Alma Mater Studiorum - Universitá di Bologna

DOTTORATO DI RICERCA IN

INGEGNERIA BIOMEDICA, ELETTRICA E DEI SISTEMI

Ciclo XXX

Settore Concorsuale: 09/G1 - AUTOMATICA

Settore Scientifico Disciplinare: ING-INF/04 - AUTOMATICA

A GROUND ROBOT FOR SEARCH AND RESCUE IN HOSTILE ENVIRONMENT

Presentata da: Seyedmohsen Mirhassani

Coordinatore Dottorato **Supervisore**

Prof. Daniele Vigo **Prof. Lorenzo Marconi**

Esame finale anno 2018

Abstract

The recent sheer developments in the field of robotics has encouraged the researcher around the world to consider the robots assisting human in different aspects of life. In this context, search and rescue is a very interesting ambient where the capabilities offered by the robots can be used to not only augment the quality of service but also impose lower risk to the human members of the rescue team. To this purpose, project SHERPA[1] has been defined to investigate the values that an intelligent heterogeneous robotic team can add to a search and rescue mission.

The robotic team understudy includes flying robots such as fixed wing and quad copters for the purpose of patrolling and surveillance and a ground rover that is mainly considered to provide a mobile power replenishment service for the quadrotors. Indeed, the limited autonomy of the quadrotors is the main bottleneck of deploying them in a wide area applications like the mission understudy. In the project SHERPA, this issue has been addressed by introduction of ground rover.

Navigation of the ground rover on the unstructured outdoor environment defined by the SHERPA is of the main focuses of this thesis. Due to roughness of the terrain, there are a lot of issues on the way of a successful localization. Moreover, the planning has to be compatible with the robot and environment constraints to avoid imposing a risk of mechanical damage to the system. These issues will be introduced to the reader and proper solution will be presented.

To accomplish the battery exchange operation, the rover is equipped with two auxiliary devices namely "Sherpa box" and "Sherpa robotic arm". In this thesis, firstly, designs of the two devices are introduced to the reader in details. Secondly, their integration with the ground rover will be covered.

Finally two important benchmarks of the SHERPA project, namely "human leashing" and "battery exchange operation", will be addressed by means of the developed system. The human leashing will be investigated in two approaches; 1) Only using the rover and 2) Through a cooperation between the rover and the robotic arm. The battery exchange operation will be also investigated and the corresponding experimental results will be presented.

[1] `http://cordis.europa.eu/project/rcn/106964_en.html`

Dedication

Firstly, I dedicate this thesis to my family who supported me emotionally and financially to conclude this chapter of my life in Italy.

Moreover, I would like to dedicate this thesis to all members and founders of Italian language school of "Aprimondo" who help people like me with learning Italian completely free of charge. The result of their dedication is definitely one the best gifts I have received in my life.

Last but not least, I would like to dedicate this thesis to the great souls of "Khwaja Shams-ud-Din Muhammad Hafez-e Shirazi" *a.k.a* Hafez, "Abu Hamid bin Abu Bakr Ibrahim" *a.k.a* Attar and "Jalal ad-Din Muhammad Rumi" *a.k.a* Rumi who accompanied me in this journey with their precious spiritual poetry.

Declaration

I declare this document is the result of my three years of research activities as a PhD students in Department of Electrical electronic and Information engineering (DEI) of “Alma mater studiorum - Università di Bologna”.

Acknowledgements

I acknowledge support of my supervisor Professor Dr Lorenzo Marconi in my research in University of Bologna. I also acknowledge support of my supervisor during the abroad period Dr.Raffaella Carloni in my research in University of Twente.

Last but not least, I would like to thank my colleagues Dr Nicola Mimmo, Eng Eamon Baret, Eng Mark Reiling and Dr Michele Furci who helped me with problem definition and solving, technical aspects, and publications.

Contents

List of Figures

Chapter 1

Introduction

This thesis is written based on my involvment to a European project called "SHERPA"[1] [3,4]. SHERPA project investigates the adoption of a heterogeneous team of robots assisting the rescue teams in the Alps in the context of a search and rescue mission. In this context, the project considers different scenarios where the presence of robots can make a considerable contribution to the mission. Based on the scenarios, the project defines some benchmarks to be satisfied by the members of the robotic team while benchmarks includes a set of task that may involve either one agent or multiple robots.

The heterogeneous robotic team, also known as "SHERPA Animals", includes "Fixed Wing"[2], "Rmax"[3], "wasp"[4] and "Donkey Rover".[5] There are also two more robots namely "Sherpa Box" [5] and "Sherpa Arm" serving as auxiliary devices for the *Donkey Rover*. The robots are integrated by means of two main software of the project namely "Delegation Framework" and "Sherpa World Model". The heterogeneous robotic framework is deployed to help the "Busy Genius" referring to the human member of the rescue team who is supposedly most of the time busy doing crucial tasks concerning search and rescue. However, he is required to be available to trigger and partially supervise the mission [6].

This thesis is dedicated to the *Donkey rover*, a ground rover that is considered to support *wasp*s and human members of the rescue team. Therefore, here we seek to cover the research scope of navigation of the ground rover from localization to planning. Moreover, *Sherpa arm* and *Sherpa box* are presented. Lastly, some of the benchmarks of the SHERPA project concerning the rover are presented and investigated.

[1] Smart collaboration between Humans and ground-aErial Robots for imProving rescuing activities in Alpine environments

[2] A fixed-wing aircraft able to patrol large areas with low energy consumption, equipped with eagle-eye surveillance capability and suitable for normal weather condition.

[3] A rotary wing aircraft able to patrol large areas, equipped with eagle-eye surveillance capability and suitable for critical weather condition.

[4] A small scale quad-rotor with surveillance capability.

[5] A ground rover able to perform rough terrain navigation.

1.1 Motivation

Presence of the *wasp* is particularly useful in a search and rescue mission since the hardware provides a high manoeuvrability plus a fast and easy deployment. In the context of the SHERPA project, a *wasp* can easily reach a terrain that is inaccessible for the human. Being able to carry a camera, the rescuer may directly check the real-time video streaming provided by the *wasp*. Moreover, this data can be analysed by an image processing based software to locate the victims of an avalanche incident and tag them on a map that is being shared with the Busy genius. On the other hand, high energy consumption limits deploying this kind of drone in a search and rescue scenario. Indeed, a desired level of autonomy requires a heavy battery and consequently the autonomy of *wasp*s are limited to 20 minutes. This problem can be addressed by defining an autonomous power replenishment station for the *wasp*s.

1.1.1 Ground Rover

To service the *wasp*s in a dynamic mission with shifting areas of interest and, thus, to extend their operational radius, we chose a mobile ground base, as the strategical placement of stationary service stations across the operational environment [7,8] is not a practical solution for search and rescue or emergency response operations. In [9,10] recharging stations have been mounted on commercial mobile ground vehicles, but these platforms were intended for laboratory environments. Instead a tracked vehicle with robust outdoor navigation capabilities was developed for the SHERPA project. Indeed, support for the *wasp* is the main motivation of introducing the *Donkey Rover* in the SHERPA project.

The ground rover is the main focus of this thesis. Navigation on an unstructured outdoor environment presents different challenges to be addressed. For example, in some cases the ground rover is required to navigate rough terrains implying a risk of getting stuck and even flip over. Moreover, the localization of the robot in such an environment is quite a challenge. In the future chapters, hardware and software aspects of the Sherpa rover, the issues facing the system and our solutions will be presented.

1.1.2 Sherpa Box

The *wasp*'s energy supply can be replenished either by recharging or by exchanging the depleted batteries. Several design concepts for both recharging and exchanging the batteries are in [11].

Automated battery recharging stations are the most common solution due to their lower complexity, but also show a lower vehicle utilization due to the long recharging time [7, 9, 10]. On the other hand, a battery exchange station needs a mechanism that can extract and replace the batteries from the *wasp*, as well as a storage mechanism to hold the spare

batteries. Yet it implies a promising solution as it requires no additional time to charge the batteries before the *wasp* is operational again, which typically takes several times the flight time. The *wasp* needs to be designed such that the battery is easily exchangeable but still mounted well enough to withstand the vibrations experienced in flight and landing. Battery exchange stations, that combine rotating battery magazines with at least one linear actuator to move the battery containers from the *wasp* into the magazine and vice versa are presented in [12–14], while [15] uses linearly arranged battery bays.

As analyzed in [11], recharging stations are more economical for low coverage scenarios in terms of the provided coverage, *i.e.* , how many *wasp*s are operational at a given time, versus the total system cost. Nonetheless we opt for introducing the *Sherpa box*, a battery exchange system, to minimize the amount of *wasp*s and service stations to be transported to the mission areas.

1.1.3 Mobile Manipulator

The service stations presented in the literature require the *wasp* to precisely and safely land on a landing pad on the station. Most of the time, however, the landing precision of the *wasp* is not sufficient to engage the replenishment mechanism. This is especially the case for battery exchange mechanisms, where a close mechanical fit is required. To overcome the alignment issue some of the proposed stations have passive guidance systems that function like funnels [11, 13], while others are equipped with active alignment systems, like small arms or wire mechanisms [8, 14] that position and secure the *wasp*s during replenishment. However, many systems also still rely on external position sensing in order to land the *wasp* on the platform.

While these systems alleviate the problem of retrieving the *wasp*s, the landing operation is still delicate. The SHERPA mission requires that the robotic platform can reliably retrieve the *wasp*s without human involvement. Hence, we chose to introduce the *Sherpa Arm*, a robotic arm capable of robustly retrieving, docking, and deploying the *wasp*s, to be mounted on the ground rover. In order to increase robustness during the manipulation and docking of the *wasp*s, the arm is designed with a variable mechanical compliance.

1.1.4 Software

Complex missions involving a heterogeneous robotic team also necessitate a suited control and communications structure [16, 17]. As part of the SHERPA project, a framework for the automatic specification, generation, and execution of high-level collaborative mission plans has been presented in [18] which is a more robust and flexible solution than systems based on a central planning and scheduling algorithm [7, 19].

Moreover, the heterogeneous robotic team requires interacting with different aspects of the

environment. For example, terrain specifications is interesting for the rover while the weather condition information serves the flying robots. It is also necessary to have a platform where the agents can share the data and basically come up with a model representing the environment *i.e.* a "world model". In the context of SHERPA project, *Sherpa World Model* (SWM) is introduced to address this concern. Architecture of the SWM is presented in [20–22].

1.2 Sherpa System Overview

Complex collaborative tasks that would be difficult or impossible to perform for individual agents require the cooperation and coordination of the participating agents. This section presents the distributed communication and control architecture, that enables a larger heterogeneous human-robot team to effectively work together in a robust and versatile manner, even under adverse operating conditions.

1.2.1 Delegation Framework

The delegation framework and semantic structures developed for the SHERPA mission, as described in [18], are designed to enable a team of robotic agents to cooperate in planning and executing complex and hierarchical tasks in a dynamic mission environment. An overview of the delegation process and the agents and components involved in the battery exchange are shown in Figure 1.1.

A generic task or mission goal, such as the exchange of the *wasp*'s battery, is expressed in the form of a goal request Task Specific Tree (TST), *i.e.* a hierarchically structured description of the task. The internal nodes of a TST represent control statements for the task's execution, while leaf nodes represent domain specific functionalities. The transformation of a high-level request into a goal request TST takes place during the mission by dynamically instantiating generic TST templates.

The involved agents interact through their delegation modules, which contain a TST factory that can create TST nodes and link them to ancestors and descendants across agents, as well as a TST Executor Factory that provides platform specific functionalities for the execution of a task.

In the first phase of the delegation, a goal request TST is negotiated. When an agent requests a goal, the goal request TST is sent to its delegation module, where the distributed delegation process allocates each node of the tree to a suitable agent by means of an auction. The most suited bidder for a task is determined by solving the constraint problem belonging to a cost function, taking into account the mission requirements and platform capabilities. The result of a successful delegation is an expanded collaborative plan TST, where all nodes have been allocated to the participating agents. This TST can then be executed in the second

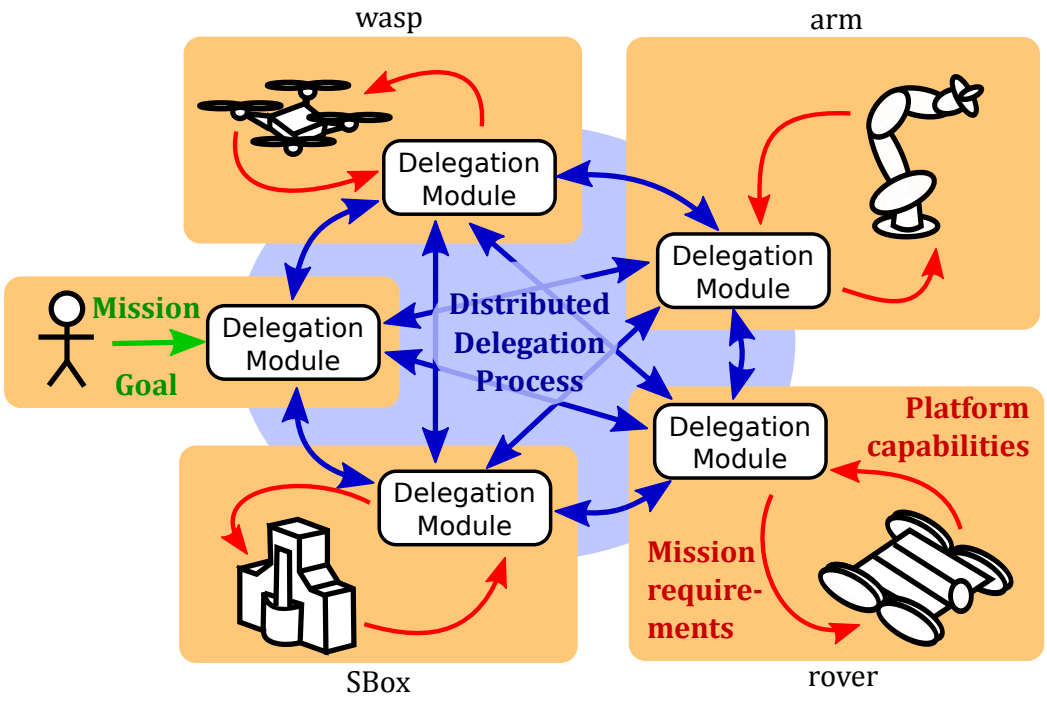


Figure 1.1: Overview of the distributed delegation process and of the agents for the autonomous battery exchange during the SHERPA mission. Each agent has a delegation module that mediates between mission requirements and platform capabilities. The human operator can request or approve mission goals, but the system is also able to request, plan and execute missions completely autonomously without human supervision in a distributed delegation process, in which tasks are assigned to the most suited agents.

phase of the delegation.

In order to have a clear idea of the implementation of the system, this section is dedicated to the SHERPA system overview. Therefore, we cover three important modules that are in the scope of this thesis; “Donkey rover”, “Sherpa Robotic arm” and “Sherpa box”.

1.2.2 Sherpa World Model

SWM endows the heterogeneous robotic team with a semantic world model. Moreover, each robot has its own SWM module through which contributes to the world modelling. The robots also sync their own status and pose information with the SWM.

The functionality of SWM can be enlightened by example of Figure1.2. The human operator triggers a search mission by specifying the search region on his map using human interface software. The rover uses the elevation data of the terrain that is acquired and processed by the fixed-wing to check if the region is traversable. As soon as the mission has been validated to be possible, the rover makes an inquiry to the SWM to check whether there is

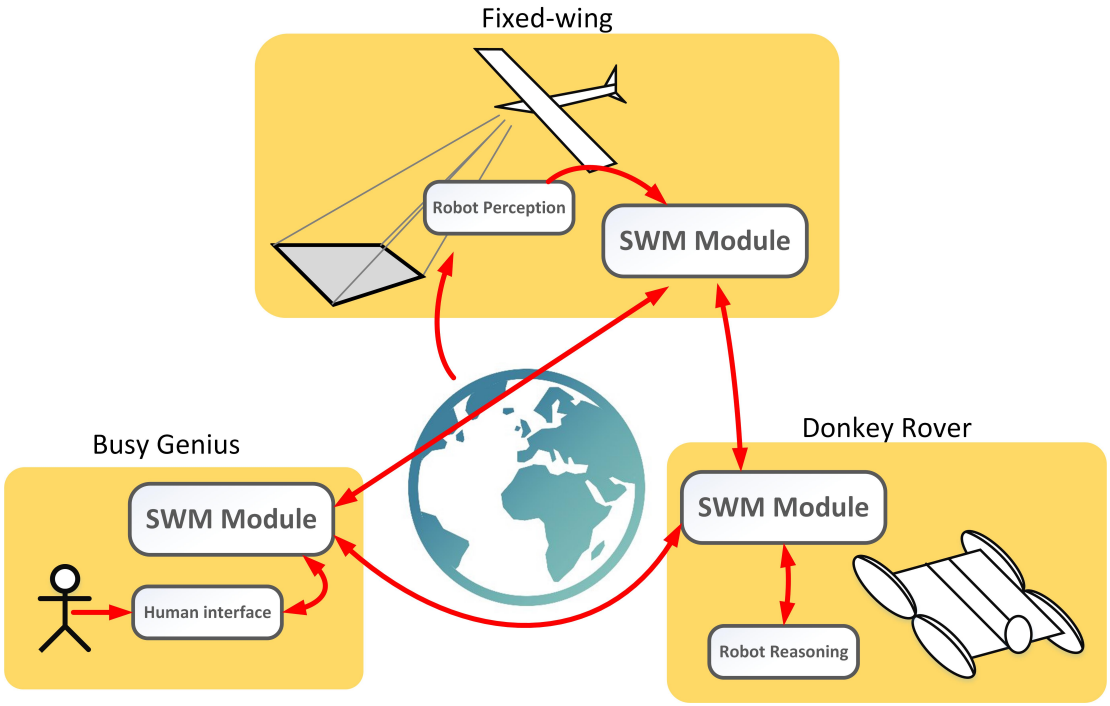


Figure 1.2: An example of functionality of SWM in a mission. A new search mission is designed by the user through the corresponding SWM module. The mission requires the rover to be sent to a new position. In the meanwhile map information of the mission site is updated by the *fixed-wing*. The rover makes an inquiry to the SWM for the map of the environment to be used in the navigation algorithm.

already an asserted path connecting the initial and target areas. Based on result of the inquiry, the rover may then follow an existing path in the system or plan a new path.

As can be seen in this example, the structure of SWM allows for storing information from different types and categories *i.e.* elevation map, paths, point cloud, robot pose, etc. Attending to more technical aspect of the SWM structure is out of the scope of this thesis. We, therefore, encourage the curious reader to refer to [21, 22].

1.2.3 Ground Rover

A specially developed ground rover[6] serves as mobile base for the *Sherpa box* and the *Sherpa arm*. It is characterized by its high degree of autonomy, endurance, and payload capacity. The rover is driven by four actuated tracks, that are mounted on a passively configurable chassis. These tracks allows the rover to traverse rough, mountainous terrain at a maximum speed of 0.8 m/s. Its range of sensors include a tilting laser scanner (LiDAR), IMU and GPS systems, as well as encoders for the tracks. The system is designed to run autonomously for three to

[6]Hardware is designed by Bluebotics, Switzerland (http://www.bluebotics.com/).

