to the navigation module was developed and are described in section 3.5 and 3.6. The theoretical design expected from the physical layer is considered in this Chapter; implementation concerns are further debated in Chapter 5.

Figure 3.1 summarizes the main goal of the system, highlighting the essential role of communication between human and machine in this architecture.

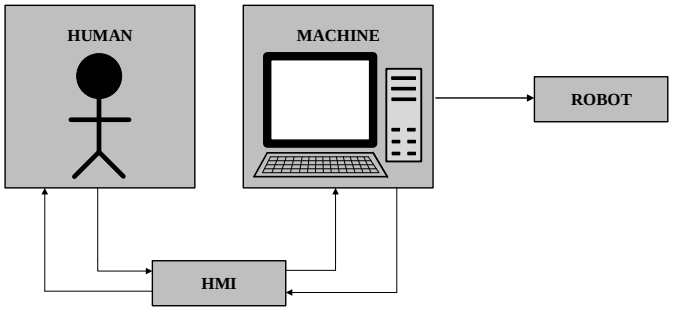


Figure 3.1: Communication channel between human and machine

3.2 Generic System overview

Figure 3.2 represents the system overview, with the five fundamental modules of the designed system architecture, described in the following sections. The scheme represents the information flow between the modules and how they interact.

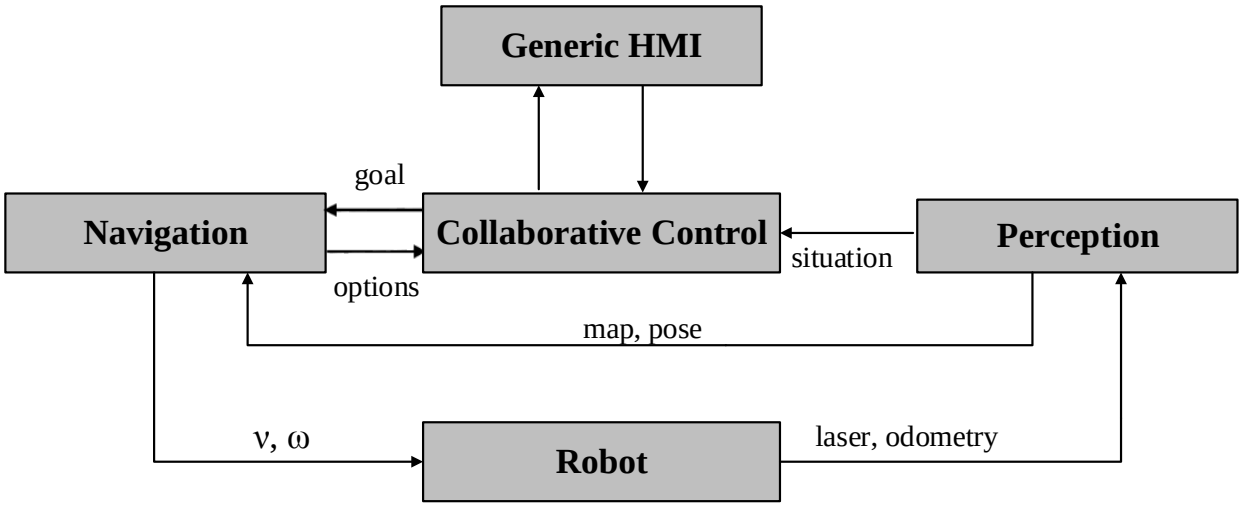


Figure 3.2: Assistive Navigation System Overview

3.3 HMI protocol

Human-Machine Interfaces represent a communication channel between a machine and a human, such that human intent can be perceived by the machine. This machine component can take various forms such as computers or robots. Also, the human intent can be extracted from different methods, such as EEG signal processing (known as BCI) or eye tracking.

In this dissertation our concerns did not rely in the methods used to determine the human intent, but rather on the set of controls allowed and the final choice of the user. To accommodate this aspect, and to allow different HMIs, a protocol was designed. For this protocol,

the following set of controls was defined: STOP, WC, HALL, HELP, YES, NO, RIGHT90, RIGHT45, FORWARD, LEFT45 and LEFT90 (see Figure 3.3). Any HMI fulfilling this protocol requirements will work with the implemented system.

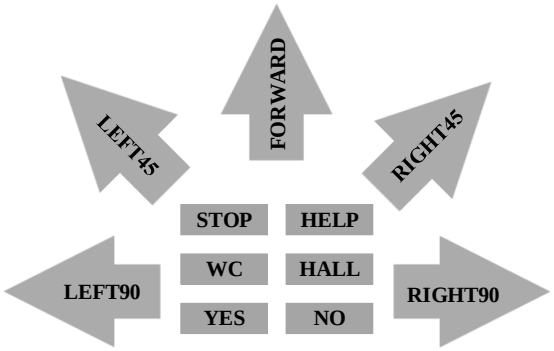


Figure 3.3: Example of a visual display paradigm for the proposed protocol

3.4 Collaborative Controller

The Collaborative Controller module has a central role in the navigation system. It must deliberate on the perceived situation and decide when to ask the user or the navigation module for commands. This module receives information from the perception module to update the current situation; it trades information with the user through the HMI; and lastly it receives path information from the navigation module. Based on all the collected information, the CC module outputs the navigation goal. The CC design is described in detail in Chapter 4.

3.5 Navigation module

The considered navigation module has its core function path planning and a module for executing those paths (path follower). It receives information from the perception module concerning the existence of unexpected obstacles in the way, when executing a path, and localization data to perform the path tracking. In some cases, like the presence of unexpected obstacles, the path planning must output multiple path options to the CC.

The path planning module takes into account the received goal from the CC module and the chosen path is delivered to the path following module. The function of the path following module is to calculate the linear velocity $v(t)$ and the angular velocity $\omega(t)$ and deliver that information to the robot.

Figure 3.4 represents the navigation module components and its interaction with the other modules.

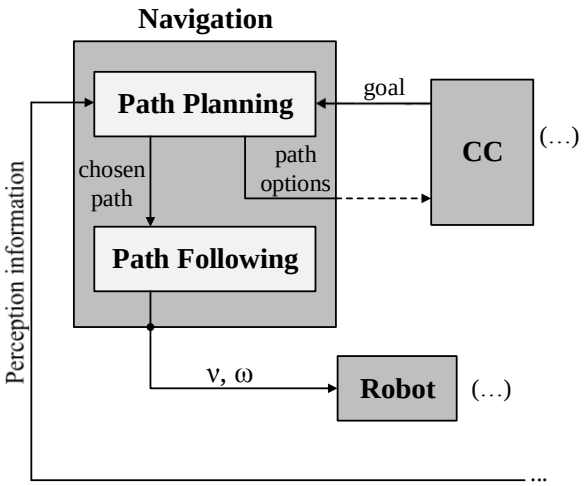


Figure 3.4: Navigation Module components and its interactions with the other modules

3.6 Perception module

The main objectives of the perception module are robot localization and detection of user-aid-required situations (UARS). Obstacle detection, forked path and deadlocked way are some examples of UARS.

The perception module is composed by SLAM, required to produce an estimate of the current robot pose, and it can also integrate several UARS detectors. In this work unexpected obstacles, multiple path and deadlock detections were considered. Nevertheless, more UARS detectors can be further added to improve the system robustness.

This module receives sensor data from the robot, this data can comprise for example laser readings, odometry data, image or others. Internally, SLAM builds a local map that may aid detection of UARS. For example, obstacle detection can compare the original map with the current map given by SLAM. When a UARS is detected, the module outputs an notification to the collaborative controller, which acts accordingly. It also outputs pose estimation for navigation purposes.

Figure 3.5 shows the internal components of the perception module.

3.7 Robot

The robot is the physical layer of the system and it establishes the link between the system and its environment. Once the implemented system was designed to be used with a motorized wheelchair, only differential driving was considered, simplifying the navigation task.

The robot receives from the navigation module linear and angular velocities which are converted to speeds to the left and right wheel, through inverse kinematics. Sensors mounted on the robot provide data that is then transmitted to the perception module.

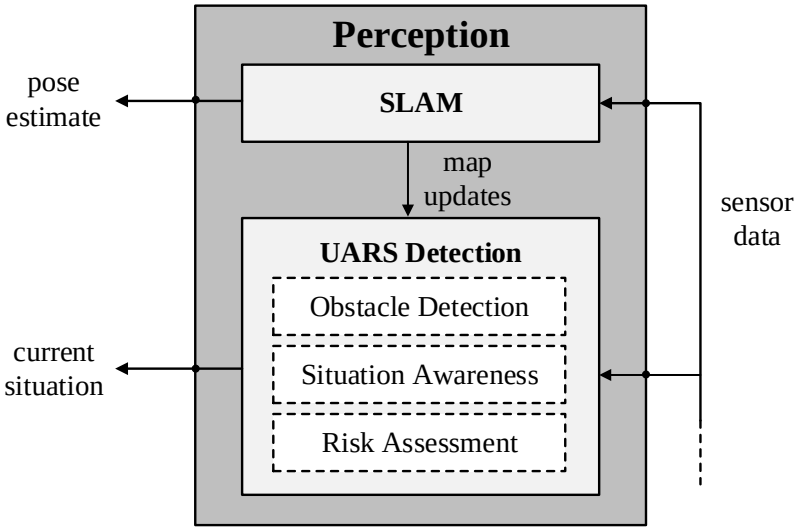


Figure 3.5: Perception Module

Figure 3.6 shows the data flow from and to the robot.

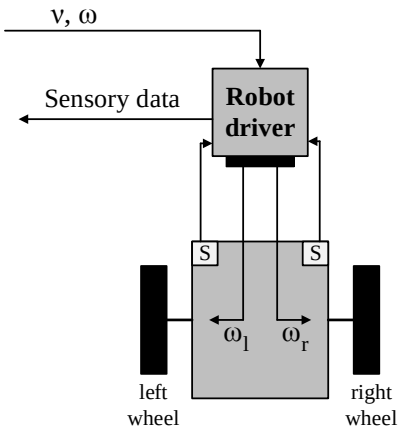


Figure 3.6: Robot's role in the system (S stands for sensors).