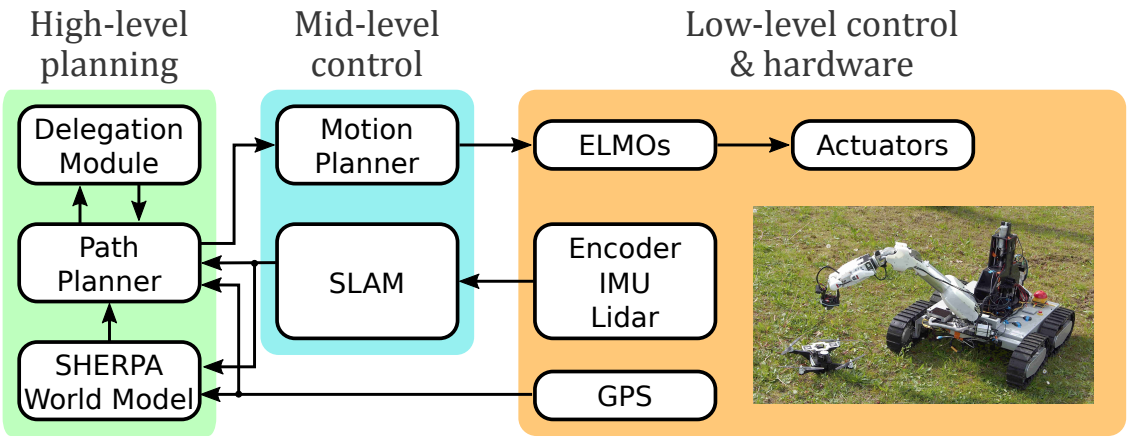


Figure 1.3: System architecture of the mobile base station (ground *rover*).

six hours, depending on the usage of actuators. Moreover, the rover's battery can be changed without shutting down the hardware. The rover is furthermore equipped with an embedded computer, Wi-Fi interface and internal power electronics.

Figure 1.3 shows the system architecture of the *rover*, which is divided into high-level planning, mid-level control, and low-level control and hardware. The high-level control interacts with the delegation framework, and formulates requested tasks as navigation problems that are solved by the path planner. A mid-level motion planner resolves the planned path and coordinates the individual motor drivers. On the same level, a SLAM[7] module combines data from the LiDAR, encoders, and IMU and provides a map of the environment with the rover's position to the high level path planner. The actual execution of the trajectory, and control of the actuators, is performed by ELMO Whistle digital servo drives[8].

1.2.4 Sherpa Arm

The ground *rover* is equipped with a robotic arm [23], in order to robustly deploy and recover the *wasps*, and thus to facilitate the autonomous servicing and battery exchange operation. Figure 1.4 shows the system architecture of the arm.

The arm has 7 DOF and a reach of one meter, and is designed for a payload of 2 kg. While being able to operate independently, it is considered to be mounted on the *rover* resulting in a mobile manipulator that is referred to as Ground Rover and Arm (GRA). To enable compliant and safe interaction with the *wasps* and *Sherpa box* during grasping and docking, the arm is equipped with two variable stiffness mechanisms in the shoulder and wrist joints. The arm's end-effector is a custom made gripper, that interlocks with an interface mounted on the *wasp*. As the arm is able to retrieve *wasps* without human assistance or requiring the *wasps*

[7]Simultaneous Localization and Mapping, the theory will be studied in section 2.3.
[8]Elmo Motion Control Ltd., Israel

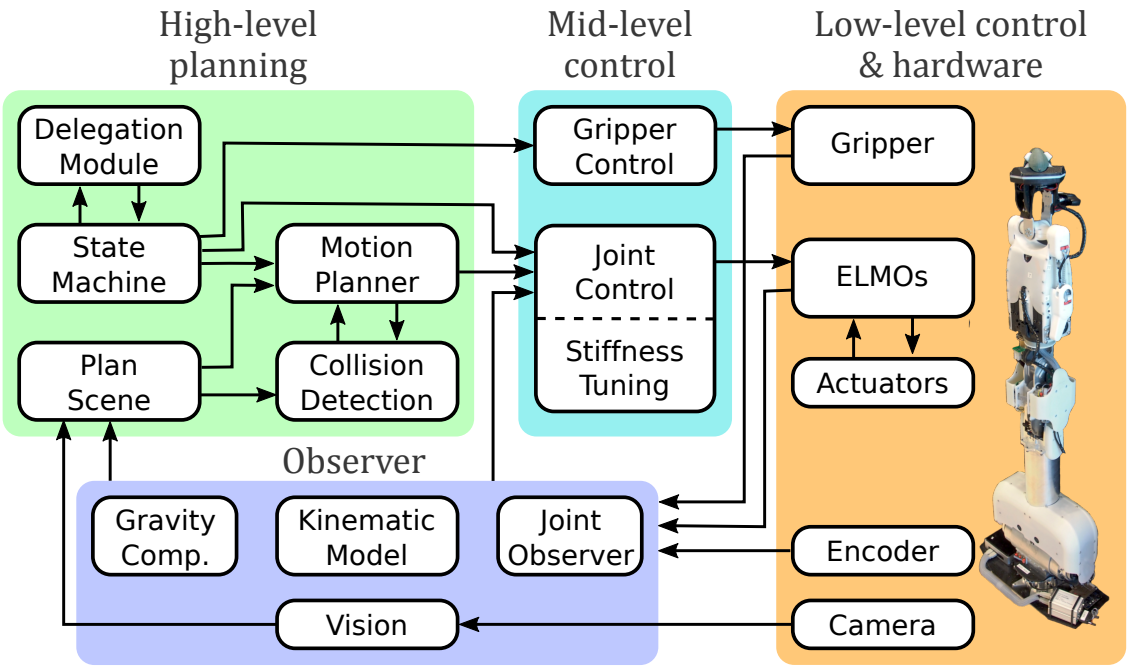


Figure 1.4: System architecture of the arm.

to perform precise and sensitive landing operations. Thus, It greatly extends the system's robustness and autonomy.

From the system hardware point of view, the high-level computational tasks are executed on an Intel NUC computer, while the low-level motor control is performed by ELMO Whistle digital servo drives. Moreover, mechanical design of the arm will be presented in section 3.2.

The arm's high-level planner includes a delegation module that interacts with the other agents, and a hierarchical finite-state machine (HFSM) based on the ROS decision making package, where the states of the HFSM are triggered by executors. When a task requires the arm to move to a certain Cartesian goal, a suitable joint space trajectory is generated using the rapidly-exploring random tree (RRT*) algorithm implemented in MoveIt! [24]. The mid-level joint controller then translates these trajectories to motor set-points and acts as supervisory controller.

The arm's observer combines data from the different sensors, including a camera[9] mounted on the gripper, to obtain the state estimates. The vision-based localization of the *wasp* is also carried out by the *arm* observer. Markers placed on the *wasp* interface are detected using an open source software[10]. After its detection, a virtual *wasp* is placed in the plan scene of the arm, and its pose is published to other agents, such as the *rover*. The virtual placement in the plan scene is necessary for motion planing of the arm taking into account the *wasp* that is attached to it.

[9]Logitec c920 webcam
[10]ROS AR TRACK ALVAR wrapper

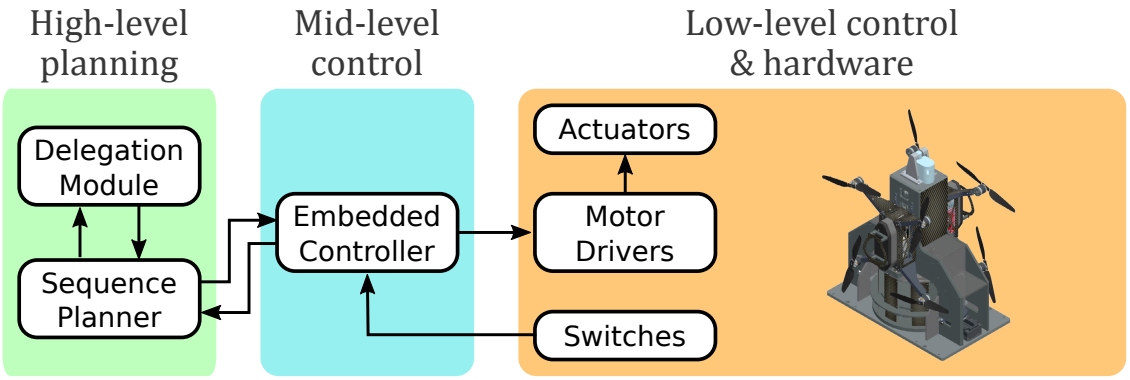


Figure 1.5: System architecture of the UAV service station and communications hub - the Sherpa box.

1.2.5 Sherpa Box

The *Sherpa box* is designed to function as the service station for docking and replenishing the *wasps*, as well as the computational and communications hub for the mission. However, its main mechanical functionality is in *wasp battery exchange benchmark* where a mechanism is required to latch the *wasp* and exchange the exhausted battery with a fresh one. Mechanical design of battery exchange mechanism of the *Sherpa box* is elaborated in section 3.3. The configuration of the *Sherpa box* leads to the advantage of leaving the arm potentially free during the battery replacement phases when the drone is locked on the docking surface. Thanks to this solution, the arm can look for, grasp and dock a second *wasp* thus speeding up the procedures when in presence of several flying vehicles.

As shown in Figure 1.5, the *Sherpa box* has a hierarchal control structure. A high-level sequence planner and delegation module, which run on an Intel NUC, coordinate with the delegation framework. They translate requests for the Arduino-based embedded controller that interfaces with the sensors and actuators of the *Sherpa box*.

1.3 Selected SHERPA Benchmarks and Thesis Outline

As mentioned earlier, the benchmarks are required tasks to be accomplished by the SHERPA agents. In this section we overview some of the benchmarks concerning the ground rover.

1.3.1 Way Point Navigation in Unstructured Outdoor Environment

This is the most basic operation that should be accomplished by the *rover*. The command is sent by the delegation framework through *Driveto* executor containing a target GPS coor-

dinates. This includes one of the main scopes of this research being navigation of a ground rover on a rough terrain. As mentioned earlier, roughness of the terrain implies a risk of damage and getting stuck. Therefore, a precise planning is desired that can be achieved only by coupling the planner with kinematics and dynamics of the system. Localization is an other topic of interest in the context of navigation of the Sherpa rover. The GPS localization provides a global coordinate system that is particularly useful in multi-robot projects. As a result the GPS system is adapted as the standard localization protocol in the SHERPA project. Yet the data has an accuracy of a few meters that does not suffice the desired navigation application. Consequently, the GPS localization has to be complimented by more precise localization strategy *e.g.* LiDAR and Vision based methods. In Chapter 2, we will provide an overview of the available localization and planning methods offered by the robotic literature to be used in the navigation of the ground rover.

1.3.2 Human Leashing

As called "Donkey", the rover is required to provide a human following capability so in practice the "Busy Genius" will be able to *leash* the *Donkey*. It is also desirable to endow the rover with the capability of being leashed not only to the rescuer but also to the *wasp*. The leashing command is sent through the *Leash* executor with a target that can be specified as a parameter. In this thesis we have considered two different cases of leashing. In the first case, the rover is assumed to be equipped with a fixed vision based sensor to recognize the human. While this strategy provides a superior precision comparing to GPS location of the human, the limited rage of operation of the sensor is a challenge to deal with. The objective is, therefore, designing a controller that guarantees the stability in the presence of the state and actuator constraints, *i.e.* the target stays in the operation field of the sensor and the actuator limits are respected. Since the sensor is fixed on the rover, performing the obstacle avoidance is difficult. This is because it implies two conflicting interests of following the target and deviating from the target to avoid an obstacle. Therefore, in the first case presented in section 5.1 we assume a safe navigation environment in which there is no risk implied by obstacles and roughness of the terrain.

In the second case, we take the advantage of having the robotic arm on the rover to provide a more realistic solution to the leashing problem. We therefore use the arm equipped with a similar vision based sensor to follow the target independently from the rover's heading. In the meanwhile, the rover approaches the target while considering obstacles and roughness of the terrain. To achieve a superior performance, we couple strategies taken by the arm and the rover through a cooperative framework based on the game theory. This solution is presented in section 5.2.

1.3.3 Battery Exchange Operation

As was motivated earlier, the *battery exchange operation* is to be provided by the *GRA* to increase the flying range of the *wasp*s. During the mission the rover is needed to approach the *wasp* and turn such that the drone ends up in the arm's workspace. Subsequently the arm picks up the drone and proceeds with docking it to the Sherpa box. Finally the Sherpa box performs the battery exchange mechanism and the wasp is ready to be deployed.

This is a complicated task of automation that needs to be done respecting the sequence of actions and verifying completion of each subtask. To this purpose, the SHERPA system has been introduced that provides a strong automation platform. The architecture of the SHERPA system has been elaborated earlier in this chapter (see section 1.2). To verify the system's performance, the system is deployed in an experiment that will be presented in section 5.3.

1.4 Contribution of this Thesis

The contribution of this research includes navigation of the Sherpa ground rover, integration of the Sherpa arm and Sherpa box with the ground rover and ultimately performing the benchmarks of the SHERPA project concerning the ground rover as listed in section 1.3.

The navigation of the ground rover is divided into subtopics of localization, mapping and planning. Although some of the blocks are adapted from the open source algorithms, our contribution is studying the pros and cons of these strategies and adapt what suits the best the application understudy. Chapter 2 presents an overview of the modern techniques of localization, mapping and planning that are available in the robotic literature.

Integration of the Sherpa robotic arm and Sherpa box with the ground rover is another scope of this research. In fact the ground rover, Sherpa box and Sherpa arm were developed by three different partners of the project, however they are desired to work as a single system. To this purpose, we firstly introduce the reader to the hardware and specification of these three systems in chapter 3.

Subsequently in chapter 4, we will study the integration of the Sherpa arm with the ground rover from the mathematical point of view. We, therefore, study the modelling of the rover and the robotic arm. We will then present and address the challenges associated with controlling these systems. Finally we will present modelling and controlling of the integrated system as a mobile manipulator.

Lastly chapter 5 focuses on the benchmarks of the SHERPA project concerning the ground rover. As explained in section 1.3, we will approach the problem of human leashing using two different strategies. The strategies will be motivated and elaborated and the corresponding result will be presented. Moreover, we present the battery exchange operation that verifies the performance of the SHERPA system. The corresponding results will be presented and discussed as well.

Chapter 2

An Overview of Problems and Tools For Autonomous Navigation

In order to make an autonomous mobile robot navigation possible, there are a lot of blocks that have to work in harmony with one other. Think about a human navigation, a human can go from **A** to **B** provided; i) *he knows where he is*, ii) *he can figure out how to get to the destination* and finally iii) *he does not get lost on the way*. Similarly a robot needs to; i) *know the environment*, ii) *come up with a feasible path to the goal*, iii) *figure out its position continuously*. In the robotic literature, these concepts are referred to as; i) "Mapping", ii) "Planning", iii) "Localization".

Before to start any movement, it is necessary for the robot to recognize the "Untraversable" and "Unpreferable" parts of the environment facilitated through the sensory information such as sonar, laser scans and vision. Classification of the environment raises an inevitable question of *Mapping*. For a robot, a map is a database which stores environment information acquired from the recognition stage. In the robotic literature, there are two categories of *Mapping*: "metric" and "Semantic". A *metric map* provides a discretization of the environment in which the information is presented in standard metric coordinates *e.g.* Cartesian [25–27]. The quantum of the environment discretization is referred to as "cell". A *semantic map*, on the other hand, presents a semantic graph in which objects (*e.g.* A house, a victim, a tree, *etc*) and their relative distances can be found [28–30].

Planning is referred to the process of coming up with a path which safely takes the robot from the departure point,**A**, to the destination **B** [31–33]. The choice of the planning algorithm highly depends on the navigation environment [34]. For example, for a robot navigating in an office environment the *Untraversable* cells describe fixed object in the environment while the *Unpreferable* cell may represent vicinity to an obstacle. In this context, a cell that is neither *Untraversable* nor *Unpreferable* is "free". It is important to note that the concept of free cell comes from the assumption that the plain of navigation is flat and free cells do not imply different risk or energy consumption over one another. On the contrary, in a rough

terrain, the *free cell* becomes a vague concept since each cell might have a different roughness and traversabilty is a function of robot capabilities. In such an environment, the cost of travelling does not only depend on the length of path and a longer but less difficult path might be more desirable.

Localization is accomplished by means of sensory information that can be categorized as; i) "Motion sensory information", ii) "Global localization Information", and iii) "Local localization Information". Describing motion of the robot, *Motion sensory information* can be integrated to guess the the current state of the robot based on a previous known state. In robotic literature, this process is referred to as "Odometry" [35]. Accelerometer measurements and wheel encoder information are two examples of this kind. The Global information, instead, refers to information describing robot's state with respect to a known global frame. Some examples of this category are GPS information, barometer and magnetometer measurements. The last category, on the other hand, comes from specific environment features *e.g.* visual features. This kind of information usually needs to be elaborated in considerably computationally heavy algorithms. Finally, *Localization* is estimating the state of the robot in some known coordinates using the aforementioned ingredients.

This chapter provides a general overview of the tools that realize the concept of navigation. For the Sherpa ground rover, the majority of the navigation blocks that we will study in this section are adapted from open source algorithms. The contribution of this chapter is then studying the available algorithms and their pros and cons and applying them to the hardware. Later, in implementation of the SHERPA benchmarks concerning the rover (see section 1.3), the navigation blocks will be running on the background to complete the developed algorithms by closing the control loops. The remainder of this chapter is organized as follows; Starting with attitude estimation, we describe the process of calculating 6 DOF[1] robot pose. We then proceed with Odometry. The concept of closed loop Odometry or bundle adjustment will then be explained. This brings us to Simultaneous Localization And Mapping (SLAM). After that, we will cover the GPS and concept of filtering. Subsequently, different techniques of mapping will be presented and, finally, the path planning of the ground rover will be covered.

2.1 Attitude estimation

The main objective of the attitude estimation is finding a 3D representation describing 6 DOF pose of the robot in a fixed frame. This can be performed using at least a fixed inertial frame and a body frame attached to the robot. There are different conventions for the inertial and body frame in the corresponding literature. Here for the inertial frame we follow the North East Down (NED) convention that considers a Cartesian frame with its origin somewhere on the surface of the earth, its x pointing toward magnetic North, y toward east and z pointing to earth's center. The body frame, on the other hand, is assumed to be a Cartesian frame originated on robot's center of gravity having x axes coming out from robot front side, y

[1] Degree of Freedom

toward right side of the robot and z pointing upward. The attitude estimation is then the task of finding the pose of the body frame, originated on the robot's center of gravity, in the inertial frame. Based on [36], we briefly describe the *static case* in which robot's acceleration compared to gravity's is small that is the case for the *Donkey Rover*.