Chapter 4

Collaborative Controller

In this chapter, the Collaborative Controller design is described in detail. In the first section, an introduction to the module is given, highlighting its requirements and operation. In the next section, the CC design is discussed and its main components are described. Its central role in the system is analyzed in the last section, with a global perspective of its inputs and outputs.

4.1 Controller design issues

The collaborative controller can be dubbed as the mediator between human and machine. In Chapter 2, we reviewed the seven key design issues established in [Fong, 2001] that should be considered during design phase of a Collaborative Controller. In order to design a successful controller, these issues will be considered in this section, one by one, and a solution for each issue will be sought. These issues must also be tackled in accordance with the design thought in Chapter 3.

Design begins by defining a limited number of situations to be considered, in order to achieve a feasible implementation. For this work we will consider three UARS:

1. Detection of unexpected obstacle
2. Fork in the path
3. Deadlock of the way

The controller designed in this chapter must solve this situations by requesting the user's assistance. With UARS detection reviewed, it is now important to explore related design issues in order to tackle them.

The first design issue to consider is *Dialogue*. The need of a good communication channel is fairly understandable. Even though the robot performs well in some tasks, its understanding of the world is very limited. To enable the human to help it, a good, well-defined communication protocol must be defined. Following, other issue to consider is *Awareness*, which means that the CC must understand what both human and robot lack. *Self-reliance* is the third issue listed in [Fong, 2001]. This means having the ability to maintain security by its own means. This issue can be viewed from very different standpoints (electrical conditions, environment hazards, collisions, etc). The primary concern of this design will be navigation and planar

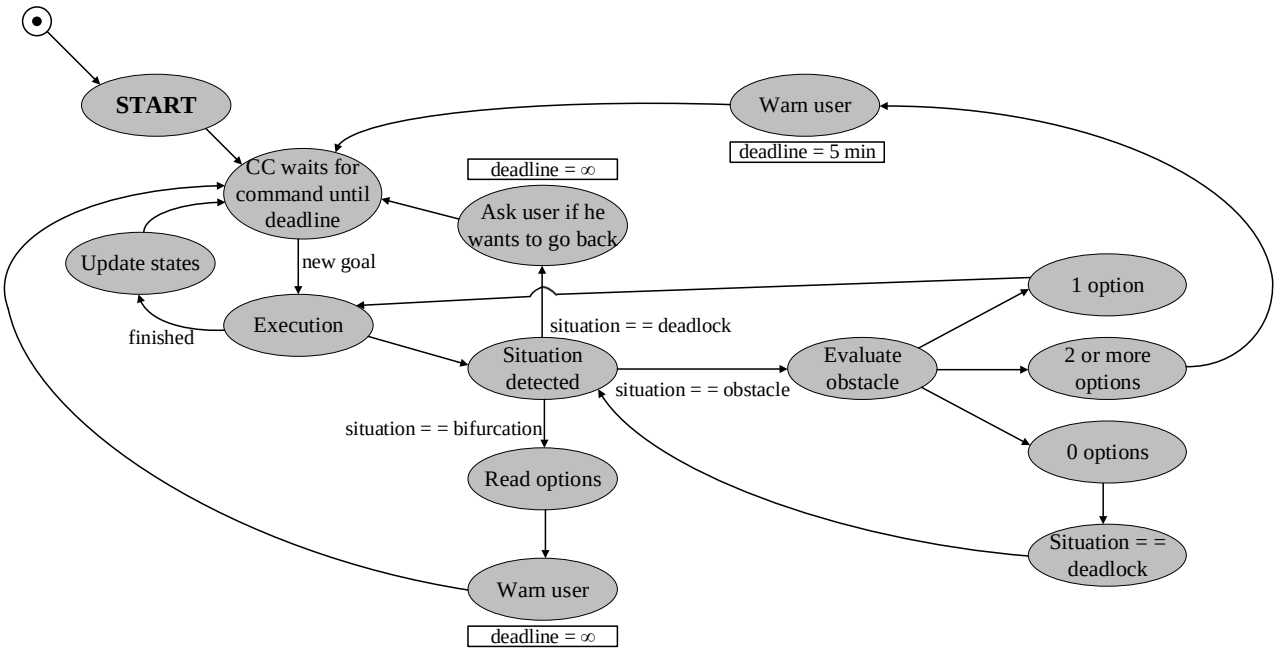


Figure 4.1: Collaborative Control course of action

collision avoidance problems. *Adaptiveness* relates to the ability of the controller to accept users with different characteristics. A possible solution to this problem consists in defining a list of skill levels, and attribute a skill level to each user. *Human-Robot Interaction* issue is related with improving the status of HRI to a more natural, partner-to-partner level, and further way from the *robot as machine* concept. Finally, there is a issue to have in mind with the *User Interface Design*. This is as much an issue to HMI design as it is for the Collaborative Control design. It concerns about maximizing usability. It will not be an explicit concern in the design of this controller, as it obeys a protocol defined in Chapter 5.

Before delving into the module design, it is important to understand how is it intended to work. Figure 4.1 shows a finite-state machine describing the intended operation of the system, taking into account some of the issues discussed above.

Dialogue is tackled with the existence of a two-way information exchange within a well defined protocol, and the use of an appropriate HMI, in this case a BCI which is adequate for people unable to control their motor functions. The use of such HMI is very challenging due to the low information transfer rate and an associated error rate. In other words this means that only sparse commands can be provided and some of them may be not reliable.

The robot must identify situations where aid is required and ask the user for assistance in such situations. This is directly related with *Awareness*, which is well accommodated by the predefined UARS.

Self-Relience is tackled by allowing the robot to act by its own, after a deadline defined in some situations. In all situations not requiring user aid, the RW works as if it was autonomous.

The next sections will define a control design that complies with discussed issues.

4.2 Collaborative Control Design

The Collaborative Controller is designed as a two layer module; a Virtual Constraint layer and an Intent Matcher layer. It also contains a module to send information to the human, allowing the system to have a HRI. Figure 4.2 outlooks the design, and illustrates the flow of information and communication channels. The module is updated with the current perceived situation by the perception module and with path options by means of communication with the navigation module. By informing the user of the situation, and with his feedback, it decides what action should be taken by filtering and matching the intent of such feedback. This process is done by the Virtual Constraint and Intent Matcher layers, described in the two next subsections. A Dialogue Manager module is also designed to deal with sending information to the user.

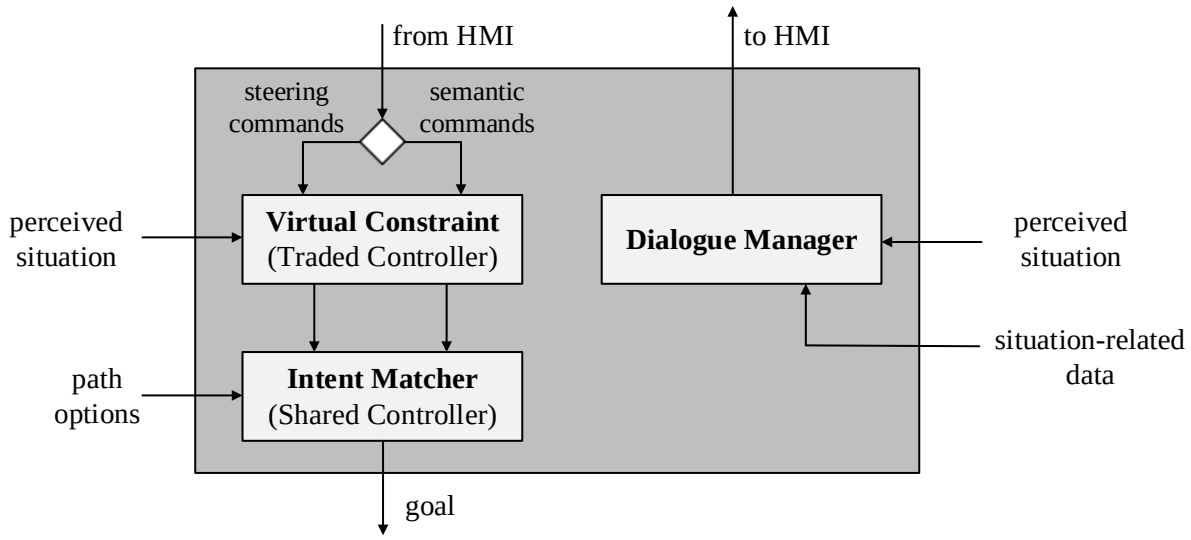


Figure 4.2: Overview of the Collaborative Controller module

4.2.1 Virtual Constraint (Traded Controller)

The Virtual Constraint (VC) layer of the CC is a rule-based filter. It acts as a Traded Controller, effectively denying or allowing the enforcement of user commands. For each user skill level, there should exist a rule table (like Table 4.1) that defines what commands are passed on (coded by 1s) to the next layer and what commands are cut off (coded by 0s).

4.2.2 Intent Matcher (Shared Controller)

The Intent Matcher (IM) module, as the name suggests, tries to find an action that better approximates the user's command, given a set of possible actions. Its output depends on the current situation. It acts as a Shared Controller by joining user coarse directions with definite geometric goals, used to create the detailed path plans.

Table 4.1: Rule-based Traded Controller

	$\uparrow$	$\nwarrow, \nearrow$	$\leftarrow, \rightarrow$	STOP	HELP	HALL	WC	YES	NO
Driving	0	0	0	1	1	1	1	0	0
Forking	1	1	1	0	1	1	1	0	0
Obstacle	0	1	1	0	1	1	1	0	0
Deadlock	0	0	1	1	1	1	1	1	1

Some commands have a very specific meaning and lead to the same action, apart of the situation. The commands "HELP", "HALL" and "WC" are always accepted by the VC because their meaning is clear. The robot response to "HELP" should launch a helping task. This could result in calling someone for help, or launching a dialogue between human and robot to discern what kind of help the human needs. "HALL" and "WC" command are explicit goal commands, and lead the CC to order the navigation module to drive the robot to those goals.