Static attitude estimation can be accomplished using accelerometer and magnetometer data. Accelerometer gives a set of acceleration measurement $[a_x\, a_y\, a_z]^\intercal$ while the measurements represent the gravity force a_g as acceleration of the robots system is assumed to be neglectable. The first angle to be calculated is the roll angle, ϕ. This can be estimated by rolling the system about the x axis until the second element of the intermediate vector $[x_1\, y_1\, z_1]^\intercal$ is zero:

$$\phi = atan2(\frac{-a_y}{a_z}), \quad \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \tag{2.1}$$

Similar approach can be used to obtain pitch, θ, where the intermediate vector of $[x_1\, y_1\, z_1]^\intercal$ is used to pitch about y until a_x is zero:

$$\theta = atan2(\frac{x_1}{z_1}), \quad \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \tag{2.2}$$

After these two rotations, the body frame is parallel to the inertial frame and there is only one more rotation about the z axis needed to have body and inertial frames aligned. This can be done using the transformed magnetometer vector $[m_x\, m_y\, m_z]^\intercal$. It is obvious if the robot points to the magnetic north, the levelled magnetometer will read zero on the y axis. We, therefore, get:

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix}, \quad \psi = atan2(\frac{y_2}{y_1}) \tag{2.3}$$

Therefore the three roll, pitch and yaw angles are obtained. Moreover the 3D rotation matrix to convert body frame coordinates to inertial frame coordinates, NED, can be described as follows:

$$\begin{bmatrix} N \\ E \\ D \end{bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} \tag{2.4}$$

2.2 Odometry

In simple words, Odometry is counting your steps. In robotics, Odometry can be done using the motion sensors or environment features. In either of the cases, the measurement is in

the body frame and the rotation matrix of (2.4) can be used to calculate the trajectory in the inertial frame:

$$v^{IF} = R_{IF}^{Body} v^{Body} \tag{2.5}$$

where R_{IF}^{Body} is the 3D rotation matrix obtained from attitude estimation and v^{Body} is the measured speed in the body frame. It is obvious that for a nonholonomic robot. *e.g. Donkey Rover*, y and z elements of the v^{Body} vector are always zero. Furthermore, the rotation matrix R_{IF}^{Body} can be updated based on rotational velocity vector ω calculated from the Odometry:

$$R_{IF}^{Body\ k+1} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} R_{IF}^{Body\ k} = S(\omega) R_{IF}^{Body\ k} \tag{2.6}$$

The following section briefly studies different approaches to Odometry in robotics.

2.2.1 Dead reckoning

"Dead rocking" is, basically, integrating speed to find the position while the speed measurement is based on on-board motion sensors *e.g.* wheel encoders and IMU[2]. Consider the differential robot of Figure 2.1a, the linear and angular speed of the robot can be calculated from the left and right wheel speeds, V_{Left} and V_{Right} as follows:

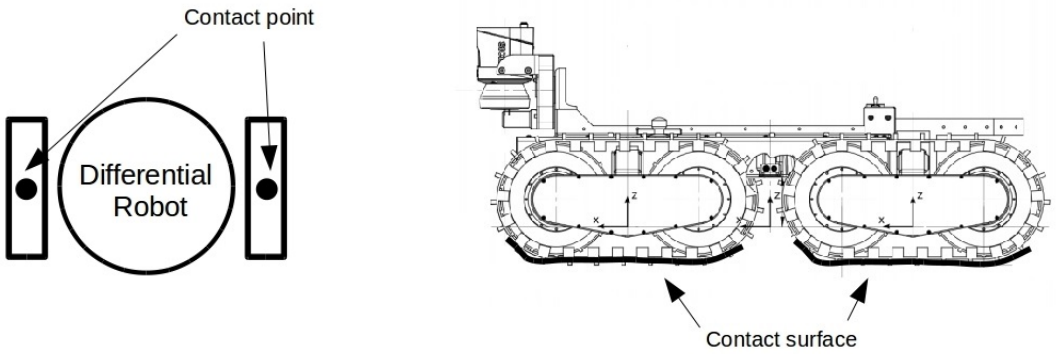
$$\begin{cases} V = \frac{V_{Left}+V_{Right}}{2} \\ \omega = \pm\frac{(V_{Left}-V_{Right})}{R} \end{cases} \tag{2.7}$$

Where V and ω are linear and angular speed of the robot, R is distance between right and left wheel, and for the angular $+$ or $-$ sign has to be chosen based on coordination system. The main drawback of *dead reckoning* is considerable amount of accumulative errors causing the estimated position trajectory diverging from the real position trajectory. The errors is originated from different sources:

1. It is hard to estimate the wheel contact surface with the terrain (Figure 2.1b).
2. Usually the robots are skid driven rather than differential, this causes a considerable drift in on spot rotation which is also known as "wheel slip".
3. Measurement errors due to limited accuracy.

To meliorate the rotational speed estimation, gyroscope measurements from the IMU can be used instead of (2.7). Moreover, encoder and IMU measurements can be filtered to achieve a better estimation [37, 38].

[2]Inertial Measurement Unit



(a) differential robot scheme.

(b) Donkey Rover.

Figure 2.1: a) An ideal differential robot with only two point of contact to the terrain. b) Sherpa rover, which is an example of a skid driven robot. As can be seen, it is hard to calculate the track contact surface.

2.2.2 Laser Odometry

Laser scanner, also known as LiDAR[3], provides a set of ranges with associated angles that can be used to estimate the movement of the sensor based on two consecutive sets of measurement. This problem is well investigated by the literature and one of the most popular approaches to it is Iterative Closes Point (ICP) algorithm. ICP seeks to estimate the sensor movement by minimizing point to point distances of the point clouds derived from the LiDAR measuerments [39–41].

Consider the situation depicted in Figure (2.2) where there are two consecutive set of data. Let us define the most recent data as a set of points $\{p_i\}$ while the older data is considered to be the reference surface S^{ref}. The goal of ICP is to find a transformation $\hat{T}$ to minimize the accumulated distance between transformed point Tp_i and its projection on the reference surface:

$$\hat{T} = \arg\min_{T} \sum_{i} \left\| Tp_i - \prod(S^{ref}, Tp_i) \right\| \tag{2.8}$$

Where operation $\prod(S, .)$ is the Euclidean projection on the surface S. Note that, for a 2D laser scanner, the transformation that is being acquired from such an algorithm is obviously a 2D transformation *i.e.* two linear velocities and one rotational velocity. Therefore, the robot must be navigating on a flat surface *e.g.* office environment. However, information from the attitude estimation might be used to extend this approach a rough terrain navigation.

[3]Light Detection And Ranging

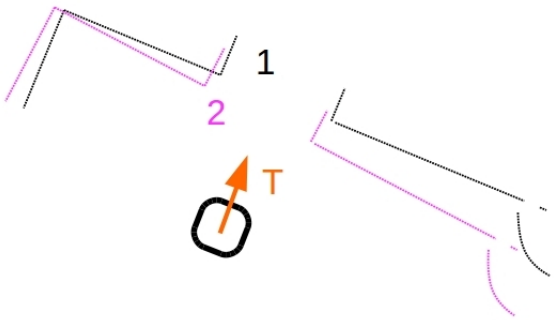


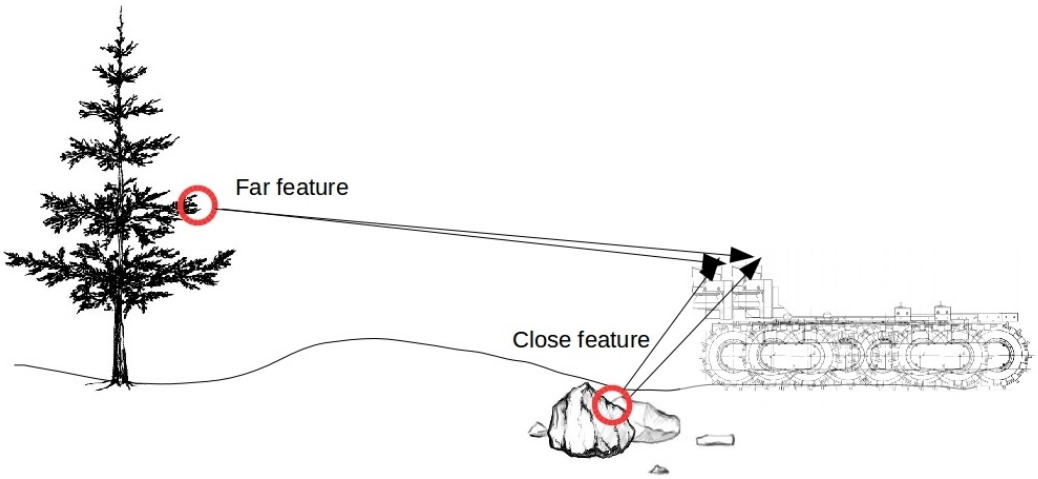
Figure 2.2: Laser odometry, ICP algorithm finds the transformation T that minimizes point to point distance between measurement sets 1 and 2.

2.2.3 Visual Odometry

Visual Odometry (VO) is a method of calculating the velocity of the robot using by means of visual features acquired by a camera. Visual features are distinguished parts of the picture that can be detected and tracked in consecutive frames. The camera motion trajectory can then be reconstructed from movement of the feature in consecutive pictures. The VO suggests an interesting alternative to Laser Odometry since the sensor is relatively cheap and light comparing to LiDAR and yet it provides a 6 DOF velocity estimation. On the other hand, the performance highly depends on the light condition and existence of visual features in the environment. However, the problems can be overcome by taking the right measures. For example, the camera angle with the horizon has to be tuned to receive the maximum amount of features of the environment. It is important to note that the further is the feature, the less information it can reveal about the sensor motion. Therefore, as it is illustrated in Figure 2.3a), distant features are less interesting for the VO application. For a ground rover, the desirable features can mostly be found on the terrain. Therefore, the type of terrain on which the robot is navigating determines how much VO is reliable. This is demonstrated in Figure 2.3b in which one can see how the quantity of features can be different from one terrain to another.

In VO, the motion estimated in the images has to be mapped into a real world motion. To this cause, there are four important coordinates to deal with namely; “world”, “camera”, “image” and “pixel” coordinates. The *world* frame refers to an inertial frame. The *camera* frame is originated on the focal distance of the camera while the *image* frame is originated on the picture. Finally there is the *Pixel* frame that unlink the others is not a metric coordinate but a matrix containing the information associated with pixels. These coordinates are presented in Figure 2.4.

The raw camera data are available in the *Pixel* coordinates. Therefore, the change of coordinates is performed through intrinsic and lens distortion parameters which are sensor specific (*i.e.* different from one camera to another). Camera intrinsic parameters are aqcuired



(a) Far versus close features.

(b) Terrain visual features.

Figure 2.3: a) As seen distant features need a huge sensor movement to move in the Pixel coordinates and therefore they are less informative b) An example of a terrain with a lot of features on the left versus a terrain with few features on the right.

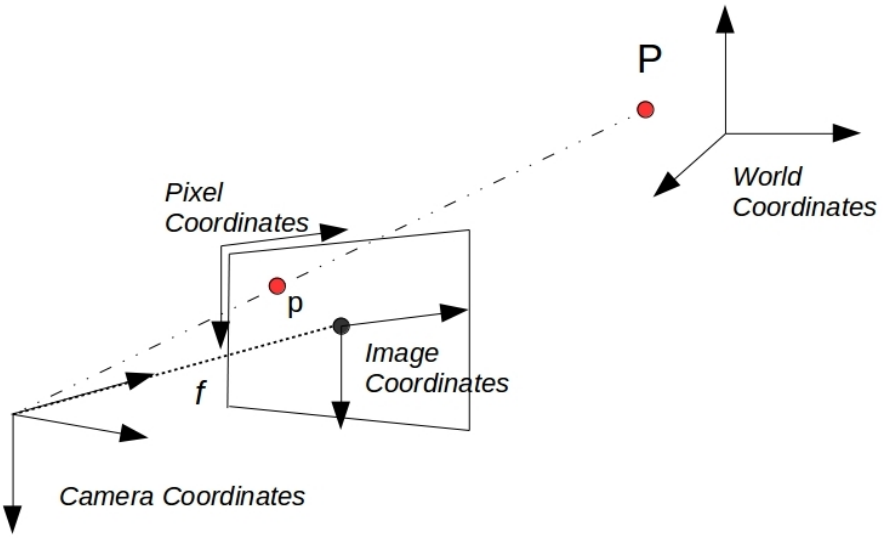


Figure 2.4: Different camera coordinates

through "Camera Calibration" that can be performed via available tools in MATLAB and OpenCV [42, 43].

After finding the intrinsic parameters, we can perform the change of coordinates up to the *camera* coordinates. Information about the *world* coordinates, however, can only be completely achieved using some extrinsic data. In other words, to retrieve the translation from *camera* frame to *world* frame, there must be some information more information rather than only the picture. The extrinsic information might be provided by for example markers with known dimensions, another camera with a known distance between the two camera and inertial sensor information.

In case there is no extrinsic available, the calculated translation is scaled by a fixed unknown factor known as "scale ambiguity". Figure 2.5 shows an example of scale ambiguous picture. In this example, the extrinsic data that helps us to understand the bottle is closer to the camera is our perception of bottle's dimension and size of human.

To address the scale ambiguity, [44] has incorporated the IMU data in VO. By means of the accelerometer and gyroscope data an estimation of robot movement is perfomed. This information is then fused into the VO deploying two different methods of spline fitting and extended Kalman filter. Another way to approach this problem is using a second camera which forms the stereo camera configuration. This method is well described in [2], some successful implementation of stereo VO can be found in [45, 46].

Figure 2.5: An example of scale ambiguity explaining the necessity of extrinsic information to retrieve translation information. The picture can suggest a giant bottle or a miniature human. However based on everyday life experience, we realize that the bottle is not giant but simply closer to the camera. So in this case, our everyday life experience is an extrinsic information that helps our perception.

2.2.3.1 Feature detection

Feature detection is an important prerequisite to a sucessfull VO implementation. In the computer vision literature, feature detectors are classified into two categories of "corner detectors" and "blob detectors". Forstner [47], Harris [48], and FAST [49] are some examples of corner detectors while SIFT [50], SURF [51] and CENSURE [52] are some well known blob detectors. Although, details of feature detection is out of the scope of this thesis, it is important to choose the right method based on the application understudy. Blob detectors are more robust to significant changes in the photo such as scale and skew while corner detection are faster to compute. Moreover, blob detectors ignore corners while for example urban and office environment are full of corners. For the application of VO, use of corner detectors are more popular.

2.2.3.2 Matching the features

Yet, it is desired to find corresponding features in consecutive frames. To do this one can either try to match corresponding features, *i.e.* "feature matching", or go through an index of features found in a frame and try to find the corresponding feature in the successor frames, *i.e.* "feature tracking" [1]. The latter is faster provided the features are not significantly displaced from one frame to another. For the VO application the camera rate with respect to the camera's motion is usually high enough to have this assumption held.

Nontheless, to achieve a desirable performance there are two important issues to deal with; outliers and drift. Outliers are wrong feature association caused by imperfection of the feature tracking algorithm. The effect of outliers in the estimation of the trajectory is demonstrated in Figure 2.6. As seen, to achieve a desired performance, it is necessary to remove the outliers before computing the motion. A successful solution outlier removal algorithm is RANSAC [53] which is being commonly used for the feature matching/tracking application

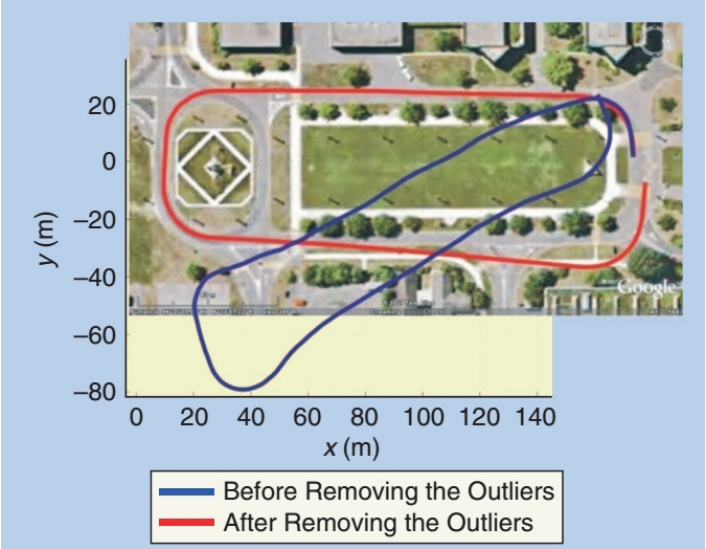


Figure 2.6: The effect of outliers in the VO performance [1].

in the ambient of computer vision. The idea, behind is very simple; RANSAC comes up with a "hypotheses" based on a sampled data points and checks the hypotheses on the rest of the data points. The hypotheses that holds for the majority of the data points is then selected and the solutions which do not comply with the hypotheses are marked as outliers.

Motion estimation in two consecutive frames even after removing the outliers is associated with a small error. These errors are accumulated in the in the course of the navigation causing the calculated trajectory drifts from the reality. To deal with drift, an optimization algorithm can be run in the background to minimize the projection error over a window of *n* consecutive frames (instead of two) by modifying the calculated camera poses and 3D landmarks for this set of images. This procedure is called "window bundle adjustment" [54] and significantly improves the VO performance.

2.3 SLAM

Simultaneous Localization And Mapping is the problem of placing environment's features in a map and at the same time localizing the robot in the map of features. Suppose there is a sample set of states X_T, a sampled set of actions U_T, and finally a set of measurements Z_T:

$$\begin{cases} X_T = \{x_0, x_1, ..., x_T\} \\ U_T = \{u_0, u_1, ..., u_T\} \\ Z_T = \{z_0, z_1, ..., z_T\} \end{cases} \tag{2.9}$$

Where the set of states represents a discrete trajectory of robot position, the actions are data from the Odometry and finally the set of measurements can be acquired from different

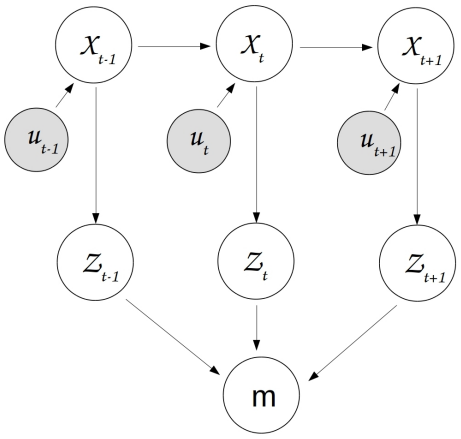


Figure 2.7: Graph formulation of the SLAM problem. A set of position, *e.g.* X_t, is estimated based on the corresponding set of odometry, *i.e.* u_t, and the previous set of position *i.e.* X_{t-1}. The measurements, *i.e.* Z_t, are then acquired and registered in a map of features based on the corresponding estimated position X_t. The SLAM is problem of correcting the estimated position such that the error introduced by position of the same features in different sets of measurement is minimized.

range sensors *e.g.* LiDAR and camera. The SLAM is then about collecting the measurements in a map, m, while recovering the sequence of states simultaneously [2]. It is therefore desired to find:

$$p(X_T, m|Z_T, U_T) \tag{2.10}$$

Note that equation above seeks the entire trajectory of the robot, X_T. In the robotic literature this is referred to as "Full SLAM" problem [2, 55, 56]. Depending on the application, the SLAM problem might be simplified to "Online SLAM" problem [57–59]. In the Online SLAM only the current location of the robot is desired:

$$p(x_t, m|Z_T, U_T) \tag{2.11}$$

To proceed with this purpose there are two important models required; First, a "system model", *i.e.* $g(x_{t-1}, u_t)$, that derives the next state of robot from the current state and Odometry data. Second, a "measurement model", *i.e.* $h(x_t, m)$, that provides a relationship the measurement corresponding state.

$$\begin{cases} x_t = g(x_{t-1}, u_t) \\ z_t = h(x_t, m) \end{cases} \tag{2.12}$$

The architecture of SLAM problem is shown in Figure 2.7. To approach the problem, there are three main paradigms that available in the literature; Extended Kalman Filter, Graph-based methods and particle-methods.

2.3.1 Kalman Filter approach to SLAM problem

The SLAM problem can be formulated in terms of a Kalman Filter where the state vector contains a set of robot locations and environment features [60, 61]. In the most simplified form, *i.e.* 2D Online SLAM, the state vector of the filter μ_t contains three values for the robot pose, *i.e.* (x, y, θ), and $2N$ position value for N features. This implies a state vector of the dimension $(3 + 2N)$ and ,subsequently, a square covariance matrix of the dimension $(3 + 2N) \times (3 + 2N)$ to be inverted at each *filter update*. As a result, the Kalman filter approach to SLAM problem does not expand properly.

2.3.2 Graph based SLAM

Graph-based approaches [62–65] draw their intuition from graphical representation of the SLAM problem, *i.e.* Figure 2.7. Formulating the *Full SLAM* problem in the form of a graph, a "node" can be either a position or a feature while an "edge" is the metric distance between two nodes. This is illustrated in Figure 2.8 where an example of two "poses" and one "feature" is demonstrated. As can be seen, *pose* to *pose* information expresses immediate predecessor and successor *poses* while *feature* to *pose* information expresses relative distance of a *feature* to any number of *poses*. In other words, the graph formulates the *Full SLAM* problem into log-posteriors where an event of robot movement gets the form of $\log p(x_t|x_{t-1}, u_t)$ and a sensor measurement expressed as $\log p(z_t|x_t, m)$. Subsequently the SLAM problem is defined as:

$$\log p(X_T, m|Z_T, U_T) = const + \sum_t \log p(x_t|x_{t-1}, u_t) + \sum_t \log p(z_t|x_t, m) \tag{2.13}$$

The SLAM problem can then be solved by finding:

$$X_T^*, m^* = \arg\max_{X_T, m} \log p(X_T, m|Z_T, U_T) \tag{2.14}$$

In a probabilistic framework, a Gaussian Noise distribution can be considered for the system and measurement models, *i.e.* :

$$\begin{aligned} p(z_t|x_t, m) &\sim \mathcal{N}(h(x_t, m), Q_t) \\ p(x_t|x_{t-1}, u_t) &\sim \mathcal{N}(g(x_{t-1}, u_t), R_t) \end{aligned}$$

Where Q_T and R_t are square matrices of the proper dimension representing noise covariance and indexed by time. Thus, the posterior of (2.13) can be expressed in the quadratic form of:

$$\begin{aligned} \log p(X_T, m|Z_T, U_T) = \; & const \\ & + \sum_t [x_t - g(x_{t-1}, u_t)]^\intercal R_t^{-1} [x_t - g(x_{t-1}, u_t)] \\ & + \sum_t [z_t - h(x_t, m)]^\intercal Q_t^{-1} [z_t - h(x_t, m)] \end{aligned} \tag{2.15}$$

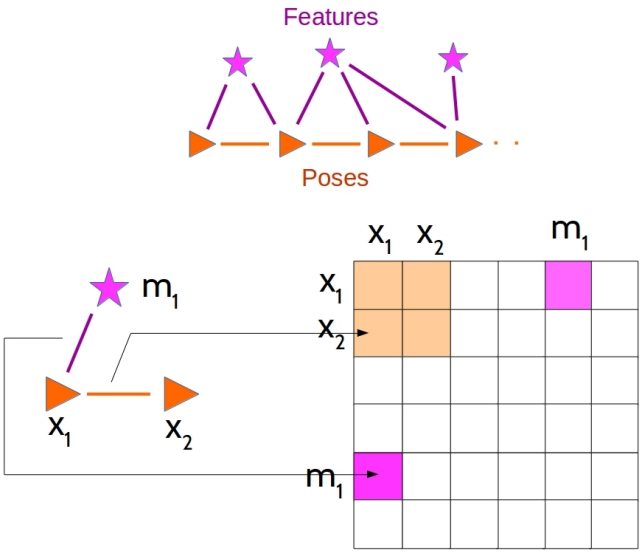


Figure 2.8: Graph Based solution to SLAM problem.

2.3.3 Particle Filter based SLAM

Finally, the last paradigm of SLAM solution is particle filter approach in which a posterior is being represented by a set of "particles". A particle is a guess about the robot pose and the map of features. As a result each particle encodes considerable amount of data, *i.e.* particle pose and features, that is the main bottleneck on implementation of this approach to SLAM problem. To address the issue, two important concepts of "Importance factor" and "Resampling" are introduced by the literature [66–69]. The idea behind introduction of these concepts is to depopulate the particle swarm by evaluating likelihood of the particles, based on the measurement and system models, and eliminate those who are less likely to be a true.

Here is how the algorithm works; Based on the probability distribution model assumed for the robot a random set of particle is generated. On the event of a robot movement (*i.e.* a new Odometry data), each particle is moved based on the stochastic kinematic robot model. Thus, assuming particles are indexed by k we have:

$$x_t^k \sim p(x_t|x_{t-1}^k, u_t) \tag{2.16}$$

The next step takes place on the event of a measurement when the algorithm assigns an *importance factor* to each particle based on the new measurement. Assume that each landmark is indexed with n with mean value of μ and covariance of Σ. The *importance factor* of particle k is defined as follows:

$$w_t^k := \mathcal{N}(z_t|x_t^k, \mu_{t,n}^k, \Sigma_{t,n}^k) \tag{2.17}$$

The *importance factor* is then used in *Resampling* step where a subset of the particle swarm is sampled to survive while the probability of a particle being drawn is its *normalized*

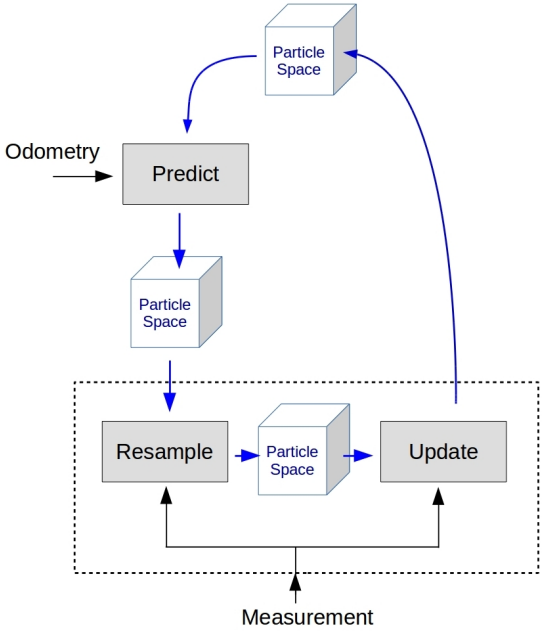


Figure 2.9: Summary of Particle Filter solution to SLAM problem.

importance factor. Subsequently the particles who are less likely to present the true state of the robot are eliminated and, ultimately, the particle swarm is depopulated.

Finally, the new set of measurement is used to update $\mu_{t,n}^k$ and $\Sigma_{t,n}^k$ based on standard Kalman filter update procedure. This is depicted in Figure 2.9.

2.3.4 SLAM implementations

2.3.4.1 Laser SLAM

General SLAM architecture, can be implemented for either a fixed or a tilting LiDAR. Laser SLAM privileges the inherent advantages of LiDARs, *i.e.* high accuracy, long ranges and a performance not affected by the lighting conditions and visual features. On the other hand, LiDARs are usually expensive, heavy and they do not provide a promising 3D measurement tool even in a tilting configuration.

An implementation of particle Filter based laser SLAM can be found in [25, 26] which is available open source and known as "gmapping" [70]. Using the odometry data and Rao-Blackwellized particle filter, they seek to reduce the particle swarm and ultimately converge to a concrete hypothesis about the position of the robot. Taking the advantage of high accuracy measurement of the LiDAR, a faster approach can be achieved using the "scan matching" methods [66, 71] which can be classified under the graph based SLAM category. A reliable implementation of this kind can be found in [71] which is available open source as "Hector

Figure 2.10: An example of Laser SLAM.

SLAM" [72]. Using an occupancy grid based strategy, the algorithm creates a map of the first scan and compares each scan with the map to come up with an idea about the movement of the robot. Therefore, as the robot proceeds to explore the environment, the position is calculated and new features are added to the map. Furthermore, this method is also similar to ICP which was mentioned in Laser odometry, although using a using occupancy grid lower the computational burden of the algorithm. Moreover, by incorporating the IMU data into the Laser measurements, *Hector SLAM* is also able to provide a limited 6 DOF localization. Figure 2.10 demonstrates an example of Laser SLAM created by the *Hector SLAM* algorithm.

2.3.4.2 Visual SLAM

Applying *window bundle adjustment* is a step toward SLAM as the localization is closed loop for the window of frames. In the visual SLAM, the concept of window bundle adjustment is expanded to "key frames". *Key frames* form a subset of the set of all frames acquired in the course of running the algorithm and the bundle adjustment is performed among the *key frames*. An example of map of features created by a visual SLAM algorithm is demonstrated in Figure 2.11 where the features and key frames are being stored. Visual SLAM can be classified under the category of Graph based SLAM. In this context the concept of *pose* is equivalent to the concept of *key frame*.

A successful implementation of monocular visual SLAM can be found in [73] where the authors use ORB features [74] which is a combination of FAST and BRIEF [75] and provides a good degree of robustness against change of viewpoint and illumination. Using the same features for all the subtasks of "tracking", "mapping", "relocalizing' and "loop closing", they have developed a fast and efficient system that is also available open source as "ORB-SLAM". However, the classical problem of scale ambiguity for monocular visual system also applies here. In another work [76], the authors have come up with a platform that uses RGB-D cameras (*e.g.* kinect) or Stereo vision to cope with the scale ambiguity. As an extension

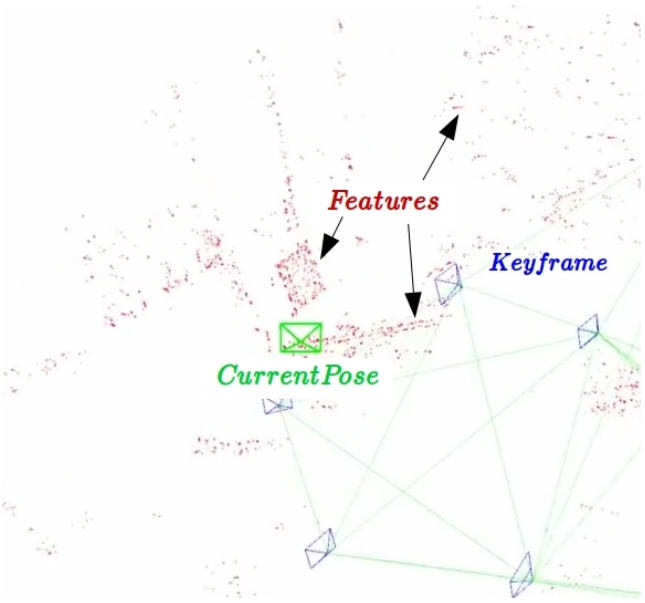


Figure 2.11: An example of map of features created by a visual SLAM algorithm.

to their previous work, this works is available open source as "ORB-SLAM2". Figure 2.12 demonstrates a depth map constructed by *ORB-SLAM2* algorithm.

2.4 Global Positioning System and Filtering

The Global Positioning System (GPS) is a localization system supported by satellites orbiting around the earth covering almost every where on the planet surface. The localization works based on the data broadcasted from the satellites that contains a precise time stamp and the satellite position. The GPS receiver has to receive data packs from at least three different satellites to estimate its position while accuracy of the GPS depends on the number of satellite that have been discovered by the receiver. In average, it is reasonable to consider an uncertainty of 20 meter. Moreover, the GPS estimated position is not being provided in a continuous manner and the update rate can be too slow for the position loop closer. As a result, to achieve a continuous and precise localization it is necessary to fuse GPS data with other sensory information of the robot.

2.4.1 Complementary Filter

The main idea behind the complementary filters is fusing the information coming from the system model and different sensors to achieve a superior knowledge about a state of the system. Assume the state x has to be estimated from two different values of ${}^1x$ and ${}^2x$ while either of the values describes x with a different uncertainty. A complementary filter estimates the

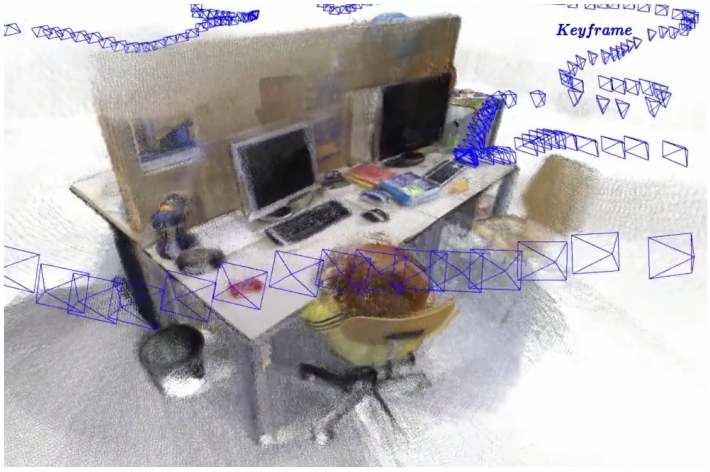


Figure 2.12: An example of a Depth frame created by ORB-SLAM2.

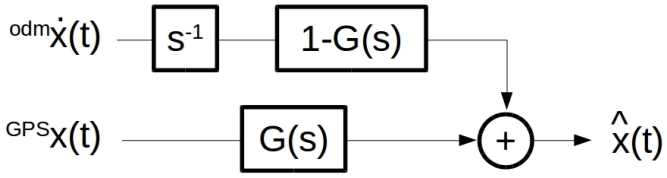


Figure 2.13: Complementary filter for fusing odometry and GPS information

state as follows:

$$\hat{x} = {}^{1}xG + {}^{2}x(1 - G) \tag{2.18}$$

Where $\hat{x}$ is the estimated state of the system and G is a real positive value less or equal to one. This concept can be extended by defining the filter gain as a system $G(s)$ to fuse the information based on the frequency [77]. This is particularly important when we have to deal with the noisy sensory information. We can, therefore, extend (2.18) as follows;

$$\hat{x} = {}^{1}xG(s) + {}^{2}x(1 - G(s)) \tag{2.19}$$

Figure 2.13 shows a classical application of complementary filter in position estimation of mobile robots. The GPS information is associated with a high frequency noise while odometry is subjected to accumulative errors *i.e.* a low frequency noise. Moreover, the architecture is capable of filtering the noise in both input signals before data fusion.

2.4.2 Kalman Filter

Kalman filter presents a stochastic extension to complementary filter that considers model uncertainties and measurement noise. This is particularly important in robotics where there

is a need to fuse data from different sensors while each means of measurement has its own uncertainties. Considering a Gaussian distribution for both system and measurement models, the Kalman filter "predicts" the system state using the previous state, system model and control input. The predicted state is then incorporated in the measurements in the "update" step to come up with a new belief about the state.

Equation (2.20) presents the system and measurement models of the Kalman filter framework. The first line is the linear transitional system model in which A is a square matrix of the size $n \times n$ where n is dimension of the state vector x. Furthermore, u is the control vector of the dimension m and B is a matrix of $n \times m$. The second line presents the measurement model where for k measurements C has a dimension of $k \times n$. Finally ε_t and δ_t are Gaussian distributed noise vectors to model the uncertainties.

$$\begin{cases} x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t \\ z_t = C_t x_t + \delta_t \end{cases} \tag{2.20}$$

Remember, a Gaussian distributed posterior is formulated by multi variable normal distribution:

$$p(x) = det(2\pi\Sigma)^{-\frac{1}{2}} exp\left(-\frac{1}{2}(x-\mu)^{\intercal}\Sigma^{-1}(x-\mu)\right) \tag{2.21}$$

Where the probability of the vector x is characterized by a mean value of μ and a quadratic symmetric positive definite covariance matrix *i.e.* Σ.

Algorithm of the Kalman filter is presented in Algorithm 1 where R_t and Q_t are covariance matrices associated with ε_t and δ_t in (2.20) respectively. The *predict* step is accomplished in lines 1:2 while lines 4:5 refer to *update* step. Furthermore, Kalman gain K_t is being calculated in line 3 which is considered the most computationally expensive step of the algorithm while the complexity increases as the measurement vector grows. Referring to (2.18), it should be noted that the Kalman gain is equivalent to G. Thus Kalman filter adaptively adjusts the filter gain based on its notion of system and measurement uncertainties.

Algorithm 1 Kalman Filter Algorithm [78]

Require: $\mu, \Sigma_{t-1}, u_t, z_t$
1: $\bar{\mu}_t = A_t\mu_{t-1} + B_t u_t$
2: $\bar{\Sigma}_t = A_t\Sigma_{t-1}A_t^{\intercal} + R_t$
3: $K_t = \bar{\Sigma}_t C_t^{\intercal}(C_t\bar{\Sigma}_t C_t^{\intercal} + Q_t)^{-}1$
4: $\mu_t = \bar{\mu}_t + K_t(z_t - C_t\bar{\mu}_t)$
5: $\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$
6: **return** $\mu_t,\ \Sigma_t$

2.5 Mapping

An other important topic that we need to deal with is mapping. Whether it is complicated map of features, or a simple obstacle map it is important to investigate how to store data in a way that is easily accessible and computationally light. In the context of rough terrain navigation, the map serves for evaluating traversablity and chassis terrain interaction. In the this section a brief overview on 2D and 3D mapping techniques in the literature is provided.

2.5.1 2D mapping

2D metric mapping strategies suggested by the robotic literature can be categorized into three groups namely "Line Maps", "Topological Maps" and "Occupancy grid". Among those, occupancy grid is the most popular and has been used frequently for different robots. Here we only explain the Occupancy grid, for further readings, we encourage the reader to refer to [2].

Occupancy grid provides easy access to the information while it is also capable of presenting unobserved areas. On the other hand, discretization error and high memory requirement are two disadvantages of this method. The idea behind it is very simple, it discretizes the environment into a grid while each cell of the grid represents a posterior probability of the traversablilty of corresponding area. The information is maintained in form of a Matrix while rows and columns correspond to x and y axes of the inertial frame. Such a data structure has advantage of high accessibility *i.e.* origin of the matrix and resolution are all that is needed to acquire the data corresponding to the external world. On the other hand, it implies keeping a rather large database to store the grid information while the grid dimensions increases as the robot moves. This issue is the bottleneck of discretization algorithm making them impractical in 3D mapping scenario.