For the other commands, the current UARS must be taken into consideration, in order to find the best matching action to the user's command.

First situation to consider is when there is no UARS, which means that the robot is in *Driving* situation referred in this subsection. In this situation only semantic commands are accepted. Receiving a "STOP" command halts the navigation process and establishes a forking situation, with the last stable point and old goal as the available options.

For the case of *Forking*, two kinds of commands must be taken into account: steering and semantic commands. Semantic commands allowed are "HELP", "HALL" and "WC", with actions already described above. For steering commands, there is an algorithm that finds the best matching option. It develops as follows:

For each steering commands $(\rightarrow, \nearrow, \uparrow, \nwarrow, \leftarrow)$, there is a respective correspondent angle command $\theta_{HMI} \in \left\{-\frac{\pi}{2}, -\frac{\pi}{4}, 0, +\frac{\pi}{4}, +\frac{\pi}{2}\right\}$. The state of the robot (x, y, θ), and n possible goal positions (x_i, y_i) with $i = 1..n$ are also known. For each goal, a corresponding angle θ^i_{goal} is computed by equation 4.1. Each angle θ^i_{goal} establishes the directions of the available goals in the robot's horizon.

$$\theta^i_{goal} = \theta + atan2(y_i - y, x_i - x) \tag{4.1}$$

An error function is then defined by $e^i_{HMI} = \theta_{HMI} - \theta^i_{goal}$. The goal i^* that better corresponds to the received steering command is then computed by the objective function defined in equation 4.2. Figure 4.3 illustrates the problem.

$$i^* = \arg\min_i \left(e^i_{HMI}\right) \tag{4.2}$$

When an *Obstacle* UARS is detected, the obstacle is first evaluated, finding possible openings to avoid the obstacle. In this implementation, as the obstacle is considered to be one solid object, only four outcomes are considered: the obstacle is avoidable by the right, by the left, by both

sides or by none. Furthermore, the user's assistance is only called if both sides are available. Taking this assumption into consideration, the matching algorithm needs to determine if the user requested left or right. It is predefined that $\leftarrow$ and $\nwarrow$ commands mean left; $\nearrow$ and $\rightarrow$ mean right.

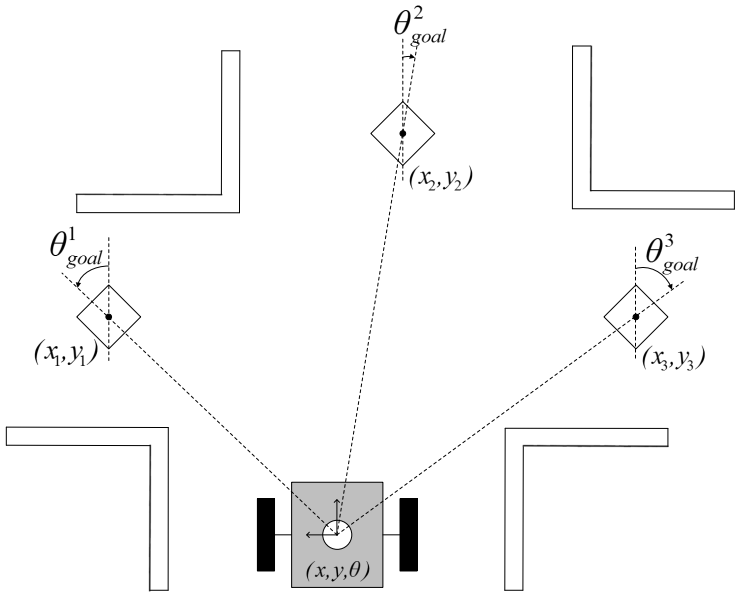


Figure 4.3: Forking situation and corresponding variables

When a *Deadlock* is perceived, the user is asked if he desires to return to the last available node. "NO" command will leave the RW pose unchanged. "YES" command will trigger a half turn and the definition of the last goal as the current goal. If a rear laser exists, backward motion may also be admissible. Commands $\leftarrow$ and $\rightarrow$ do the same as "YES", but further define to which side should the half turn happen.

Chapter 5

Implementation for RobChair

After the ANS and CC design phase, the development and implementation of the defined techniques and algorithms were carried out, using RobChair, the RW platform of ISR [Lopes et al., 2013]. Although a generic approach was taken when considering the ANS in previous chapters, the practical design will now focus specifically on RobChair. Given this fact, several implementation choices were simplified. Recent software and hardware updates done in RobChair defined ROS as the robot middleware, with arguments in favor and against discussed in [Gonçalves, 2013]. It was on top of that decision that all software was designed, and so most software of the ANS was written in C++. Furthermore, the existing sensors and hardware played a decisive role in implementation decisions.

The first five sections will detail the software implementations in ROS environment of the HMI protocol, navigation, perception and collaborative controller modules and simulation of this system with gazebo. The final section describes the physical layer available components as well as adaptations made to help successful implementation of the system.

5.1 HMI integration

For communication with HMIs that incorporate the protocol described in Chapter 3, a communication protocol able to operate on the network must be defined. A TCP/IP protocol was defined in 2011 specifically for BCI two-way communication [Pires, 2011b]. It defines the same type of commands as the defined protocol in this work (primarily because the protocol proposed in this work is based on that BCI).

The proposed TCP/IP protocol for this work is a simplified version of the last one but also adds to the previous one a new Request Type that allows the transmission of an RGB image (3 bytes per pixel, 1 byte per pixel color). This new protocol can be used to show the user a picture of the environment for various purposes, namely to show with detail the current view of the way, allowing him to analyze in an easier way a given situation (see Figure 5.1 for an example).

Tables 5.1 and 5.2 list the communication protocol, comprising request and command byte sequences, respectively. In the special case of image sending, the code sent is variable in size, being defined in the second and third bytes. Length is defined by sz_x and width by sz_y.

This communication protocol was implemented in a ROS node, *hmi_ comm*, subscribing to

Figure 5.1: Example of the reception of the environment image

Table 5.1: HMI Communication Requests (from the ANS to HMI)

Request code	Description
0-0	Inactive (value 0 is special)
1-0	Reserved
2-sz_x-sz_y-IMAGE($3 \times sz_x \times sz_y$)-0	Image sending
3-0	Not defined

/hmi_out messages, coming from the CC and publishing /hmi_in messages to the CC. Figure 5.2 illustrates this simple node.

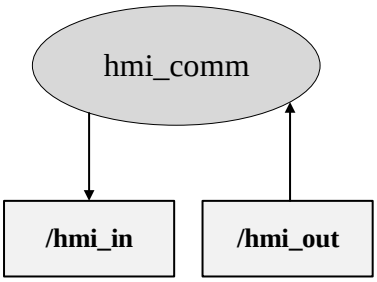


Figure 5.2: ROS node hmi_comm, a HMI driver

These concepts were proved using a visual HMI implemented in MATLAB that worked as a simulator of the BCI system. As two way communication is not yet implemented for such BCI, the full two-way communication specification could not be tested with the pair BCI - ANS system.

5.2 Navigation module

A description of this module requirements has already been presented in Chapter 3. Navigation requires a planning algorithm that given the current pose estimation of the robot and a final goal must produce a path plan. Given that path plan, a path follower algorithm is necessary to execute it, by computing robot speeds to achieve such plan.

Table 5.2: HMI Communication Commands (from HMI to the ANS)

Command code	Description		Command code	Description
0:19-0	Reserved		45-0	STOP
20:31-0	Not defined		46-0	YES
32:40-0	Reserved		47-0	LEFT90
41-0	FORWARD		48-0	LEFT45
42-0	RIGHT45		49-0	WC
43-0	RIGHT90		50-0	HELP
44-0	NO		51-0	HALL

These tasks are executed by the navigation stack available in ROS distributions [Marder-Eppstein and Perko, 2014], more specifically by the *move_base* package. The functioning of this package is described in subsection 5.2.1. Furthermore, the role of our navigation module is also to deliver path options to the collaborative controller module. For that, a submodule able to compute such options is required. This submodule will be called *option_server* node and is further discussed in subsection 5.2.2.

The general view of the navigation module implementation is shown in Figure 5.3, referring both packages and information flow between them and the outside.

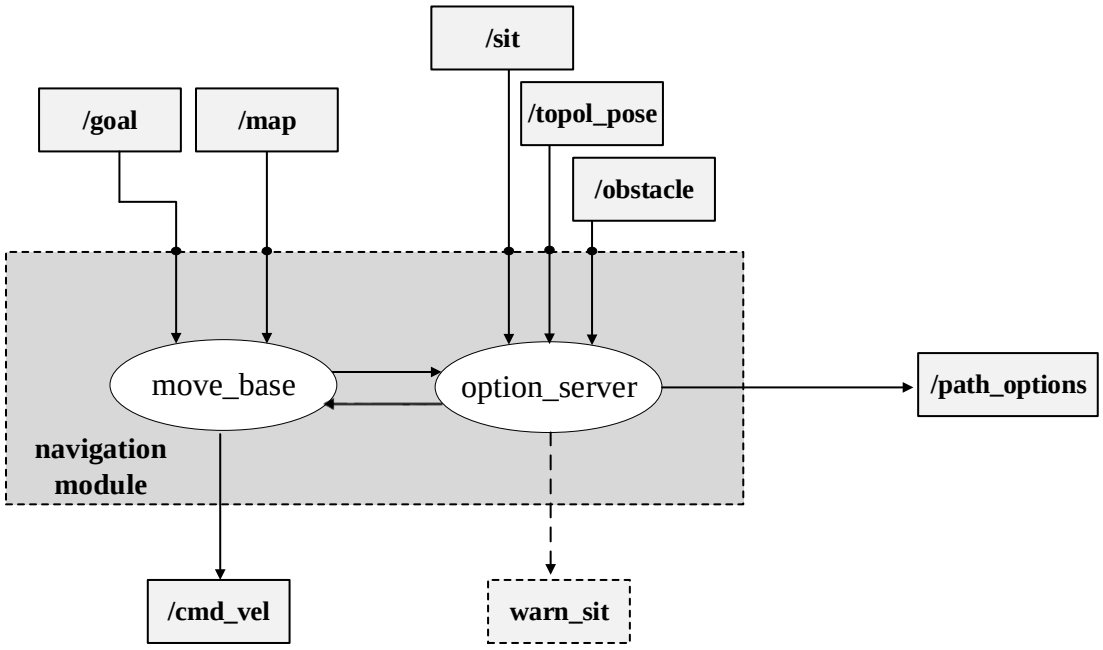


Figure 5.3: Navigation module nodes

5.2.1 *move_base* package

This package is one of the most used in ROS-based robots. It performs a task that is essential to any autonomous wheeled robot. It consists of a global planner that produces global trajectories between two points in the world map, and a local planner to follow that path in the most optimal manner.