Occupancy grid assumes an independent probability for each cell which is rather a conservative assumption since in practice an object has a bigger dimension that a single cell and has to be presented by several cells. The default probability of a cell being occupied is considered 0.5. This probability might then rise or fall depending on the observation. Such a structure facilitates deployment of highly noisy sensors like ultrasonic. An example of *Occupancy grid* is demonstrated in Figure 2.14.

2.5.2 2.5D mapping

Elevation map also known as 2.5D map can be considered as a 3D extension to 2D grid approach. The idea of the plane discretization is the same as occupancy grid, although in an elevation map, the traversability probability is replaced with elevation data. This simplifies a lot the 3D mapping and basically project the 3D environment into a 2D grid. On the down

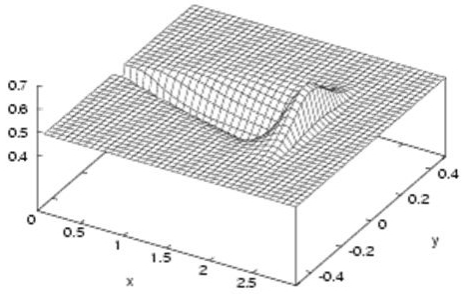


Figure 2.14: A sample occupancy grid with default probability of 0.5 [2].

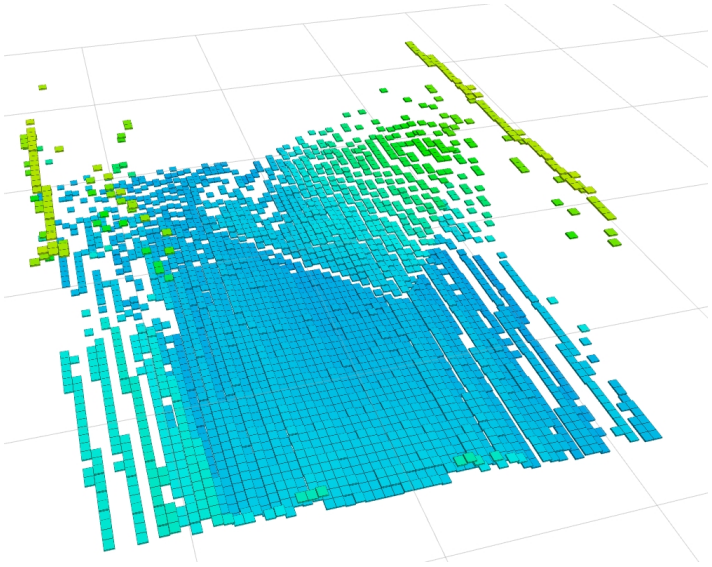
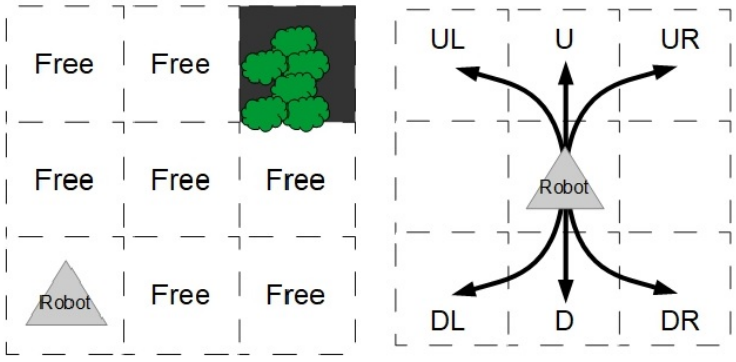


Figure 2.15: An example of elevation map.

side, this method is not capable of representing multilayer environments, *e.g.* a tunnel, since there is only one elevation value allowed for a cell. As a result this approach is convenient for the ground robots navigating on unstructured outdoor environment. Figure 2.15 demonstrates an elevation map created by tilting LiDAR mechanism of the *Donkey Rover*.

2.5.3 3D mapping

Some applications cannot afford the limited precision and direction dependency of the elevation grids. In this case, 3D maps can be used in which the 3D points received by the sensor is not being projected to a 2D plain and the consequent problems can be avoided. On the other hand, 3D data are huge which makes the store and access challenging. In the robotic literature, data structures based on dynamic 3D grids and meshes are proposed to overcome these problems. However, for the application understudy elevation maping suffices. For more information about the 3D mapping, we refer the reader to [2].



(a) Discretization of the environment.

(b) Discretization of the robot action.

Figure 2.16: Discretization of environment and robot action that helps to decrease the space of solution. In 2.16b U,D,L and R respectively stand for Up, Down, Left and Right.

2.6 Path Planning

Given a map of the environment and current position of the robot in the map is provided, the path planning answer to the question of how to define a path that can **safely** take the robot to the desired destination. As mentioned earlier, it is common to perform a state discretization on the environment to decease the space of possible solutions. As a result, the environment is transferred to a grid where each cell represents properties of the corresponding area. In the most trivial case, the cell can have either values of "occupied" and "free". This idea is illustrated in 2.16a. The path planning problem is defined as finding a sequence of cells that takes the robot from the start cell to the goal cell. In the computer since literature, different search algorithms have been proposed to address this problem, *e.g.* Dijkstra, Astar, Dstar, Dstar lite.

In practice, however, the problem of path planning is coupled with the kinematics and dynamics of the robot. In other words, the robot may not be able to execute the sequence solution due to its kinematics and dynamics constraints. To address the issue, one can extend the trivial path planning problem from finding a sequence of *cells* to finding a sequence of *actions* that can take the robot to its destination. Again to decrease the solution space, robot space of actions can be discretized into a set of "primitives". The *primitives* are a set of actions that comply with kinematic and dynamic constraints of the robot. An example of action primitives for a ground rover is demonstrated in 2.16b. Due to nonholonimic constraints, the ground rover is not able to perform left/right action. As can be seen in the figure this property is encoded in the primitives as *Right* and *Left* primitives are not defined for the robot.

Among many different strategies proposed by the literature, Discrete Event System (DES) theory provides a powerful tool to approach the problem of path planning. Therefore, throughout the rest of this section, we focus on DES planner. This is part is mainly derived from [79] where an extensive tutorial about DES based planner is provided.

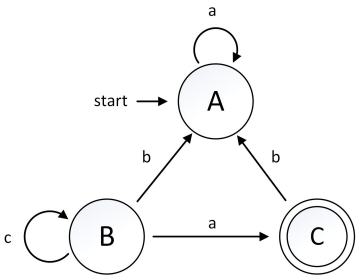


Figure 2.17: An example of Automaton representation.

2.6.1 Introduction to DES theory

DES defines a system in which the state space is discrete and transitions among "states" take place trough the "events". A DES can be represented by a six-tuple as follows:

$$G = (X, E, f, \Gamma, x_0, X_m) \tag{2.22}$$

Where X, E and f are respectively sets of states, events and transition functions. Furthermore, $\Gamma : X \rightarrow 2^E$ refers to set of all events for which $f(x, e)$ is defined. Finally, x_0 and X_m are set of initial and marked states respectively.

"Automaton" is another useful representation of a DES. In an *Automaton*, circles represent *states* while arrows represent *events*. Moreover, the transition function are nicely encoded in the Automaton representation. For example in the automaton representation of Figure 2.17 it can be seen $f(A, c) = A$, $f(B, a) = C$, $f(B, b) = A$, etc.

One of the most used operation on Automata[4] in DES planner is "parallel composition". Yet we need to define "Accessible Part" operation that is a prerequisite to *parallel composition*. Accessible part is of Automaton G is defined as follows:

$$\begin{aligned} &Ac(G) = (X_{ac}, E, f_{ac}, x_0, X_{ac,m}) \\ &X_{ac} = \{x \in X : (\exists s \in E^*)[f(x_0, s) = x]\} \\ &X_{ac,m} = X \cap X_{ac} f_{ac} = f|_{X_{ac} \times E \rightarrow X_{ac}} \end{aligned}$$

Where E^* is the set of all finite possible sequence of events in E.

For two Automata G_1 and G_2, the *parallel composition* operator is defined as follows:

$$G_3 = G_1 || G_2 = Ac(X_1 \times X_2, E_1 \times E_2, f, \Gamma_{1||2}, (x_{01}, x_{02}), X_{m1} \times X_{m2})$$

[4]plural of Automaton

where

$$f((x_1,x_2),e) = \begin{cases} (f_1(x_1,e), f_2(x_2,e)) & if e \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ (f_1(x_1,e), x_2), & if e \in \Gamma_1(x_1) \setminus E_2 \\ (x_1, f_2(x_2,e)), & if e \in \Gamma_2(x_2) \setminus E_1 \\ undefined, & Otherwise \end{cases}$$

2.6.2 DES path planner

To provide a solution for the path planning problem, a DES planner defines some automata for the *environment* and *robot actions* respectively referred to as "Map Automaton" and "Agent Automaton". The *Map Automaton* is further elaborated to specifically represent the problem path planning with environmental constraints, initial state and final state. The resulted automaton is called "Specification Automaton". Furthermore, *Agent Automaton* summarizes robot capabilities in compliance with kinematic/dynamic constraints and can be integrated into the *Specification Automaton* through a *parallel composition*. The resulted Automaton is called "Supervisor". The *Supervisor* represents the path planning problem in terms of a graph that can be searched by aforementioned graph search algorithms (*e.g.* Astar) to come up with a sequence of actions, *i.e.* pieces of the path, to be executed by the robot. The reminder of this section explains design of the DES planner automata in detail.

2.6.2.1 Map Automaton

As its name suggests the *Map Automaton*, *i.e.* G_{map}, contains the grid presentation of the environment by encoding the grid resolution and interconnection. Figure 2.18 illustrates an example of a Map Automaton for a grid of 9×9. The interconnection of the automaton implies, except marginal states, each state can be reached though four events of "Up", "Down", "Left" and "Right". Notice, the map interconnection does not constrain the robot movement. For example, robot can still perform a diagonal movement that can be carried out through two *events*. An other possible point of confusion to explain is that the Map Automaton does not represent any information about the traversability of the grid. Indeed, as we will see in the reminder of this section, this information will be encoded in the *Specification Automaton*.

2.6.2.2 Specification Automaton

As mentioned earlier, the Map Automaton is too generic to be used to solve the path planning. Thus, *Specification Automaton*, *i.e.* G_{spec}, elaborates the Map Automaton by applying to it the problem specific constraints *i.e.* obstacles, start and goal states. Figure 2.19, illustrates how the Specification Automaton is derived from the Map Automaton of Figure 2.18. As seen

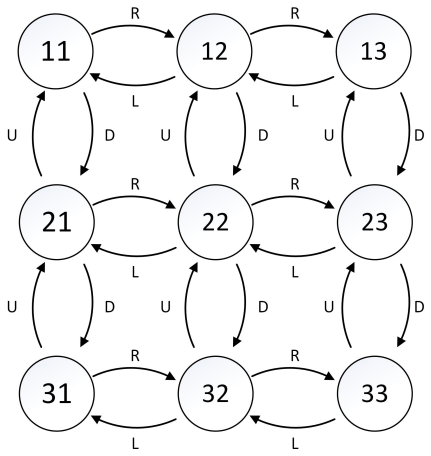
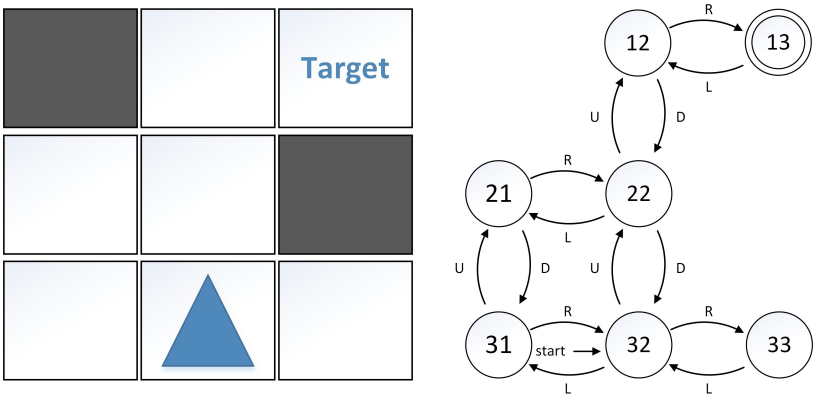


Figure 2.18: Map automaton representing a grid of 4×4.



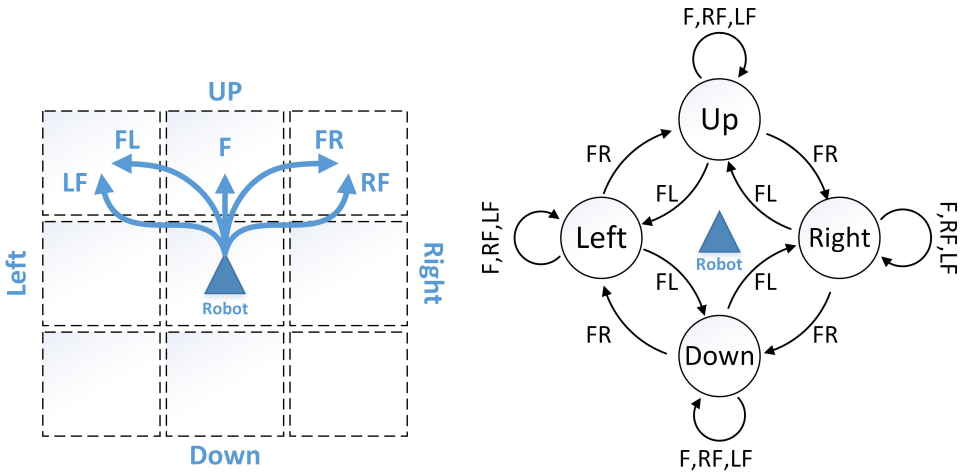
(a) path planning problem. (b) Specification Automaton.

Figure 2.19: An example of a path planning problem and corresponding specification automaton derived from map automaton of Figure 2.18 and problem constraints implied by 2.19a.

in the Figure, untraversable states are simply removed. As a result, the more untraversable cells present in the grid, the fewer states exist in the Specification Automaton and ultimately the simpler is the search.

2.6.2.3 Agent automaton

Agent Automaton describes robots manoeuvrability through the *primitives*. As mentioned earlier, a primitive describes a piece of trajectory that is kinematically feasible and provides a good access to neighbouring cells of the map. For the Specification Automaton, executing a primitive is equal to a series of events. Although, a primitive may also change the robot heading which should be presented by the Agent Automaton. Consequently, for N primitives



(a) Primitives.

(b) APL automaton.

Figure 2.20: An example of APL for a ground robot with nonholonomic constraints and five primitives of *Forward*(F), *Forward Left* (FL), *Forward Right* (FR), *Left Forward* (LF) and *Right Forward* (RF).

we need $N+1$ automata, one of them illustrating Agent Primitives Logic (APL), *i.e.* G_{APL}, and the rest (*i.e.* N) embedding the Agent-Map Interconnection (AMI), *i.e.* G_{AMI}.

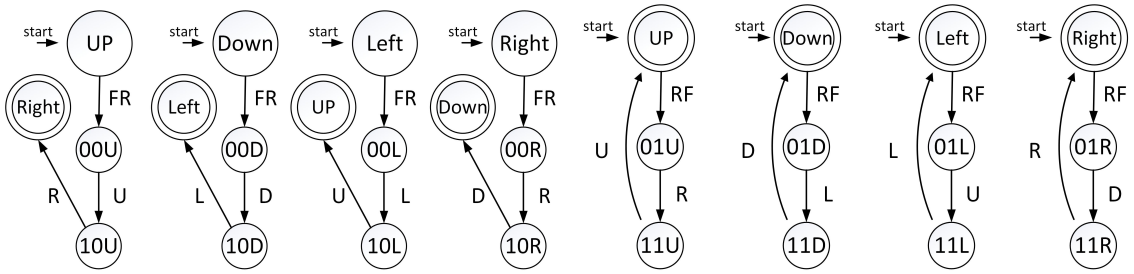
Figure 2.20 depicts an example of APL where five primitives are considered as shown in Figure 2.20a. Notice, in case the robot is not able to see the behind due to limited sensor field of view, it is not desirable to move backward. Figure 2.20b demonstrates APL Automaton where states represent the robot heading and events show the primitives. For example, if the robots heading is *UP* (U), performing a "Forward Left" (FL) or "Forward Right" (FR) primitives respectively changes the heading to *Left* (L) or *Right* (R) while either of "Forward" (F), "Left Forward" (LF) and "Right Forward" (RF) does not affect heading of the robot in the next state.

Figure 2.21 illustrates the set of AMI automaton for the example of Figure 2.20a. As mentioned earlier, there are five set of automata for the five primitives while each set considers four possible heading of "Up", "Down", "Left" and "Right". Finally, parallel composition of APL and AMI automata gives the Agent automaton G_{ag}:

$$G_{ag} = G_{APL} \,||\, G_{AMI1} \,||\, G_{AMI2} \,||\, \ldots \,||\, G_{AMIn} \tag{2.23}$$

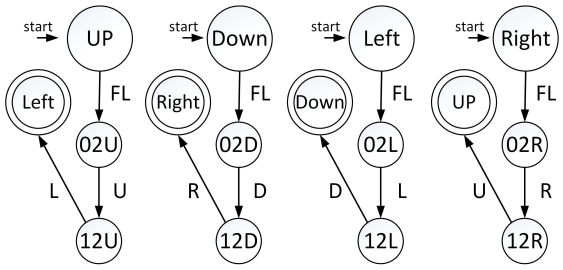
Where N is the number of primitives which is five in the example understudy. Considering the Figures 2.20 and 2.21 one can easily imagine how the agent automaton looks like. However because of sheer number of states and events the scheme of automaton is not presented.

Another nice feature of this approach is the ability of considering the robot dimensions that usually exceed the map cell size. General approach in the literature is to sufficiently inflate the obstacles so the robot can be considered a point. This method, however, is not expandable to the case of rough terrain navigation where an elevation map is used to present

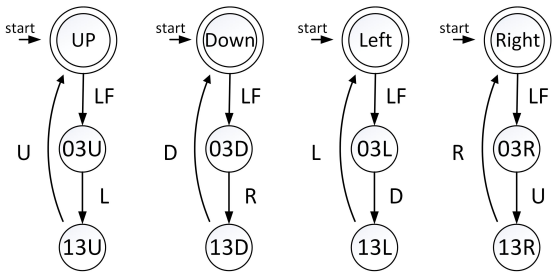


(a) Froward Right primitive AMI.

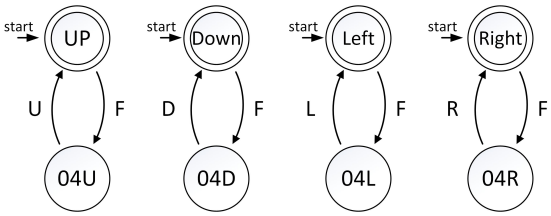
(b) Right Froward primitive AMI.