The global planner uses a global costmap built from received occupancy grid map, by inflating occupied cells to the size of the robot. This costmap is built dynamically, every time the map is updated. Based on this costmap, the global planner performs a tree search to find the optimal path. This implementation uses Dijkstra method, already described in Chapter 2.

The local planner uses a local costmap which is built dynamically with raw data available from range sensors. The obstacles are also inflated by a value equal to the robot radius. It implements a dynamic window approach to collision avoidance, having the global path as reference.

This package is also able to provide a global path by request without causing move_base to execute it, enabling external nodes (and particular option_server) to preview paths. Figure 5.4 summarizes this package role in the system.

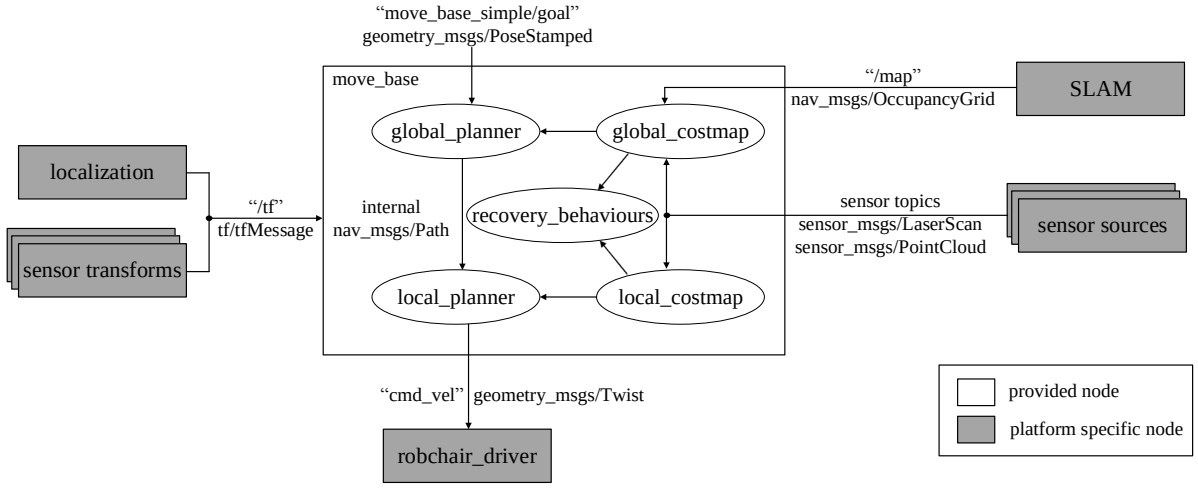


Figure 5.4: Representation of the move_base with RobChair as the robot (adapted from [Marder-Eppstein and Perko, 2014])

5.2.2 *options_server* node

This node's task is to provide options to the Collaborative Controller regarding robot motion. Three key UARS (see Chapter 3) are considered in the current implemented system:

1. Detection of an unpredicted obstacle

2. Fork reached

3. Deadlock

For the first case, unexpected obstacle detection, this node is prepared to map the obstacle in the costmap, inflating it, and assess if there is a way to the left and/or to the right of the obstacle. Each test is done by virtually closing the other path in the costmap. If both directions are viable, options are transmitted to the CC, so that it is communicated to the user. If only one passage is viable, no options are issued to the CC, and the path following is resumed. If no manageable passage is encountered, the situation is reevaluated as a deadlock, and *warn_sit* service called (see definition in Section 5.3).

In the deadlock situation, the user should be able to choose if he wants to return to a previous node, wait or call for help. No special option is assessed while in a deadlock, so the message sent to the CC only refers to the existence of the deadlock.

Finally, when arriving at a node which connects to other multiple nodes, the situation is detected by Perception module and transmitted to the Navigation module. With this information comes the related information about the situation, as is specified in the message protocol. In this case, the message carries information about the nodes linked with the current robot position (current node). This set of nodes is provided to the Collaborative Controller, which in turn becomes in charge of matching them with the user commands.

In the end of the processing, options are encapsulated in message form by a framing process (see Figure 5.5).

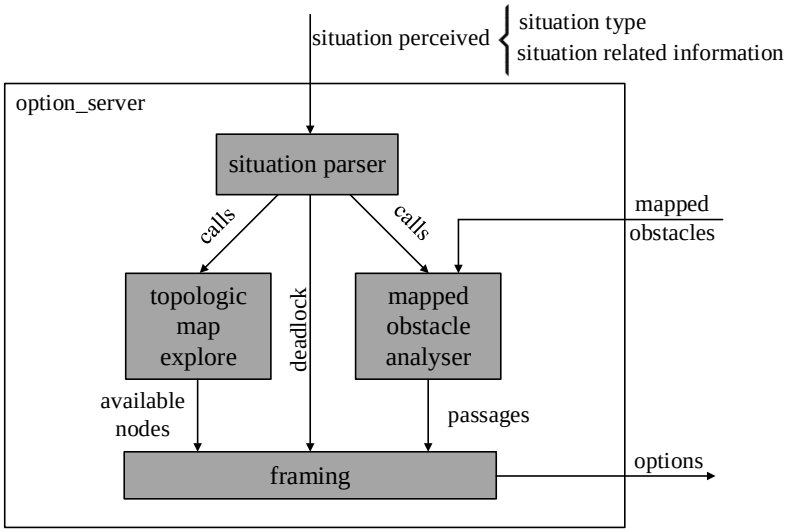


Figure 5.5: option_server design

5.3 Perception Module

The Perception module can be viewed as a system state estimator. It takes sensor data and, based on different algorithms and assumptions, it estimates some state variables of the system

and environment. As designed in Chapter 3, the current pose of the robot and updates of the map of the environment must be estimated; these estimates are given by a SLAM package. The SLAM package implemented is a modified version of the Hector Team SLAM's method, which was altered to receive an initial map of the static environment. This SLAM method is then responsible to update the gridmap and the estimate of the robot's pose. The resulting modified version was named isr_hector_mapping.

In order to detect UARS, there are also two modules to assist in such detection. One of them is the Topological State Observer, which simply matches the pose estimate given by SLAM and compares it with the graph map. It is also informed of the goal chosen by the CC to narrow its search for edges/nodes. It outputs one of two possible identifiers: a node where the robot currently stands, or the edge that connects the previously visited node and the goal node. Nodes are encapsulated in an object containing geometric information about its position, as well as attached objects containing information about other nodes linked to it. The other module of UARS detection is the Obstacle Detection method. It compares a local cut of both initial and current grid maps, by sub-sampling both maps around estimated position. From this comparison, a set of cells that are occupied in the current map but not in the initial map may be detected, thus providing an insight of a possible unexpected obstacle. The points detected are encapsulated in a polygon that is sent to the navigation module for further analysis.

Figure 5.6 represents the three essential parts integrating the perception module (on the left); Obstacle Detection task is emphasized on the right.

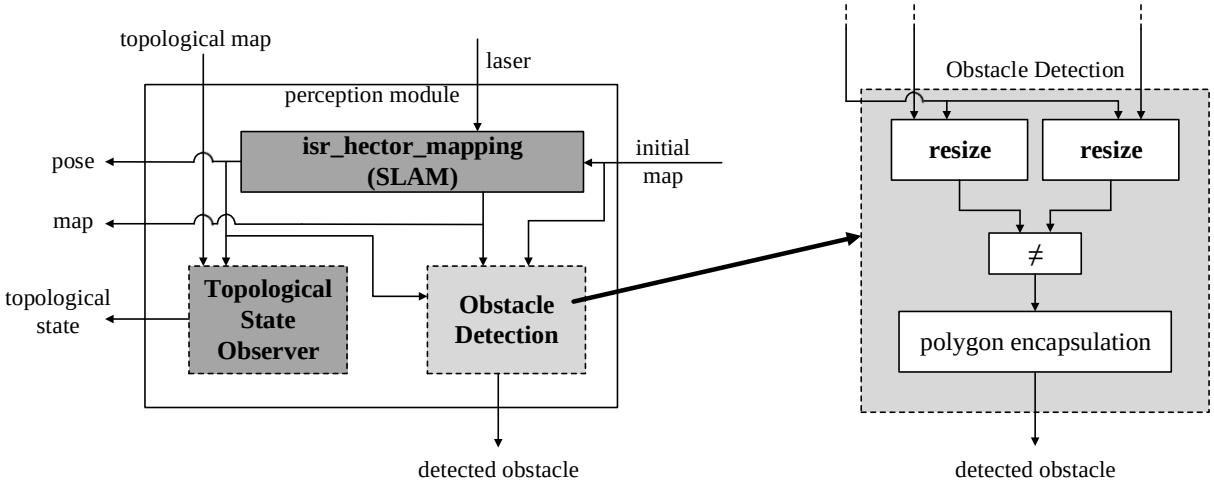


Figure 5.6: Perception module (left) and Obstacle Detection algorithm (right)

In terms of ROS implementation, there is also a virtual agent, a module that publishes to the environment the current situation. To do this, it establishes a service that UARS detectors can call to assert the detection of a certain situation. Using a module dedicated to UARS publishing simplifies the message network. Figure 5.7 represents the four ROS nodes implemented to do this tasks, highlighting message topics subscribed and published as well as available services.

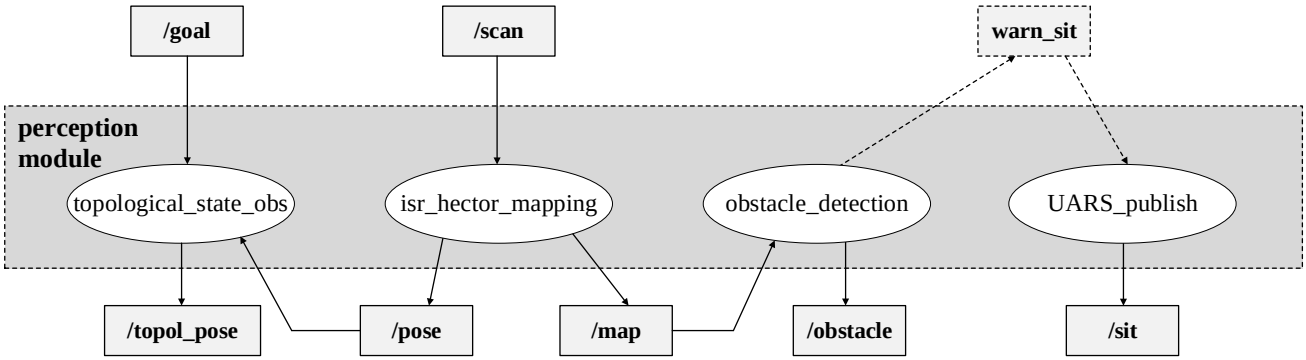


Figure 5.7: Nodes comprising the Perception Module

5.3.1 Topological State Observer

The topological state observer routine is required to update a topological state comprising the location of the robot according to the graph map. In topological terms, the robot can either be at a node or between two nodes, at an edge.

When the system starts, the robot starts at a given node and all modules are informed. When the decision of leaving a node is assumed by the CC, the observer is also informed, changing the state from a node to the given edge. Given a graph describing the network of the current environment and the pose estimate at a given time, this routine evaluates if the robot is near the goal node.

5.3.2 Obstacle Detection

This routine works by comparing two grid maps. Results are show in Figure 5.8. When updating the map, because laser range finders have some noise in their measure and because pose estimation is not spotless, the SLAM module may mark as occupied, cells that were close to other already occupied cells. To avoid considering this cells as part of detected obstacles, only predefined zones of the *a priori* map (defined by a set of geometric rules) must be considered in obstacle detection.

Figure 5.8: Result of the comparison between the two maps

5.4 Collaborative Controller module

The Collaborative Controller module is implemented as a ROS node, *collab_ control*. This node implements the design detailed in the last chapter. Figure 5.9 depicts the nodes published and subscribed topics.

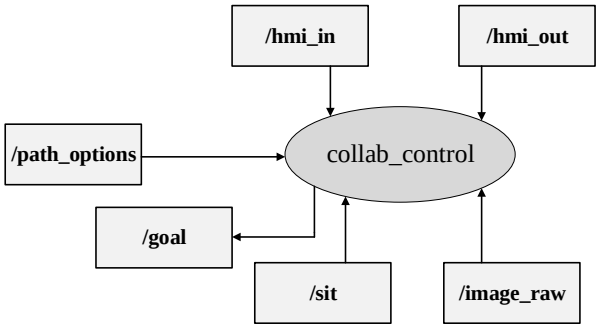


Figure 5.9: Topics associated with the *collab_ control* node

5.5 Simulation in Gazebo

Before testing the software layer with RobChair, its functionality was partially tested with gazebo simulator [Howard et al., 2014]. A model of RobChair was defined in Unified Robot Description Format (URDF); a 3D model of ISR ground floor hallways (named ISR0) was used and also models of a Laser Range Finder (LRF) sensor and kinect RGB-D camera were tested. With gazebo inner physics engine, and using both the RobChair and ISR0 models, the system was initially tested. Figure 5.10 shows an example of a simulation running.

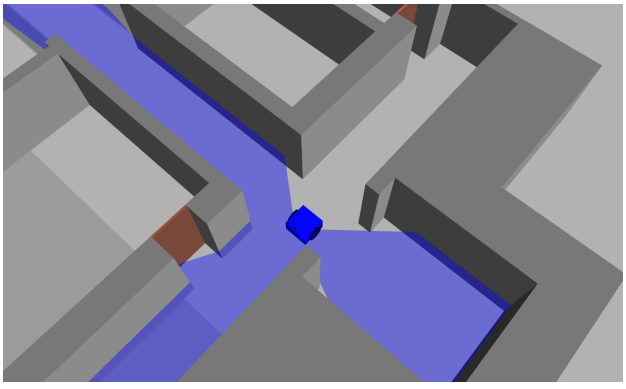


Figure 5.10: Simulation of RobChair in gazebo

5.6 Physical Layer

To allow the system to be tested in real scenarios, a physical layer is obviously needed. To be incorporated in the ANS, its components must operate at the hardware level and then there must be a viable link between hardware and software layers, known as drivers.

In the described implementation, there are three main components in the physical layer: RobChair as the robot; Microsoft Kinect as the image sensor (only RGB is used) an Hokuyo UTM-30LX for the LRF. As for the drivers, the robot driver was implemented in [Gonçalves, 2013] and is used here with a few changes, further described. The sensor drivers used in this work were obtained in the distributions available for the ROS community. Figure 5.11 shows the integration of the drivers in the ROS environment.

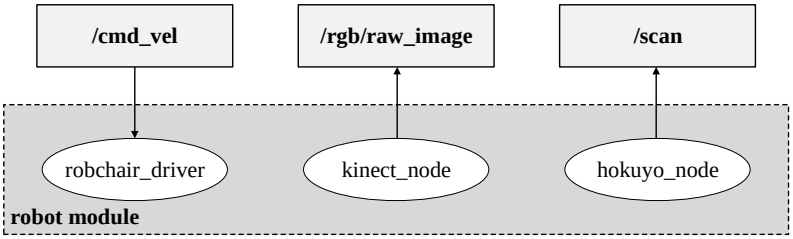


Figure 5.11: Drivers to the physical layer components

Some additions were also made to the low level hardware layer, to answer several issues. The issues to be considered were:

1. Lack of a protocol for electrical connections in RobChair;

2. Lack of physical support for sensors and interfaces;

3. Lack of a safety layer for prevention of possible hazards during testing.

To solve the first issue, an electrical connection box was created, and inputs and outputs were defined with a connector scheme for different voltages and signals. Figure 5.12 shows the created box. RobChair is a research robot and as such, its architecture is changed frequently, adding new sensors and new interfaces. Limiting electrical connections to this space simplifies the robot's architecture and facilitates changes.

The second issue relates with the lack of supports in RobChair to add interfaces, computers and sensors. To solve this, two trays were designed and added, as well as a specific support for the BCI amplifier. Figure 5.13 depicts these changes.

Lastly, the need for safety layers was discussed. To proceed to testing the ANS in real settings, with human participants, safe mechanisms are needed to halt the robot if anything does not function as planned. To solve this issue, an emergency button was added, electrically connected to the motor power drivers. Figure 5.14 shows the button.

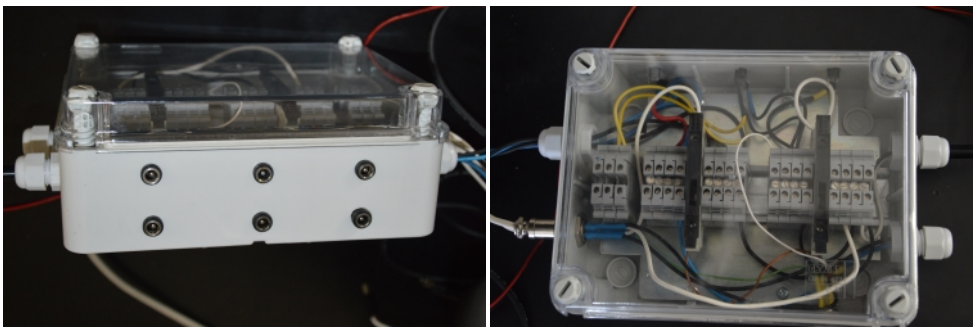


Figure 5.12: Electric connections box

Figure 5.13: Sensor and interfaces supports added

Also, to reassure the ability to safely halt the robot motion, a remote controller (RC) device was added to the layer. The microcomputer used to bridge ROS and the power driver was altered to accept a signal sent by the RC. This signal disables the microcomputer ability to send non-zero velocities to the power drivers.

Figure 5.14: Emergency button

Chapter 6

Experiments

In this chapter, experiments assessing key parts of the system performance are described. Various experiments with different focus were carried out to assess the current software and hardware implementation of RobChair ANS. There will be a section dedicated to each tackled issue.

6.1 SLAM

The perception module performs a fundamental role in the system. Without it, there is no localization, and without localization, navigation becomes compromised. Bearing this in mind, assessing the implemented SLAM module was one of the main focuses of testing. This section reviews experiments with the localization and map updating performance of the isr_hector_module.

As defined in previous chapters, the SLAM module was developed to accept a previous map. For these experiments, this map was created by running a lap with the team Hector SLAM around the ISR ground floor. Figure 6.1 shows the resultant probability grid, in log-odds representation.