(c) Froward Left primitive AMI.



(d) Left Froward primitive AMI.



(e) Froward primitive AMI.

Figure 2.21: Set of AMI automata for the example of Figure 2.20a.

the terrain's roughness. Instead, the planning approach understudy allows us to consider a swathe of map cells that will be navigated by each primitive. An example of this case is illustrated in Figure 2.22. As seen, the demonstrated primitive is translated into the map via a movement of "*L,U,L,U,L,U,R,D,R,D,R*" starting from the started cell.

2.6.2.4 Supervisor

Finally the supervisor is defined as parallel composition of Agent and Specification automata:

$$G_{sup} = G_{ag}||G_{spec} \tag{2.24}$$

The *Supervisor* represents the reachability graph of the problem. It summarizes both the environment and robot constraints,*i.e.* obstacles and robot kinematics/dynamics respectively. Figure 2.23 shows a part of the Supervisor for the example of Figure 2.19. As seen, cell "23" is occupied. Moreover, the initial heading is "UP". The Robot may then execute only three possible primitives of "F", "FL" and "LF".

The presented approach is able to demonstrate whether a mission is feasible in a very

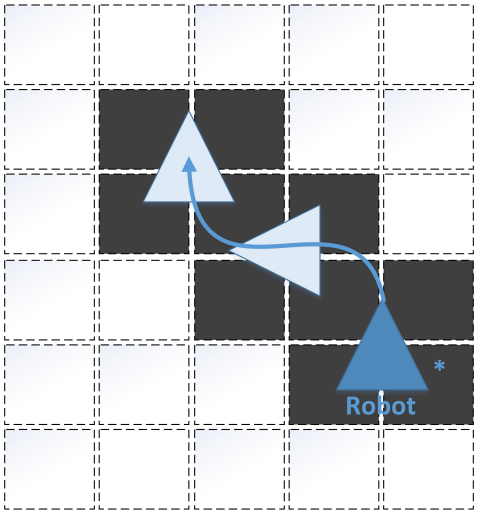


Figure 2.22: An example of a swathe presented to consider the vehicle dimensions. The primitive is translated into the map via a movement of "*L,U,L,U,L,U,R,D,R,D,R*" starting from stared cell.

abstract manner. Indeed, the mission is feasible if and only if the target cell is presented in the Supervisor, *i.e.* the set of marked states of the Supervisor is not empty.

2.6.3 DES planner for rough terrain

The presented path planner can be easily extended to rough terrain navigation. For this purpose, a rough terrain can be presented by means of an elevation map. Moreover, a traversability analysis should be done on the map to eliminate the untraversable states from the Specification Automaton. Figure 2.24a demonstrates a piece of terrain that is not traversable with respect to the robot size. Notice, traversability depends on both terrain and robot specifications. For example a piece of terrain that is not traversable for one robot might be traversable for another.

2.6.3.1 Cost Function

To be able to search the Supervisor for an optimal solution with a search algorithm, a proper "cost" value needs to be assigned to the primitives. The cost has a direct impact on the performance of the algorithm. For example a shorter path is always preferred, therefore, straight line should have a lower cost than a curve. Moreover, in the presence of an elevation map, one can also encode difference in elevation of the path's cells in the cost function. This is because, navigating on the planar part of the terrain implies lower energy consumption and risk and consequently is preferred (see Figure 2.24b and 2.24c).

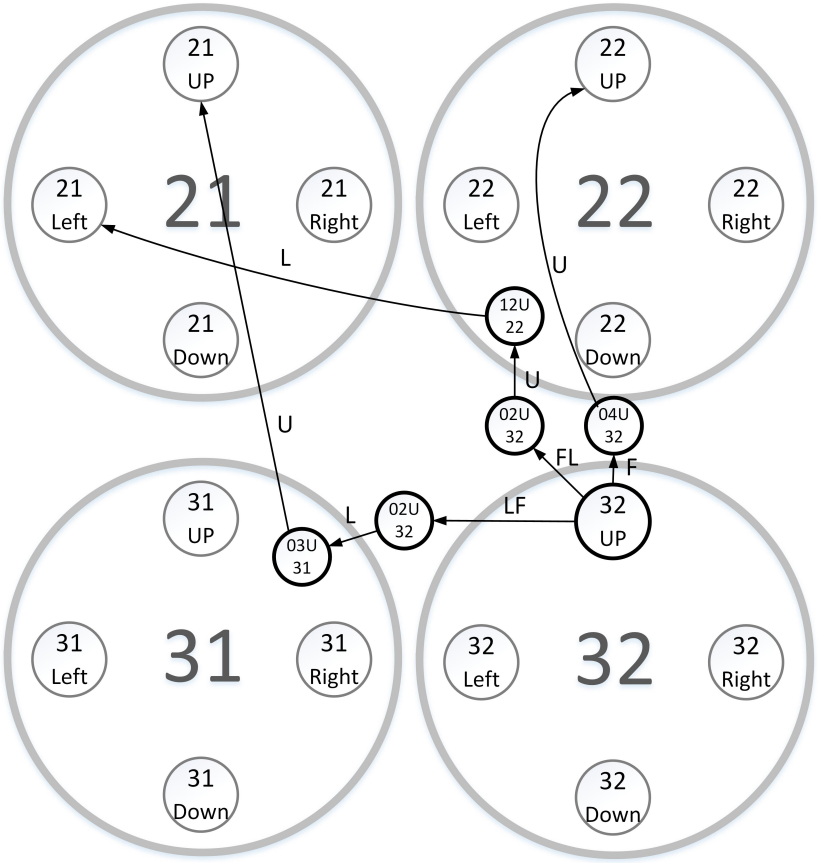
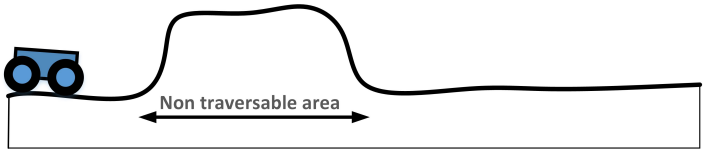
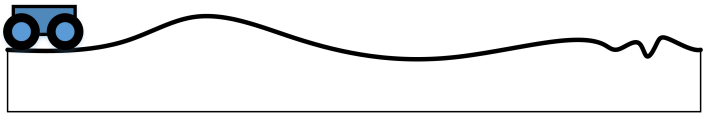


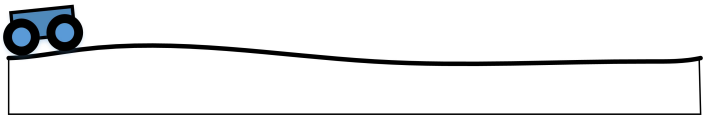
Figure 2.23: Part of the supervisor for the example of Figure 2.19 give the start heading is "UP".



(a) Non traversable terrain.



(b) High roughness.



(c) Low roughness.

Figure 2.24: Illustration of different levels of terrain roughness and traversability.

2.6.3.2 Chassis simulator

It is also possible to integrate a chassis simulator to the planner. This could be done through two possible strategies. One is to take into account robot terrain interaction in the process of graph search. This strategy allows us to come up with a more optimized path. However, considering that searching the Supervisor is already computationally expensive, imposing this extra computation on the system might impede the program to come up with a solution in real time. An improvement is to simulate the chassis terrain interaction only for a sampled set of primitives. The second strategy is only to verify the solution. This however does not allow us to optimize the path in terms of chassis terrain interaction. Yet the system can avoid the solutions implying a hazardous manoeuvre.

In this approach, we only define planar primitives. This is because, for a passive chassis ground robot, there is no control on the height and it is imposed by the shape of the terrain. Figure 2.25 demonstrates the effect of terrain roughness on the original primitive. As seen, after taking into account the roughness of the terrain, the primitive looks rather different. The chassis simulator is, therefore, necessary to estimate result of the terrain chassis interaction from the 2D primitive and elevation data.

A kinematic chassis simulator can iteratively estimate the robot footprint by locating the cells holding the tires/tracks of the robot and performing some geometrical analysis based on the elevation data associated with these cells. This idea is illustrated in Figure 2.26. As mentioned earlier this needs to be done on a sampled set of points from the path.

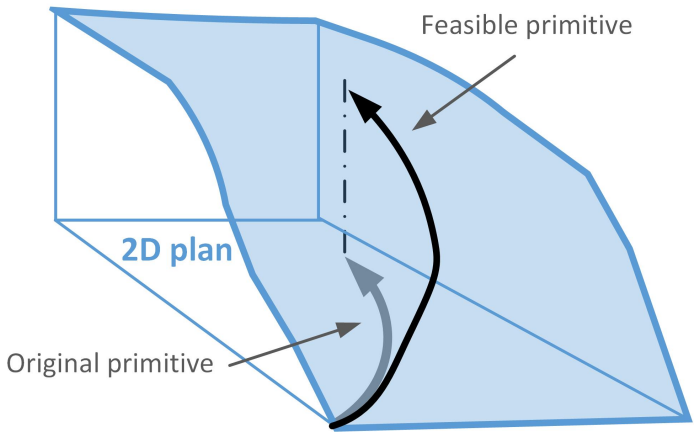


Figure 2.25: Effect of the plane roughness on the primitives.

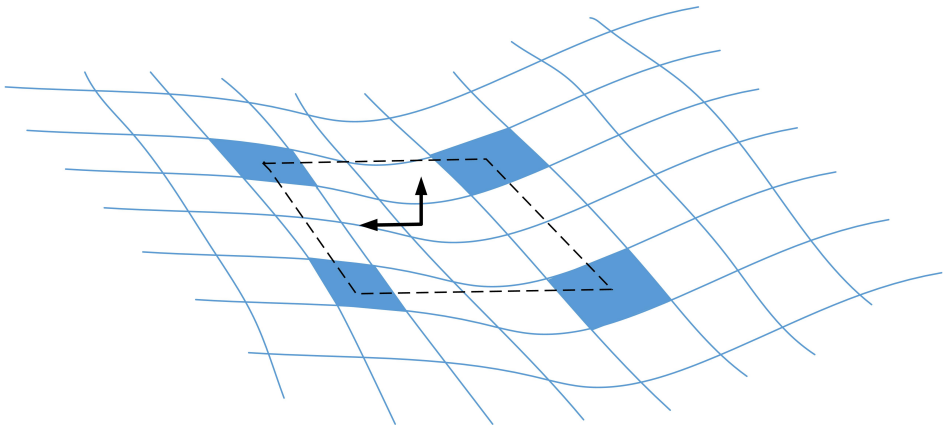


Figure 2.26: Geometric chassis simulation.

Chapter 3

Introduction to the Hardware of the System

The Sherpa ground rover, Sherpa arm and Sherpa box are three elements of this research. The rover provides a mobile station for the wasp's power replenishment while the arm and Sherpa box complete the battery exchange operation. As introduced in section 1.3.3, the arm is used to pick up the wasp and dock it to the Sherpa box while the Sherpa box must latch the wasp and exchange its exhausted battery with a fresh one. An overview of the hardware set-up of this research is demonstrated in Figure 3.1. In this chapter we study the Sherpa ground rover, Sherpa robotic arm and Sherpa box from the hardware point of view.

3.1 Sherpa Ground Rover

The ground rover is designed to perform a rough terrain navigation in the hostile environment considered by the SHERPA project. The hardware is required to provide the desirable level of autonomy, *i.e.* 6 hours, with the minimum attendance requirement by the rescuer. The hardware of the Sherpa ground rover is designed and implemented by the Bluebotics. An overview of hardware of the Sherpa rover is demonstrated in Figure 3.2. This section introduces the reader to different aspects of the rover's hardware.

3.1.1 Passive Configurable Chassis

To be able to cope with rough terrain navigation as required by the SHERPA project, the hardware must be able to provide a certain degree of mechanical flexibility. This can be achieved by a configurable chassis. Due to the desired high level of autonomy, an active configurable chassis is difficult to deploy. As a result, the Sherpa ground rover is equipped

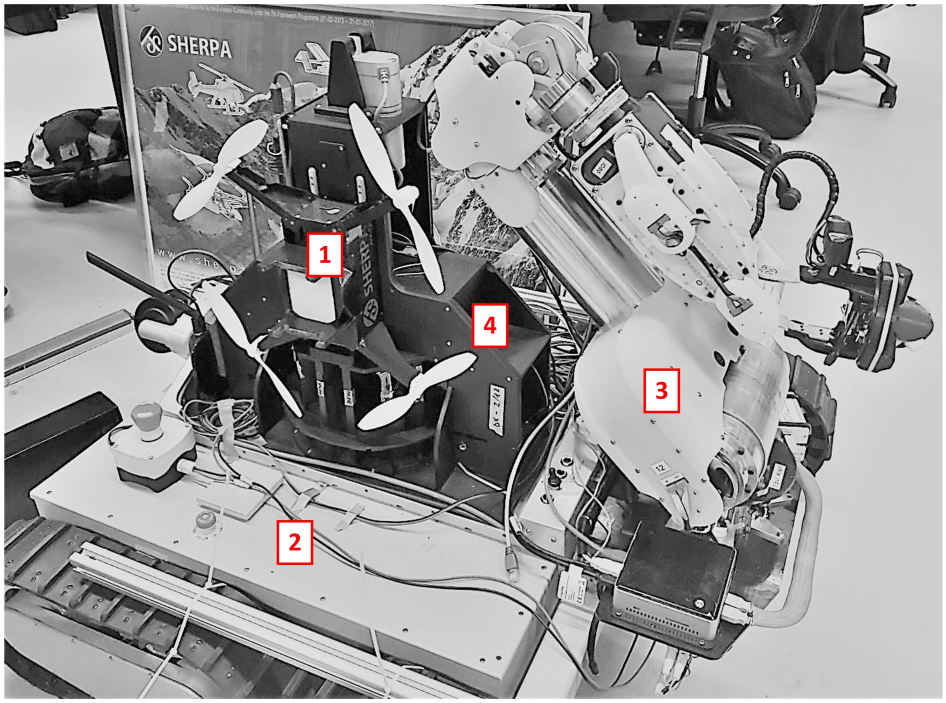
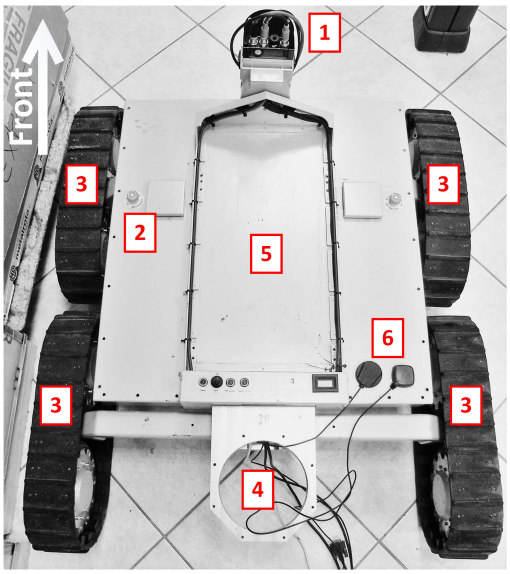
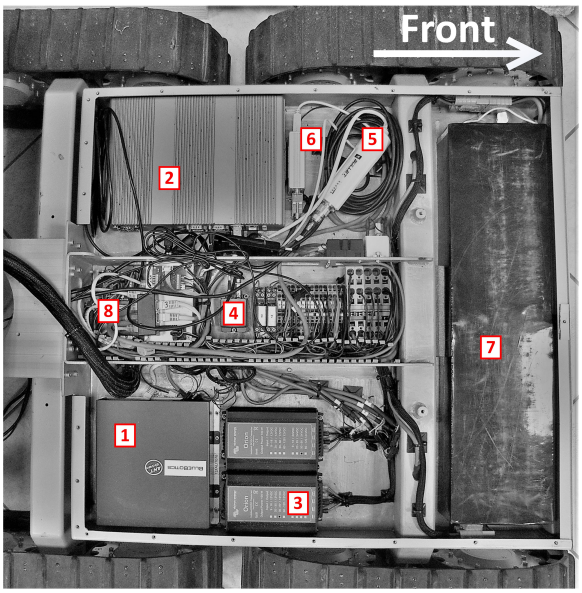


Figure 3.1: An overview of the system Hardware. 1- Wasp, 2- Sherpa Ground rover, 3- Sherpa Arm, 4-Sherpa box.



(a) Rover top view



(b) Rover Internal view

Figure 3.2: Ground rover hardware overview; (a) 1- LiDAR, 2- Emergency stop button, 3- Tracks, 4- Sherpa arm mounting space, 5- Sherpa box mounting space, 6- Wifi antenna and GPS receiver. (b)1-Bluebotics process unit, 2- Main process unit, 3- DC power supply of the LiDAR, 4- IMU, 5- Bullet router, 6- CAN BUS adapter, 7- Battery, 8- Ethernet switch.

Figure 3.3: The Sherpa ground rover is equipped with a passive configurable chassis. The flexibility of the chassis facilitates rough terrain navigation while increasing the stability of rover.

with a passive configurable chassis, powered by the gravity force, that adapts to the shape of the terrain. This allows the system to navigate rather rough terrains with low risk of getting stuck and flip over. Figure 3.3 demonstrates the chassis of the ground rover. The configurable chassis is provided by five bearing while each bearing is associated with a magnetic encoder that communicates with the process unit via CAN Bus.

The chassis of the rover has to carry the Sherpa arm and Sherpa box. For this reason the corresponding mounting spaces are considered. The arm mounting space is considered at the rear of the rover to ensure mechanical stability of the system. The Sherpa box, on the other hand, is considered to be installed on the top of the rover between the LiDAR and arm. As a result, it is easily accessible for the arm to facilitate the battery exchange operation. The mounting space of the Sherpa box and Sherpa arm are demonstrated in Figure 3.2a.

3.1.2 Sensors

To realize the robot perception a set of sensory information is required. The sensory data are to be transmitted to the process unit through the internal network of the system. The data later will be processed by the process unit. As the orientation sensor, the Sherpa ground rover is equipped with an Inertial Measurement Unit (IMU) and a GPS receiver. Moreover, the system uses a medium range LiDAR system associated with a Gimbal that provides a tilting movement to perform 3D range measurements.

(a) LiDAR

(b) IMU

Figure 3.4: Sensors of the Ground rover (a) Range sensor, as demonstrated in the figure the gimbal can provide a tilting movement to perform a 3D measurement. (b) The IMU is located on the rover's center of rotation introduced by the skid drive mechanism.