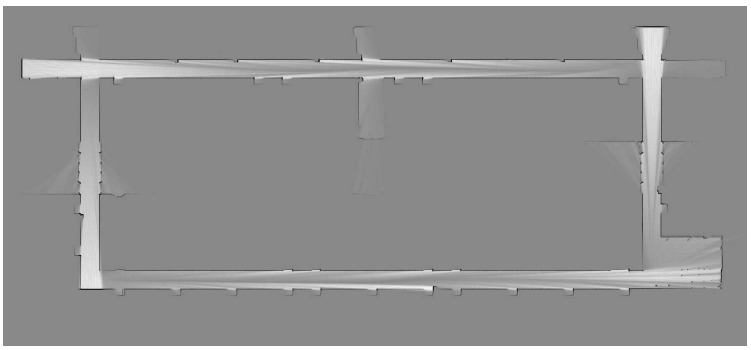


Figure 6.1: Probability gridmap with log-odds representation

This grid map representation is usually simplified in a ternary state map representation (with states Occupied, Free or Unknown) for visualization purposes. Figure 6.2 shows this simpler representation of the *a priori* map.

Having defined the initial map, two experiments concerning localization and map updating were carried out as follows. In the first experiment, a new room is mapped to demonstrate

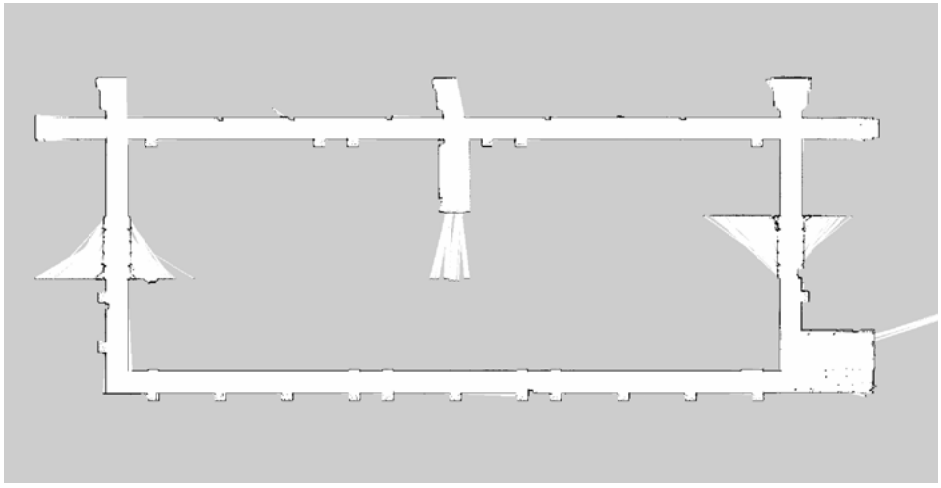


Figure 6.2: Ternary state map representation of the initial map

the ability to update the map. In the second experiment a large loop is traveled, revealing the module capability to maintain the map in long routes.

In the first experiment the robot is teleoperated along the hallways, finishing the experiment by entering a new unmapped room. Figure 6.3 shows the results of the experiment. The path taken is represented in a color continuum from green to red - green represents the begin of the experiment and red the end. Blue cells represent portions of the map updated during the test. Dark blue represent occupied cells and light blue empty ones.

Figure 6.3: Mapping a new room

In the second experiment, a clockwise lap is done around the hallways of ISR. As the initial map was done only with a simple counterclockwise lap in the same area, parts of the map remained unexplored since only one LRF is used. Figure 6.4 displays the results of the test. The same color scheme as the previous experiment is used. It is noticeable that the map remains well structured after returning to the starting point.

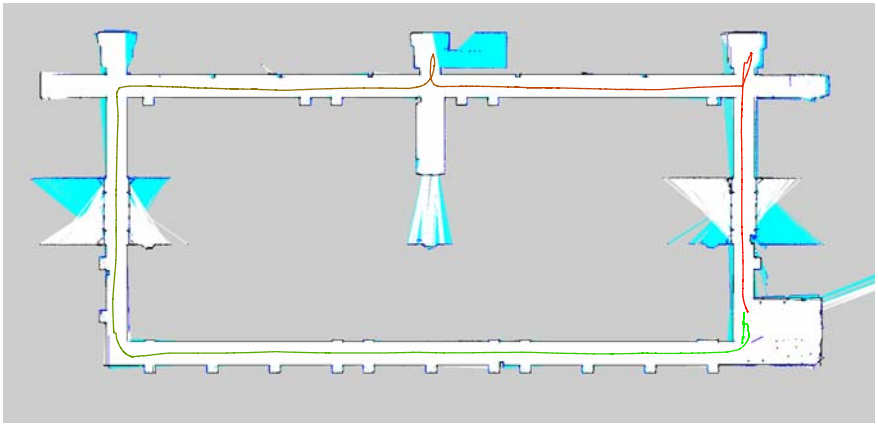
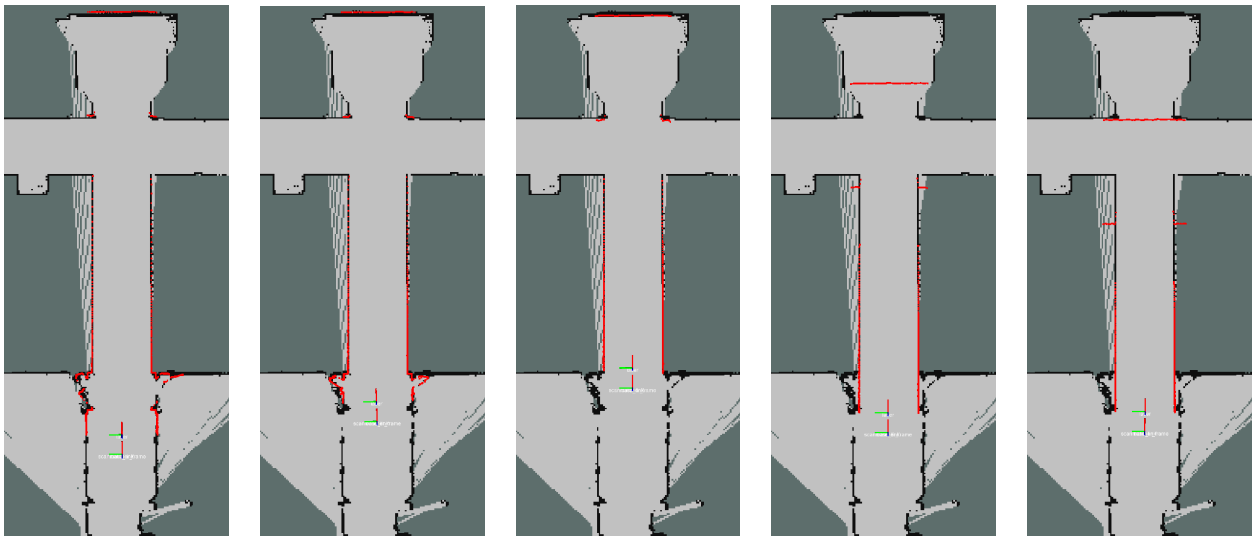


Figure 6.4: Updating a large loop

6.1.1 Detected Problems

The last two experiments lead to results qualitatively defined as successful. Nevertheless, other tests may show this SLAM implementation shortcomings. Structured environments may become a problem for the scan matcher, when physical features become scarce or non-existent, causing an aperture problem.

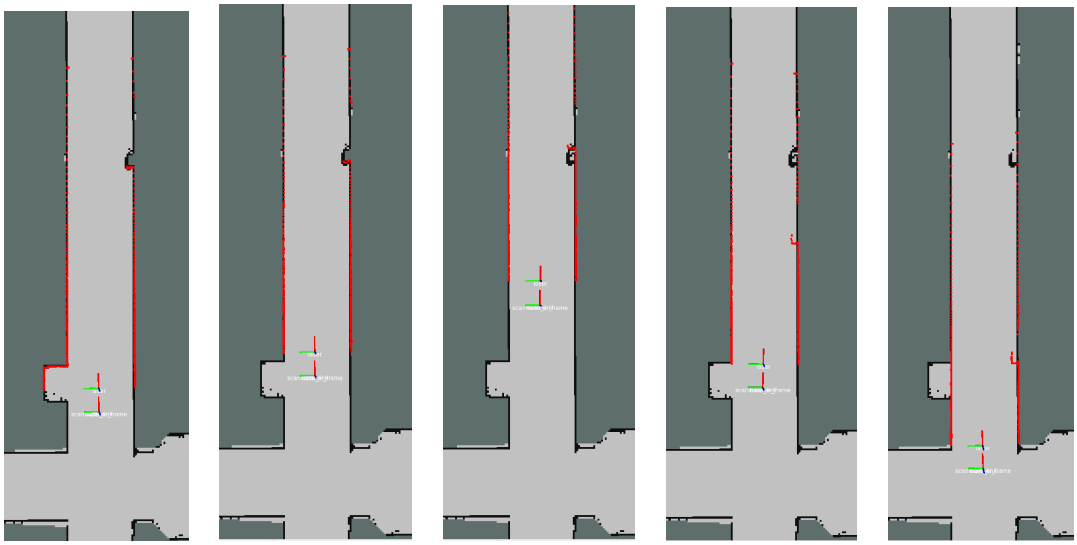
Figure 6.5 shows a sequence of pose and map updates where scan matching fails due to error gradient slipping to a local minimum. The robot is moving towards the intersection; robot estimated pose is represented by the reference axes and laser endpoints are marked in red. Frames 6.5-a) to c) show correct pose estimation. Between frame 6.5-c) and d), the scan matcher was trapped in a local minimum of the error function. High density of laser endpoints closer to the robot were matched with high occupancy probability cells, even though some sparse endpoints were left unmatched.



a) b) c) d) e)

Figure 6.5: Scan Matching failure

Figure 6.6 shows another sequence with the robot moving along a corridor, driven by the autonomous navigation module towards a goal point. This situation is specially hard because the environment is a featureless corridor, creating a low variability gradient along the corridor. In frames a) and b) a correct pose estimation is shown. From frame b) onwards, only the laser endpoints reflected by a garbage can (small rectangle on the NE corner) enable the scan matcher to detect motion along the corridor; without it, an aperture-like problem occurs. Frame c) shows a small error in pose estimation due to a local minimum found in the error function. Occupied cells in the beginning of the corridor have higher probability values because they have now been mapped twice. This results in a slope in the error function towards local minima like frames d) and e) depict.



a) b) c) d) e)

Figure 6.6: Corridor environment where scan matching failed

6.2 Autonomous navigation

In this section, simple experiments with the autonomous navigation are described.

In the first experiment, the robot is given a small graph map and is asked to navigate through some of its nodes. The robot moved at a slow and constant speed throughout the experiment. Figure 6.7 depicts the experiment. The color scheme is the same used in SLAM experiments; squares in magenta represent map points where the robot was order to drive to.

In a second experiment, the system was started in a corridor and the robot driven until a forking was detected. The user was then given three options, depicted in Figure 6.8. By choosing the RIGHT90 command, the robot continued its path, establishing the new goal node.

Another experiment was made to assess the system response to obstacle detection. In this experiment the user chose previously to be driven to a node. By detecting an obstacle, the

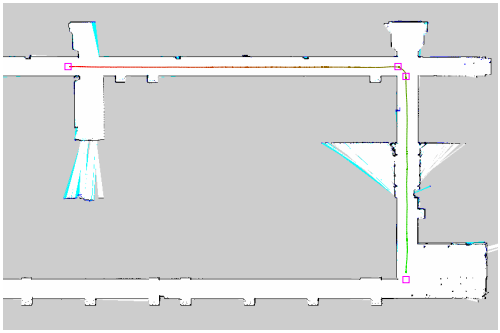


Figure 6.7: Small test concerning navigation performance

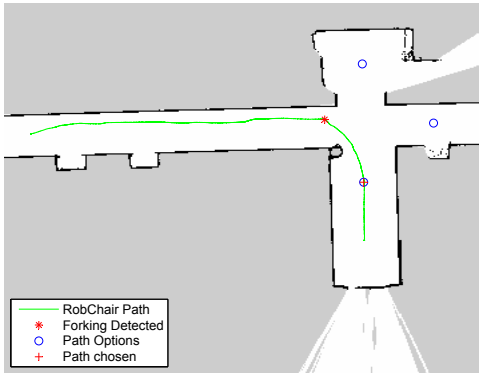


Figure 6.8: Test comprising a forking situation

CC requested user assistance in choosing which side to surpass the obstacle. By sending the LEFT90 command the user chose one path and the robot executes it, and continues to drive to the previously asserted node. Figure 6.9 depicts the experiment.

Problems with the autonomous navigation did occur, but were always related with localization, discussed above.

Results show that the ANS already has some level of autonomy. It is evident, however, that some relevant issues may present an obstacle to the good functioning of the ANS and so should be tackled first.

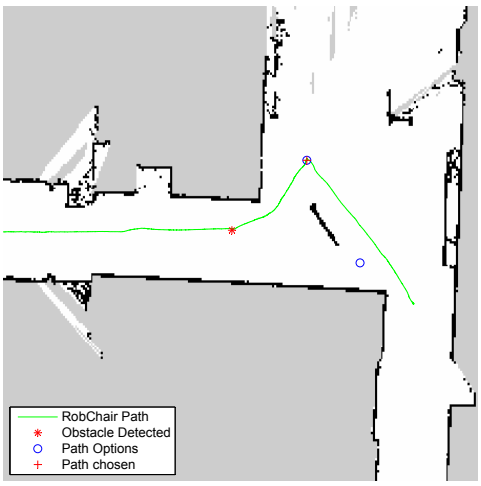


Figure 6.9: Test comprising a forking situation

Chapter 7

Conclusion and future work

7.1 Conclusion

This thesis presented a new design of the ANS for RobChair, focused in assisting patients with severely impaired mobility. A Collaborative Controller, Navigation and Perception modules were designed and developed.

The goal of providing a new take on Assistive Navigation System (ANS) was achieved, fully integrating it in ROS and with the software and hardware architecture used for RobChair. Initial testing provided promising results, leading the way to even more robust solutions. The present work adds valued improvements in the following matters:

- a new SLAM implementation for updating an initial map;
- development of the Collaborative Controller in ROS;
- configuration of the ROS Navigation stack in RobChair;
- simulation and 3D modeling of RobChair;
- updates in the physical layer.

The implementation of the CC and other algorithms in ROS was, above all things, of great value because with it great tools for testing and developing software, including real-time and scientific code like the ROS C++ and Boost libraries, became available. The new CC module implemented in this work is an important improvement to the software layer of the current ANS as well as an accessible platform for further development. Also noteworthy was the studying and analysis of a difficult SLAM implementation, Hector Mapping. It opens new possibilities in developing and understanding functional SLAM solutions.

7.2 Future work

In order to improve the current stage of development of this system several topics should be addressed in future works. An immediate task to be done is a full connection between the designed ANS and BCI. Adapting the current BCI with two-way communication features would also be beneficial.

The SLAM module may be subject to update, solving the situations where it currently fails. Also, a more robust and balanced approach to obstacle detection and analysis should be studied. Finally, to have a real functioning system, further testing should be done with patients, strongly validating the system.

Bibliography

[Antsaklis et al., 1991] Antsaklis, P. J., Passino, K. M., and Wang, S. (1991). An introduction to autonomous control systems. *IEEE Control Systems*, 11(4):5 – 13.

[Billings, 1996] Billings, C. (1996). *Aviation automation: the search for a human-centered approach*. Human factors in transportation. Lawrence Erlbaum Associates Publishers.

[Bloom et al., 2011] Bloom, D., Borsch-Supan, A., McGee, P., and Seike, A. (2011). *Population Aging: Facts, Challenges, and Responses*. The @WDA-HSG discussion paper series on demographic issues: World Demographic & Ageing Forum. WDA-Forum, University of St. Gallen.

[Bonarini et al., 2012] Bonarini, A., Ceriani, S., Fontana, G., and Matteucci, M. (2012). Introducing lurch: a shared autonomy robotic wheelchair with multimodal interfaces. *IROS 2012 Workshop on Progress, challenges and future perspectives in navigation and manipulation assistance for robotic wheelchairs*.

[Bourhis and Agostini, 1998] Bourhis, G. and Agostini, Y. (1998). The vahm robotized wheelchair: System architecture and human-machine interaction. *Journal of Intelligent and Robotic Systems*, 22(1):39–50.

[Breazeal et al., 2004] Breazeal, C., Gray, J., Hoffman, G., and Berlin, M. (2004). Social robots: beyond tools to partners. In *Robot and Human Interactive Communication, 2004. ROMAN 2004. 13th IEEE International Workshop on*, pages 551–556.

[Brock and Khatib, 1999] Brock, O. and Khatib, O. (1999). High-speed navigation using the global dynamic window approach. In *In IEEE Int. Conf. on Robotics and Automation*, pages 341–346.

[Bruemmer et al., 2003] Bruemmer, D. J., Marble, J. L., Dudenhoeffer, D. D., Anderson, M. O., and McKay, M. D. (2003). Mixed-initiative control for remote characterisation of hazardous environments. *Proceedings of the IEEE 36th Annual Hawaii International Conference on System Sciences*, pages 127–135.

[Card et al., 1980] Card, S. K., Moran, T. P., and Newell, A. (1980). The keystroke-level model for user performance time with interactive systems. *Commun. ACM*, 23(7):396–410. Available from: `http://doi.acm.org/10.1145/358886.358895`.

[Darmstadt, 2012] Darmstadt, T. H. (2012). Team hector darmstadt website. Available from: `http://www.gkmm.tu-darmstadt.de/rescue/`.

[de Jouvenel, 1989] de Jouvenel, H. (1989). *Europe's Ageing Population: Trends and Challenges to 2025*. Futures (London), 0016-3287. Butterworths.

[Durrant-Whyte and Bailey, 2006] Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping (slam): part i. *Robotics Automation Magazine, IEEE*, 13(2):99–110.

[Durrant-Whyte et al., 1996] Durrant-Whyte, H., Rye, D., and Nebot, E. (1996). Localisation of automatic guided vehicles. *The 7th International Symposium on Robotics Research (ISRR)*, pages 613–625.

[Escobedo et al., 2012] Escobedo, A., Rios-Martinez, J., Spalanzani, A., and Laugier, C. (2012). Context-based face control of a robotic wheelchair. *IROS 2012 Workshop on Progress, challenges and future perspectives in navigation and manipulation assistance for robotic wheelchairs*.

[Farwell and Donchin, 1988] Farwell, L. A. and Donchin, E. (1988). Talking off the top of your head: Toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and Clinical Neurophysiology*, 70:510–523.

[Fetz, 1969] Fetz, E. E. (1969). Operant conditioning of cortical unit activity. *Science*, 163:955–958.

[Fong et al., 2003] Fong, T., Thorpe, C., and Baur, C. (2003). Multi-robot remote driving with collaborative control. *Industrial Electronics, IEEE Transactions on*, 50(4):699–704.

[Fong, 2001] Fong, T. W. (2001). *Collaborative Control: A Robot-Centric Model for Vehicle Teleoperation*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

[Fox et al., 1997] Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics and Automation*, 4(1).

[Gerkey and Konolige, 2008] Gerkey, B. P. and Konolige, K. (2008). Planning and control in unstructured terrain. In *In Workshop on Path Planning on Costmaps, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

[Gonçalves, 2013] Gonçalves, D. (2013). Robchair 2.0: Simultaneous localization and mapping and hardware/software frameworks.

[Grasse et al., 2010] Grasse, R., Morere, Y., and Pruski, A. (2010). Assisted navigation for persons with reduced mobility: path recognition through particle filtering (condensation algorithm). *Journal of Intelligent and Robotic Systems*, (60):19–57.

[Grisetti et al., 2005] Grisetti, G., Stachniss, C., and Burgard, W. (2005). Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2443–2448.

[Grisetti et al., 2007] Grisetti, G., Stachniss, C., and Burgard, W. (2007). Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23:2007.