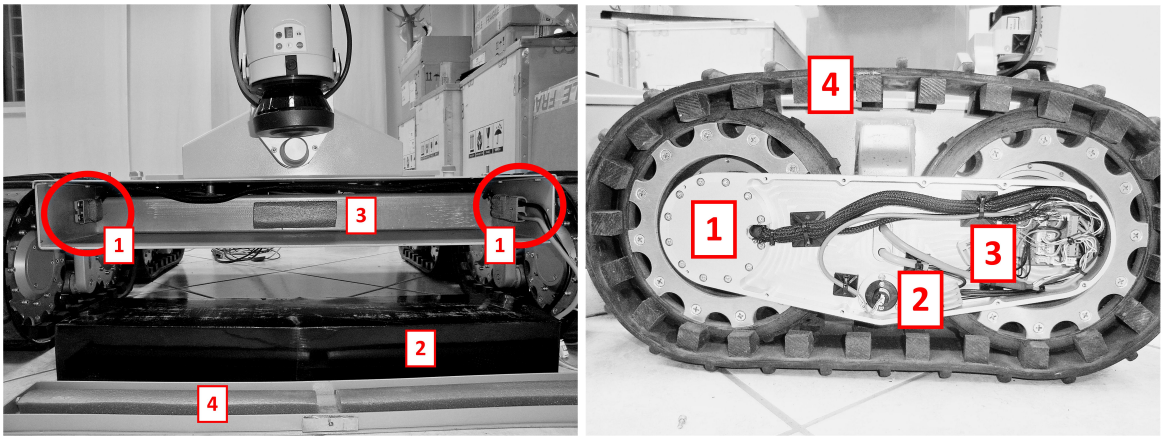
3.1.2.1 Orientation Sensor

As the IMU a Xsense MTi-G-700 GPS/INS is installed on the Rover. In order to provide a fully integrated solution for the 3D orientation, the IMU set includes an onboard GPS receiver. The MTi-G-700-GPS/INS is thus capable of not only outputting GPS-enhanced 3D orientation, it can also output augmented 3D position and velocity, so that velocity and position accuracy significantly improve with respect to the accuracy of the GPS receiver alone. Furthermore, it provides 3D sensors data, such as acceleration, rate of turn, magnetic field, the navigation solution of the GPS receiver and static pressure. Data generated from the strapdown integration algorithm are available, as all other processed data, at a frequency of 400 Hz.

To avoid imposing unnecessary computation burden on the process unit, the IMU sensor is installed on the rover's center of rotation introduced by skid drive mechanism. The IMU, and its installation place are demonstrated in Figure 3.4b.

3.1.2.2 Range Sensor

For the range measurement, a SICK LMS151-10100 is used. The sensor provides a planar scan with a scanning angle of 270 degree, a maximum resolution 0.25 degree, an effective range of 18 meter and a maximum update rate of 50 Hz. The communication between the sensor and the process unit is performed via Ethernet through an inquiry and answer telegram protocol that is available in three methods of Binary, ASCII and HEX. To be able to perform 3D measurement, the LiDAR is integrated with a servo powered gimbal associated with an encoder. The gimbal, communicates with the process unit via Ethernet through the same



(a) Battery system (b) Rover track

Figure 3.5: Configuration of the power and traction system of the Sherpa ground rover (a)The battery of the Sherpa ground rover can be exchanged without powering off the system, 1- battery connection port, 2- battery, 3- the battery space, 4-battery lid. (b) A Track is composed of a pair of wheels, one powered and the other one equipped with an encoder. The wheels are geared by a rubber belt. 1- Encoder wheel, 2- Bogie encoder, 3- Powered wheel, 4- Synthetic rubber belt.

telegram communication protocol as the Laser scanner. The LiDAR and gimbal configuration are illustrated in Figure 3.4a.

3.1.3 Traction and Power

The traction of the vehicle is provided by means of four tracks realizing a skid driven robot. As illustrated in Figure 3.5b, each track includes two wheels, one powered by a 1600 W brushless DC through an “Elmo solo Whistle digital servo drive” while the other is equipped with a MA3 absolute magnetic kit encoder. The two wheels are then geared by a synthetic rubber belt to form a caterpillar track.

To power the tracks, the Sherpa arm and other electronic parts of the rover, a 50 V 100 Ah battery is considered. Apart from the tracks drive, the battery powers the LiDAR, the process unit and network adapter through a set of DC power supplies installed on the rover. To avoid interrupting the high level SHERPA software running on the system, the battery of the rover can be exchanged without switching off the system. This is provided through a second battery connection port and a set of protective Diodes. The power system configuration of the rover is demonstrated in Figure 3.5a.

3.1.4 Internal Network and Process Unit

The Sherpa ground rover is equipped with an internal Ethernet network including a network switch, a Wifi adapter and a Ubiquiti Bullet M2 router that can be configured either as an access point or a network bridge. The internal network provides a communication solution to Ethernet base embedded and external accessories. Moreover, the router provides a way to connect the process unit to the developer's computer or the wireless network of the SHERPA system. The network configuration of the rover can be seen in Figure 3.2b.

The Sherpa ground rover carries two sets of process units. The first unit is a "IV70 BOX PC" equipped with a core i7 intel CPU and two sets of Ethernet adapters. This is considered to be the main process unit for hosting and executing the navigation and SHERPA software. The second unit, on the other hand, is a custom designed computer already programmed by the producer company, Bluebotics, and provides simple navigation and localization functionalities that can be used to test the hardware. Note that due to different wiring configurations simultaneous communication of the both units with the drivers and sensors is not possible. To implement the software developed by this research, we have used only the main process unit. The process units of the rover are demonstrated in Figure 3.2b.

3.2 The Sherpa Robotic Arm

While industrial arms have usually rigid mechanical structures, many research platforms suggest active impedance which allows a safer and smoother interaction with the environment. This is particularly beneficial when the environment is unknown and unpredictable and the system has to offer a certain degree of flexibility. This motivated the design of *Sherpa arm* that is 7 DoF variable stiffness arm with two variable stiffness joints. In the project SHERPAhere is the role of the arm:

- *Reach*: The end effector should be able to reach into the proper position on top of the Wasp. This is a preliminary step to grasping.
- *Grasp*: The end effector should be able to lock into the mechanical interface that is considered on the *wasp* for grasping. This is one of the main motivations for variable stiffness design that endows the system with a certain degree of flexibility to deal with uncertainties *e.g.* non flat terrain profile, variable lighting condition, etc.
- *Place*: Take the grasped *wasp* to the considered docking position on the *Sherpa box*.
- *Dock*: Place the *wasp* on the *Sherpa box*. This is preliminary to the battery exchange operation.
- *Deploy*: Take the *wasp* of the *Sherpa box* and place it on the ground.

Subsequently the main design requirements for the arm can be placed into the following categories:

1. *Kinematics*: The arm needs to reach all required positions, most importantly the docking position and a wide range of reach positions. Due to the required dexterity necessary to manipulate the UAVs in its workspace, a design with a 3-DoF shoulder, 1-DoF elbow, and 3-DoF wrist has been chosen.

2. *Payload*: The arm needs to lift the *wasp*, which has a maximum weight of 2kg.

3. *Interaction and Robustness*: Variable stiffness joints have been chosen to allow the arm to interact with the environment in a physically compliant, versatile, and robust way. The variable stiffness ensures an intrinsically safe interaction and mechanically controllable compliance.

4. *Weight Dimensions*: Because the arm and ground rover need to be transported for the search and rescue operations, the arm needs to fold away inside a small volume and is only allowed to have a light weight below 15kg, while still achieving a reach of 1m.

The Sherpa robotic arm is designed and realized by University of Twente. Therefore, the remainder of this section investigates details of the mechanical design of joints and variable stiffness mechanism.

3.2.1 Kinematic structure

The seven active DoF of the Sherpa arm are provided by “shoulder”, “elbow” and “wrist”. Indeed there are three DoF in the shoulder, one in the elbow, and three in the wrist. The structure of the Sherpa arm is demonstrated in Figure 3.6 while its Denavit-Hartenberg parameters of the arm are summarized in Table 3.1 .

Link #	1	2	3	4	5	6	7	8
$r_i[m]$	0.16	0	0.45	0.12	0	0.35	0	0.13
$\alpha_i[rad]$	$-\pi/2$	$\pi/2$	$-\pi/2$	0	$\pi/2$	$-\pi/2$	$-\pi/2$	$\pi/2$
$d_i[m]$	0.16	0	0.45	0	0	0.35	0	0
θ_i	$q(1)$	$q(2)$	$q(3)$	$q(4)/2$	$q(4)/2$	$q(5)$	$q(6)$	$q(7)$

Table 3.1: DH parameters of the Sherpa arm.

Table 3.2 summarizes some of the specifications of the arm including joint arms and velocities along with static safety factor, joint velocities and joint limits. The static safety factor S is defined as the nominal joint torque divided by the maximum static load while arm is carrying the nominal payload of 2Kg.

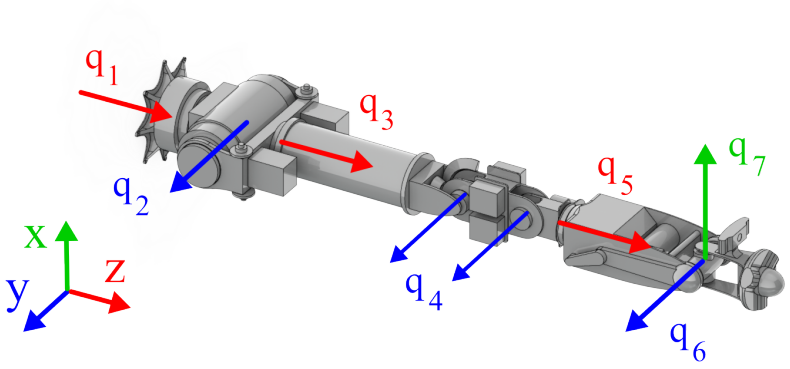


Figure 3.6: Kinematic structure of the Sherpa arm.

Joint	$q(1)$	$q(2)$	$q(3)$	$q(4)$	$q(5)$	$q(6)$	$q(7)$
Torque [Nm]	71	214	214	75	10	8.6	8.6
Static safety	1.8	3.3	3.3	1.4	1.9	1.6	1.6
Velocity $[^\circ/s^2]$	318	174	174	140	252	223	223
Limits $[^\circ]$	$\pm$180	$\pm$90	$\pm$180	$\pm$180	$\pm$180	$\pm$90	$\pm$90

Table 3.2: Mechanical specification of the arm.

3.2.2 Shoulder joint

The shoulder is a 3-DoF joint with a yaw-pitch-yaw configuration illustrated in Figure 3.7. The first DoF is driven by a Kollmorgen RBE 02210-A that is housed in the base of the arm. The second and third DoF are actuated by two differentially coupled ILM 85×13 RoboDrive motors with CSD-32-100 Harmonic Drive transmissions. Furthermore, a tendon and pulley is considered to provide the coupling. As a result of this coupling, given a rigid transmission and a set of actuators a_i the shoulder's joints, q_i can be acquired as follows:

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.5 & -0.5 \\ 0 & 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{3.1}$$

The first variable stiffness joint is located in the joint shoulder which operates based on a lever arm mechanism. As it is illustrated in Figure 3.8 the lever connects a polymer leaf-spring to the output. A hypocycloid gear train is considered to actuate positioning the pivot point q along the lever of length l. This leads to changing lever arm ratio of the spring and ultimately stiffness of the joint. As it is elaborated in [80] deflection of the spring as a result

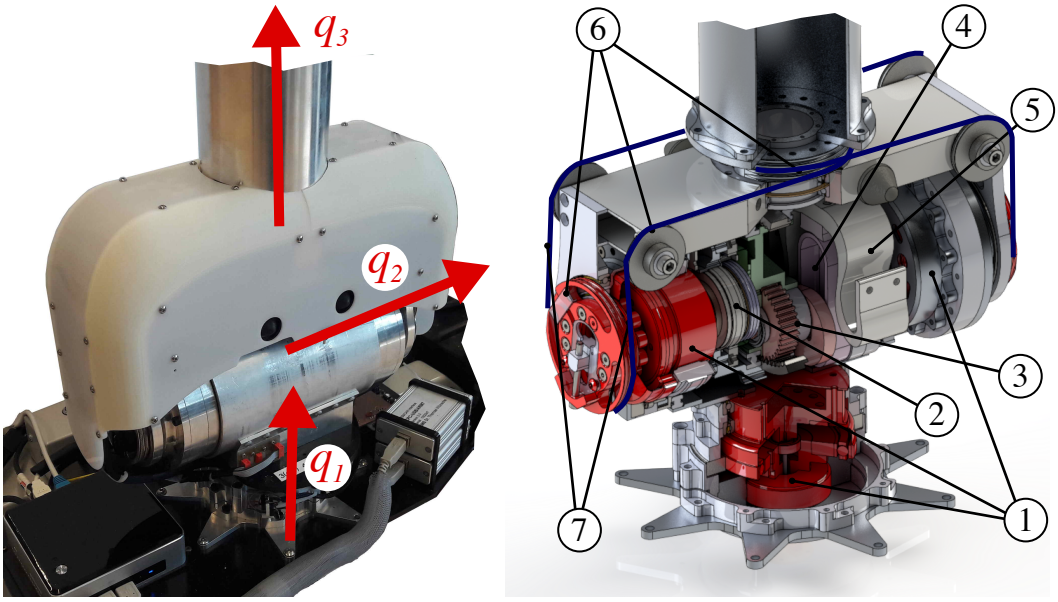


Figure 3.7: The 3-DoF shoulder joint with variable stiffness mechanism showing the DoFs and (1) joint actuators, (2) stiffness adjustment actuators, (3) stiffness adjustment mechanism, (4) lever mechanism with (5) spring, (6) pulleys, and (7) tendons.

of a deflection ϕ in the output can be approximated using the following relation:

$$\tau = \left((\frac{1}{q} - 1)R\right)^2 k_s \phi \tag{3.2}$$

Where R is the output radius and k_s is the spring stiffness. Accordingly the approximate output stiffness K of the joint can be described as follows:

$$K = \frac{\partial \tau}{\partial \phi} = \left((\frac{1}{q} - 1)R\right)^2 k_s \tag{3.3}$$

3.2.3 Elbow joint

In order to enable the arm to fully fold into its *transport* position and to enlarge the workspace, the elbow joint consist of two axes connected by an intermediate link that are driven by a single actuator. The actuator is an ILM 50 $\times$14 Robot Drive motor equipped with a CSD-25-100 Harmonic Drive gearbox and a 1:3 bevel gear transmission. In our representation, elbow is associated with q_4 joint which is the summation of the angle between the lower arm and distal link q_{4a} and the angle between the distal link and the upper arm q_{4b}. Moreover, a tendon mechanism is considered to restrict $q_{4a} = q_{4b} = 0.5q_4$ allowing a large range of motion for this joint. The structure is the elbow is illustrated in Figure 3.9.

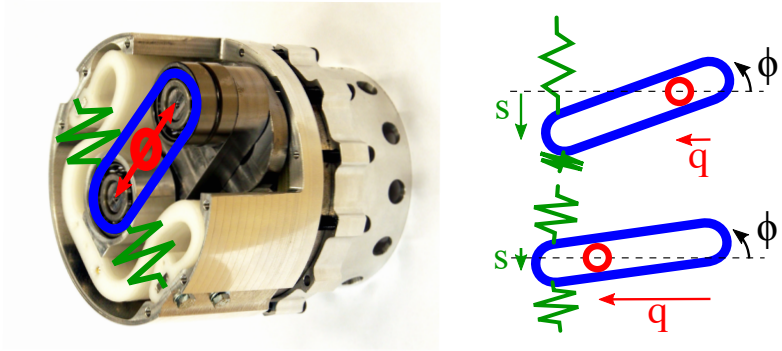


Figure 3.8: The variable stiffness module of the shoulder joint: The lever arm (blue) connects the internal polymer spring (green) to the output, through a variable transmission that is defined by the position of the movable pivot point (red) of the lever. An output deflection ϕ causes a larger spring deflection s in a stiff configuration with a small q.

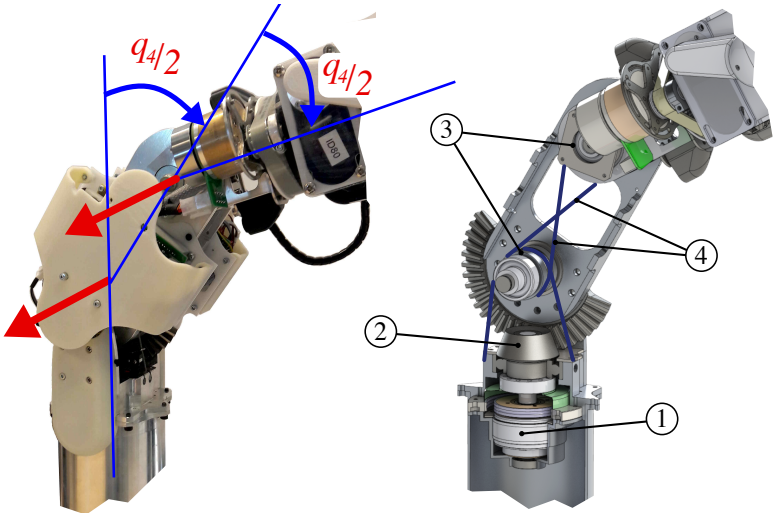


Figure 3.9: The 1-DOF elbow joint showing the coupled DoF q_4 with two rotation axes and a section view with (1) actuator, (2) bevel gear transmission, (3) pulleys, and (4) tendons. The elbow link is directly actuated through the bevel gear, while the motion of the lower arm is constrained by the tendon mechanism.

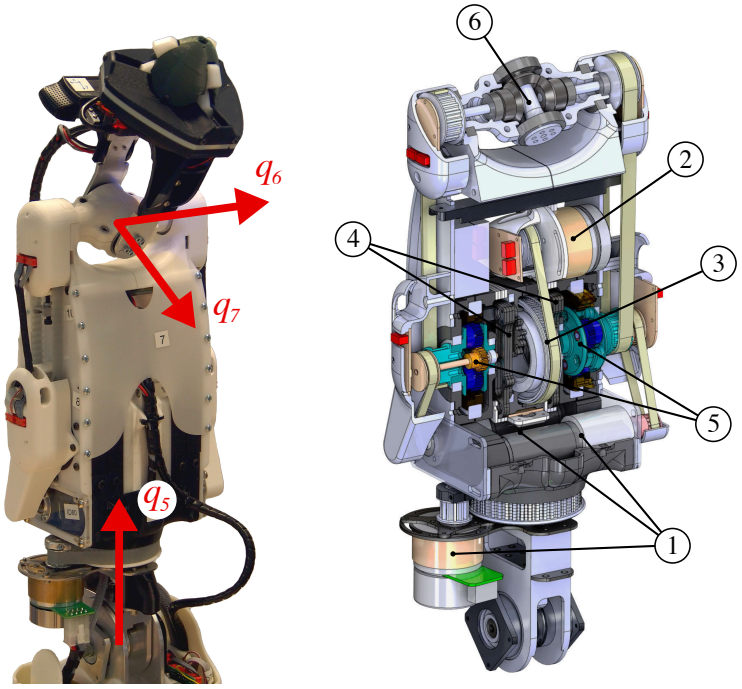


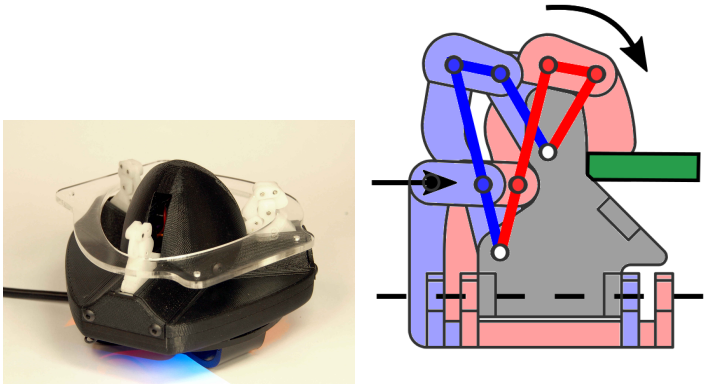
Figure 3.10: The wrist joint and end-effector showing the axes of the 3-DoFs. The section view shows the (1) joint actuators, (2) stiffness adjustment actuator, (3) stiffness adjustment mechanisms, (4) lever mechanism, (5) planetary differential, and (6) bevel differential.