[Hirzinger, 1993] Hirzinger, G. (1993). Multisensory shared autonomy and tele-sensor programming - key issues in space robotics. pages 141–162.

[Hochberg et al., 2006] Hochberg, L. R., Serruya, M. D., and Friehs, G. M. (2006). Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature*, 442:164–171.

[Hoffman and Breazeal, 2004] Hoffman, G. and Breazeal, C. (2004). Robots that work in collaboration with people. *Proceedings of the CH12004 Extended Abstracts.*

[Hoffmann et al., 2011] Hoffmann, U., marc Vesin, J., and Ebrahimi, T. (2011). Recent advances in brain-computer interfaces.

[Hoffmann et al., 2008] Hoffmann, U., Vesin, J.-M. M., Ebrahimi, T., and Diserens, K. (2008). An efficient p300-based brain-computer interface for disabled subjects. *Journal of neuroscience methods*, 167(1):115–125.

[Hoppenot and Colle, 2002] Hoppenot, P. and Colle, E. (2002). Mobile robot command by man-machine co-operation - application to disabled and elderly people assistance. *Journal of Intelligent and Robotic Systems*, 34(3):235–252.

[Howard et al., 2014] Howard, A., Koenig, N., Hsu, J., et al. (2014). Gazebo simulator website. Available from: `http://gazebosim.org/`.

[Iturrate et al., 2009] Iturrate, I., Antelis, J., Kubler, A., and Minguez, J. (2009). A noninvasive brain-actuated wheelchair based on a p300 neurophysiological protocol and automated navigation. *IEEE Transactions on Robotics.*

[Kala et al., 2010] Kala, R., Shukla, A., and Tiwari, R. (2010). Fusion of probabilistic a* algorithm and fuzzy inference system for robotic path planning. *Artificial Intelligence Review*, 33(4):307–327.

[Kohlbrecher et al., 2012] Kohlbrecher, S., Meyer, J., Petersen, K., and Graber, T. (2012). Hector slam for robust mapping in usar environments. In *ROS RoboCup Rescue Summer School*, Graz, Austria.

[Kohlbrecher et al., 2011] Kohlbrecher, S., Stryk, O. V., Darmstadt, T. U., Meyer, J., and Klingauf, U. (2011). A flexible and scalable slam system with full 3d motion estimation. In *in International Symposium on Safety, Security, and Rescue Robotics. IEEE.*

[Krotkov et al., 1996] Krotkov, E., Simmons, R., Cozman, F., and Koenig, S. (1996). Safeguarded teleoperation for lunar rovers: From human factors to field trials. In *In Proc. IEEE Planetary Rover Technology and Systems Workshop.*

[Lavalle, 1998] Lavalle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical report.

[Lee, 1993] Lee, S. (1993). Intelligent sensing and control for advanced teleoperation. *Control Systems, IEEE*, 13(3):19–28.

[Lopes et al., 2012] Lopes, A., Pires, G., and Nunes, U. (2012). Robchair: Experiments evaluating brain-computer interface to steer a semi-autonomous wheelchair. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS12).*

[Lopes et al., 2011] Lopes, A., Pires, G., Vaz, L., and Nunes, U. (2011). Wheelchair navigation assisted by human-machine shared-control and a p300-based bci. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS11).*

[Lopes, 2012] Lopes, A. C. (2012). *Mobile Robot Assisted Navigation based on Collaborative Control.* PhD thesis, University of Coimbra.

[Lopes et al., 2013] Lopes, A. C., Pires, G., and Nunes, U. (2013). Assisted navigation for a brain-actuated intelligent wheelchair. *International Journal of Robotics and Autonomous Systems.*

[Lucas and Kanade, 1981] Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. pages 674–679.

[Marder-Eppstein et al., 2010] Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B., and Konolige, K. (2010). The office marathon: Robust navigation in an indoor office environment. In *International Conference on Robotics and Automation.*

[Marder-Eppstein and Perko, 2014] Marder-Eppstein, E. and Perko, E. (2014). Navigation stack description website. Available from: `http://wiki.ros.org/navigation`.

[Minguez and Montano, 2004] Minguez, J. and Montano, L. (2004). Nearness diagram (nd) navigation: collision avoidance in troublesome scenarios. *Robotics and Automation, IEEE Transactions on*, 20(1):45–59.

[Monferrer and Bonyuet, 2002] Monferrer, A. and Bonyuet, D. (2002). Cooperative robot teleoperation through virtual reality interfaces. In *Information Visualisation, 2002. Proceedings. Sixth International Conference on*, pages 243–248.

[Montesano et al., 2010] Montesano, L., Diaz, M., Bhaskar, S., and Minguez, J. (2010). Towards an intelligent wheelchair system for users with cerebral palsy. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 18(2):193–202.

[Nijboer et al., 2008] Nijboer, F., Sellers, E. W., Mellinger, J., Jordan, M. A., Matuz, T., Furdea, A., Halder, S., Mochty, U., Krusienski, D. J., Vaughan, T. M., Wolpaw, J. R., Birbaumer, N., and Kubler, A. (2008). A p300-based brain-computer interface for people with amyotrophic lateral sclerosis. *Clin Neurophysiol*, 119(8):1909–16.

[Nuchter, 2010] Nuchter, A. (2010). *3D Robotic Mapping: The Simultaneous Localization and Mapping Problem with Six Degrees of Freedom*. Springer-Verlag.

[Ong et al., 2005] Ong, K. W., Seet, G., and Sim, S. K. (2005). Sharing and trading in a human-robot system. *Cutting Edge Robotics*, pages 467–496.

[Papanikolopoulos and Khosla, 1992] Papanikolopoulos, N. and Khosla, P. (1992). Shared and traded telerobotic visual control. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, volume 1, pages 878–885.

[Park et al., 2012] Park, J., Johnson, C., and Kuipers, B. (2012). Robot navigation with model predictive equilibrium point control. *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*.

[Patel et al., 2012] Patel, M., Miro, J. V., and Dissanayake, G. (2012). Probabilistic activity models to support activities of daily living for wheelchair users. *IROS 2012 Workshop on Progress, challenges and future perspectives in navigation and manipulation assistance for robotic wheelchairs*.

[Pires et al., 2008] Pires, G., Castelo-Branco, M., and Nunes, U. (2008). Visual p300-based bci to steer a wheelchair: a bayesian approach. *Conf Proc IEEE Eng Med Biol Soc*, 2008:658–61.

[Pires and Nunes, 2002] Pires, G. and Nunes, U. (2002). A wheelchair steered through voice commands and assisted by a reactive fuzzy-logic controller. *Journal of Intelligent and Robotic Systems*, 34:301–314.

[Pires et al., 2011] Pires, G., Nunes, U., and Castelo-Branco, M. (2011). Statistical spatial filtering for a p300-based bci: tests in able-bodied, and patients with cerebral palsy and amyotrophic lateral sclerosis. *Journal of neuroscience methods*, 195(2):270–281.

[Pires et al., 2012] Pires, G., Nunes, U., and Castelo-Branco, M. (2012). Comparison of a row-column speller vs. a novel lateral single-character speller: Assessment of bci for severe motor disabled patients. *Clinical Neurophysiology*, 123(6):1168–1181.

[Pires, 2001] Pires, G. P. (2001). Navegação assistida de uma cadeira de rodas controlada por computador e com interface de voz.

[Pires, 2011a] Pires, G. P. (2011a). *Biosignal Classification for Human Interface with Devices and Surrounding Environment*. PhD thesis, University of Coimbra.

[Pires, 2011b] Pires, G. P. (2011b). *Protocolo de Comunicação*.

[Pradhan et al., 2006] Pradhan, S. K., Parhi, D. R., Panda, A. K., and Behera, R. K. (2006). Potential field method to navigate several mobile robots. *Applied Intelligence*, 25(3):321–333.

[Rios-Martinez et al., 2011] Rios-Martinez, J., Spalanzani, A., and Laugier, C. (2011). Understanding human interaction for probabilistic autonomous navigation using risk-rrt approach. *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

[Rogers et al., 1996] Rogers, E., Murphy, R. R., Stewart, A., and Warsi, N. (1996). Cooperative assistance for remote robot supervision. *Presence*, 5:224–240.

[Shih et al., 2012] Shih, J. J., Krusienski, D. J., and Wolpaw, J. R. (2012). Brain computer interface in medicine. volume 87, pages 268–279.

[Siciliano and Khatib, 2008] Siciliano, B. and Khatib, O., editors (2008). *Springer Handbook of Robotics*. Springer.

[Siegwart et al., 2011] Siegwart, R., Nourbakhsh, I., and Scaramuzza, D. (2011). *Introduction to Autonomous Mobile Robots*. Intelligent Robotics and Autonomous Agents. Mit Press.

[Simmons, 1996] Simmons, R. (1996). The curvature-velocity method for local obstacle avoidance. In *In Proc. of the IEEE International Conference on Robotics and Automation*, pages 3375–3382.

[Stentz, 1995] Stentz, A. (1995). The focussed d* algorithm for real-time replanning. In *In Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1652–1659.

[Thrun et al., 2005] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. Intelligent robotics and autonomous agents series. Mit Press.

[Vanhooydonck et al., 2010] Vanhooydonck, D., Demeester, E., Hantemann, A., Philips, J., Vanacker, G., Brussel, H. V., and Nuttin, M. (2010). Adaptable navigational assistance for intelligent wheelchairs by means of an implicit personalized user model. *Robotics and Autonomous Systems*, 58(8):963–977.

[Vidal, 1973] Vidal, J. J. (1973). Toward direct brain-computer communication. *Annual Review of Biophysics and Bioengineering*, 2:157–180.

[Wasson and Gunderson, 2001] Wasson, G. S. and Gunderson, J. P. (2001). Variable autonomy in a shared control pedestrian mobility aid for the elderly.

[Zeng et al., 2008] Zeng, Q., Teo, C. L., Rebsamen, B., and Burdet, E. (2008). A collaborative wheelchair system. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 16(2):161–170.