3.2.4 Wrist joint

The wrist joint has a yaw-pitch-yaw configuration with 3 DoF. The joint q_5 is actuated with a Maxon EC45 flat motor and GS45-A gearbox, with an additional 18:90 pulley transmission, while Maxon RE25 motors with GP-32-C gearboxes drive q_6 and q_7 through a differential coupling and a drive that includes a second variable stiffness mechanism. Similar to the shoulder differential coupling (3.1), the wrist differential coupling is described as follows:

$$\begin{bmatrix} q_5 \\ q_6 \\ q_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.5 & -0.5 \\ 0 & 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} a_5 \\ a_6 \\ a_7 \end{bmatrix} \tag{3.4}$$

The main components of the wrist mechanism are highlighted in a section view in Figure 3.10. A planetary gear system is used as 3-port differential in order to incorporate the variable stiffness mechanisms into the drive train as follows; The levers, *i.e.* the output of the variable stiffness modules, are connected to the ring gears and the driving actuators to the sun gear, such that the planet carrier functions as output. The two drive trains are then in turn differentially coupled to each other through a bevel gear mechanism, to actuate the DoFs q_6 and q_7.



(a) Gripper design. (b) Finger mechanism.

Figure 3.11: The SHERPAgripper (Figure 3.11a) and a detailed view of one of the fingers (Figure 3.11b). The fingers are built up of a linkage mechanism mounted on a moving carriage. As the carriage moves the open finger (blue) into contact with the interface (green), the linear forwards motion of the finger's extension is transformed into a rotation of the the distal phalange, which wraps around the interface into the locked position (red).

3.2.5 Gripper

The Sherpa arm is equipped with a custom gripper shown in Figure 3.11a with integrated actuation and electronics, latching onto a light weight interface on the *wasp*. It achieves a secure form of closure through a combination of a passive latching mechanism and a driving dwell mechanism. The gripper engages the interface with three fingers that are extended outwards from the gripper's center through a spiral-shaped cam-mechanism and linear guides. This structure requires only a single actuator. Once the fingers make contact the latching mechanism encloses the interface by means of a series of linkages, as shown in Figure 3.11b. The grasping procedure is facilitated by the gripper's shape, which guides it into the interface, and together with the arm's compliance, ensures a simple pick-up operation that is robust to small misalignments between gripper and interface. The end effector is equipped with a proximity sensor and a Logitech C920 camera for detecting the *wasp* and to enable visual servoing during grasping.

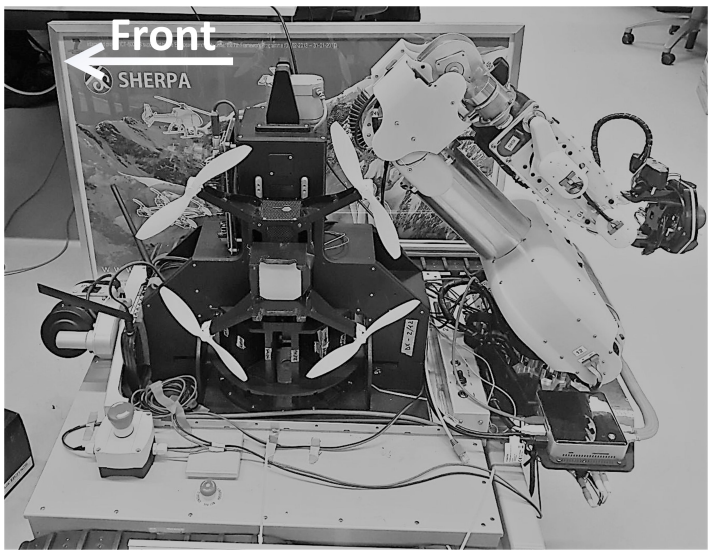
3.3 Sherpa Box

The Sherpa box is designed and realized by ASLATech and is able to operate independent from the Sherpa rover. It is equipped with core i7 NUC intel process unit considered to run the SHERPA software platforms. However, for the wasp battery exchange operation, the Sherpa box has to be mounted on the rover. As mentioned earlier, a mounting space for this purpose has been considered on the rover's chassis. Figure 3.12a demonstrates Sherpa box mounted on the Sherpa ground rover.

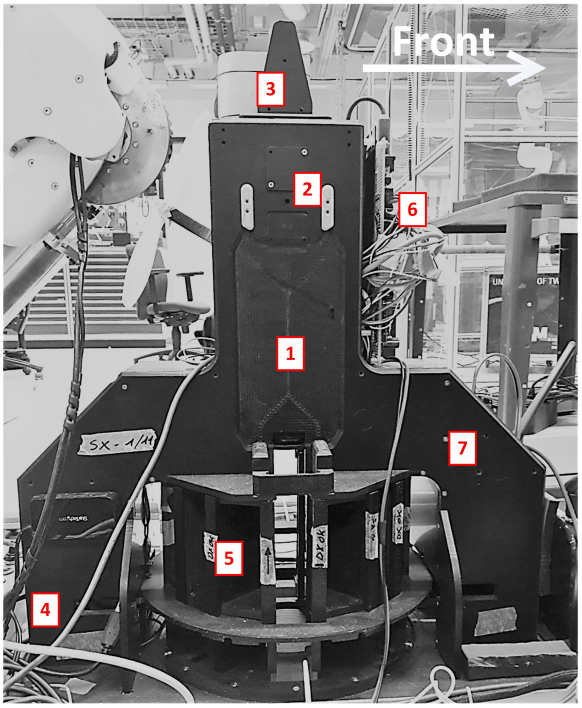
The mechanical design of the box includes four servo motors to actuate the wasp latching mechanism, the battery latching mechanism, the battery lifter and battery magazine revolver. The servos are controlled through an Arduino controller that communicates with the main processor through a USB connection. Furthermore, to facilitate the communication in the ROS environment, the mavros protocol is adapted. An overview of the Sherpa box hardware is presented in Figure 3.12b.

The main mechanically functionality of the Sherpa box is in the wasp battery exchange operation. The battery exchange mechanism itself consists of a linear actuator that slides the battery carriages from the *wasp* into a revolving battery magazine at the base of the Sherpa box and vice versa. A sectional view of the Sherpa box and the *wasp* is shown in Figure 3.13, displaying the components of the mechanism. Since the *wasps* do not have to land on the Sherpa box, the battery exchange mechanism can be arranged vertically. This allows the box to service two *wasps* simultaneously. The *wasp* battery exchange procedure is presented as follows:

1. For docking the *wasp* on the Sherpa box, the *wasp* is initially placed on two small guide rails, before it is moved into its final docking position, guided by the rails and a short funnel.
2. Once the *wasp* has been placed correctly, a switch registers that the *wasp* is *docked*, and two small clamps *lock* it securely in place.
3. A tongue then extends from the toolhead at the end of the linear actuator, and engages a notch in the battery carriage, in order to move it downwards into a free slot of the battery magazine.
4. The battery slots of the magazine are aligned with the battery compartment of the *wasp* by means of a Geneva mechanism, which rotates the magazine in discrete steps ensuring proper alignment, after the tongue has disengaged the empty battery.
5. When the charged battery has been aligned, it is engaged by the tongue, and placed into the battery bay of the *wasp*, where it securely engages the electrical connectors and locks into place.
6. The replenished *wasp* is then ready to be *released* by the Sherpa box and to continue its mission.

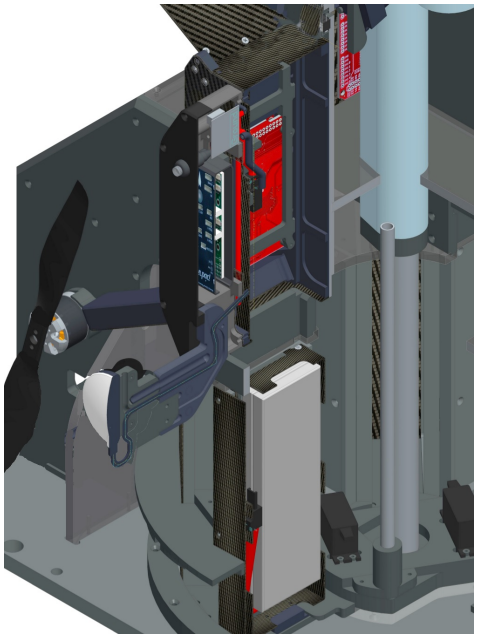


(a) Sherpa Box mounted on the rover

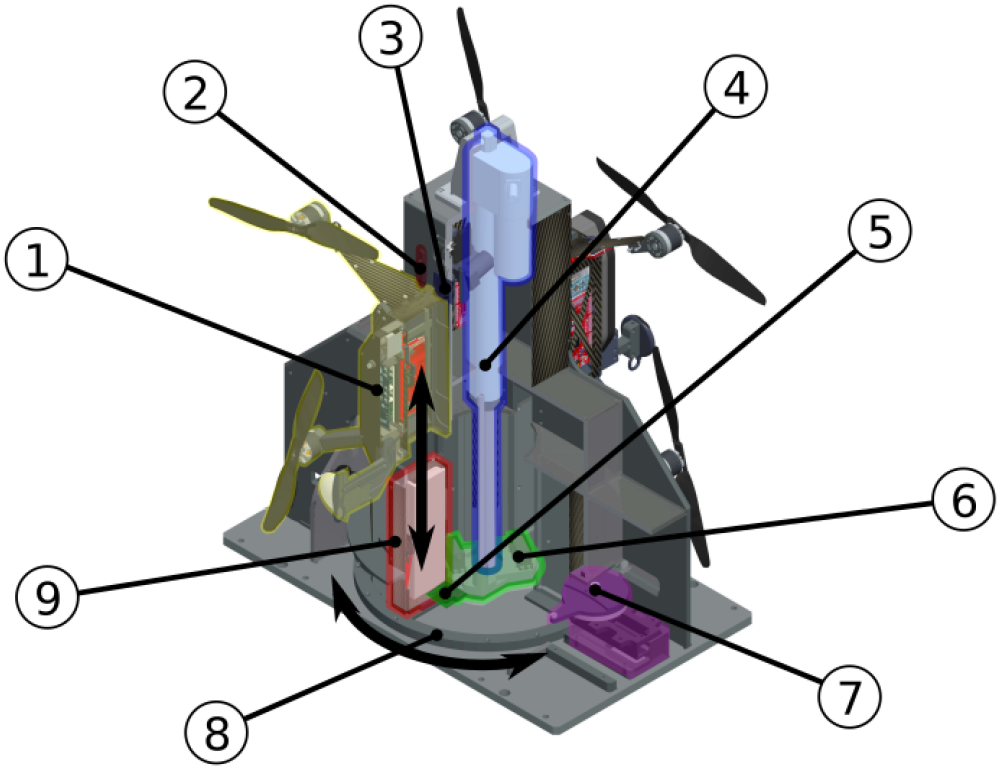


(b) Sherpa Box Hardware

Figure 3.12: Sherpa box set-up overview. (a) shows the Sherpa box mounted on the considered mounting space on the ground rover (b) A close-up of the hardware, 1- Wasp spot, 2- Wasp latching mechanism, 3- Battery lifter servo, 4- Battery magazine revolver servo, 5- Battery magazine, 6- Arduino controller, 7- Sherpa box body.



(a) Wasp battery close up.



(b) Battery exchange mechanism.

Figure 3.13: A partial section view of the *Sherpa box* and docked *wasps* showing the battery exchange mechanism. The *wasp* (1) is locked to the *Sherpa box* with the clamps (2) when the switch (3) is engaged. To remove the battery (9) from the *wasp* a "tongue" (5) engages the battery, which is pulled downwards into the battery magazine (8) when the linear actuator (4) moves the toolhead (6). To exchange the battery, the magazine is rotated by the actuator (7) which drives a Geneva mechanism. The battery exchange mechanism then places a charged battery into the *wasp*.

Chapter 4

Integration of the Ground Rover with the Robotic Arm

In the previous chapter we have presented the mechanical design and specification of the Sherpa Arm. In this chapter, we investigate integrating the Arm with the Donkey rover from the mathematical and system point of view. We therefore present kinematics of the rover, the arm and combination of the two, *i.e.* mobile manipulator. Subsequently we investigate kinematic control of the arm and mobile manipulator. Finally the compliance of the arm introduced by variable stiffness joints will be analysed.

4.1 Kinematic modelling

Identification of the system is preliminary step to the control problem. This section studies kinematics of the arm, the rover and the Ground Rover and Arm (GRA) which is the integrated system.

Before proceeding with the mathematical modelling, we define a notation that will be used throughout the entire chapter. It is well known that the pose of a floating object in space can be described by 6 variables, 3 for the position and 3 for the orientation. We therefore describe a pose by a 6×1 vector, *i.e.* $\mathcal{X}$, composed of a 3×1 position vector, *i.e.* ρ, and a 3×1 orientation vector, *i.e.* ν. Moreover, the frame with respect to which the pose is described is presented on the upper left side of the pose.

$$\mathcal{X} = [\rho\ \nu]^\intercal = [x\ y\ z \mid \theta\ \phi\ \psi]^\intercal$$
$${}^{A}\mathcal{X}_{ef} \text{ : pose of end effector with respect to frame A}$$

Moreover, some frames of interest that we will talk about throughout this chapter are to be introduced. "End effector" frame that is a frame attached to the base of end effector and is identifying by ef. "Arm base" frame, a frame attached to the base of the arm, *i.e.* base

of the *shoulder*, and represented by A. "Rover base" frame that is originated on the rover's differential rotation axes and on the robot's surface that is denoted by R. Finally "Inertial frame", also knows as "World", is a fixed frame in the environment denoted by IF.

4.1.1 Arm Kinematics

In the approach phase, the arm is used in the stiff mode as a compliant arm does not add any strategic advantage to the approach problem. Moreover, this way we decrease the complexity of system by fixing two extra DoF introduced by the compliance.

The Jacobian of the arm can be calculated by means of differentiating the forward kinematics of the system. The position of the end effector with respect to the arm base can be described as follows:

$$^{A}\mathcal{X}_{ef} = f(\theta_1, \theta_2, ..., \theta_n) \tag{4.1}$$

Where θ_i refers to the i^{th} joint angle and n is the DoF of the arm, in our case is 7.

Differentiating the forward kinematics of (4.1) yields:

$$\begin{aligned} \delta^{A}\mathcal{X}_{ef} &= \frac{\partial f}{\partial \theta}\delta\theta : \\ \delta x_i &= \sum_{j=1}^{n} \frac{\partial f_1}{\partial \theta_j}\delta\theta_j, \quad i = 1, \cdots, 6. \end{aligned} \tag{4.2}$$

Representing joint variables by a vector $q_A = [\theta_i \ \cdots \ \theta_n]^{\intercal}$ we can rewrite (4.2) as follows:

$$^{A}\dot{\mathcal{X}}_{ef} = J_{Aef} * \dot{q}_A \tag{4.3}$$

Where J_{Aef} is the Jacobian matrix that relates velocity of the end effector with respect to the arm base frame.

If the rover pose in the inertial frame is known, the pose of *Arm base* in the inertial frame is known since the *Arm base-rover* is fixed known pose. In this case, velocity of the end effector frame with respect to the inertial frame can be described by the following Jacobian matrix:

$$^{IF}\dot{\mathcal{X}}_{ef} = J_{IFef} * \dot{q}_A \tag{4.4}$$

where

$$J_{IFef} = \begin{bmatrix} {}^{IF}_{A}R & 0_{3\times3} \\ 0_{3\times3} & {}^{IF}_{A}R \end{bmatrix}_{6\times6} * J_{Aef} \tag{4.5}$$

and ${}^{IF}_{A}R$ is a 3D rotation matrix presenting the rotation of the Arm base frame with respect to inertial frame.

4.1.2 Rover Kinematics

Although in the context of SHERPAthe navigation strategy has to cope with a rough terrain, it is reasonable to assume a flat navigation in the very last phase of approach where the *wasp* can be reached by small displacement of the platform. Making this assumption, the kinematics of a differential driven robot subjected to nonholonomic constraints driving on a flat surface can be described as follows:

$$\begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\psi}_R \end{bmatrix} = \begin{bmatrix} \cos\psi & 0 \\ \sin\psi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{D} \\ \dot{\psi} \end{bmatrix} \Rightarrow {}^{IF}\dot{\mathcal{X}}_R = J_{IFR} * \dot{q}_R \tag{4.6}$$

Where ψ_R refers rover's orientation, *i.e.* yaw, and $\dot{D}$ and $\dot{\psi}$ are linear and angular velocities of the rover *i.e.* inputs of the system.

Equation (4.6) presents the difficulty of controlling a nonholonomic robot. As can be seen, the matrix J_{IFR} loses its rank for a certain heading angles *i.e.* $k\pi, k \in \mathbb{Z}$. In fact, this is no surprise since differential and car like robots cannot slide to their sides. Thus, to accomplish trajectory following "look ahead" strategy can be deployed. In this method, a new frame is introduced with a bias distance b in front of the robot and the controller reject the disturbance introduced by error between origin of b and the desired position trajectory. This combination is demonstrated in Figure 4.1a. Subsequently, the kinematics of the point b can be described as follows:

$$\begin{bmatrix} \dot{x}_b \\ \dot{y}_b \\ \dot{\psi}_b \end{bmatrix} = \begin{bmatrix} \cos\psi & -b.\sin\psi \\ \sin\psi & b.\cos\psi \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{D} \\ \dot{\psi} \end{bmatrix} \Rightarrow {}^{IF}\dot{\mathcal{X}}_b = J_{IFb} * \dot{q}_R \tag{4.7}$$

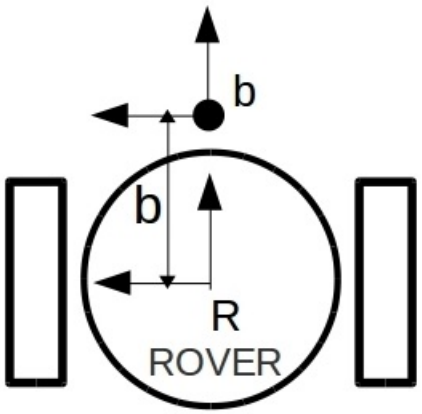
Therefore, unlike the rover frame, there is an element along y axis in the b frame.

$${}^{R}\dot{\mathcal{X}}_R = \begin{bmatrix} \dot{D} \\ 0 \\ \dot{\psi} \end{bmatrix} \Rightarrow {}^{b}\dot{\mathcal{X}}_b = \begin{bmatrix} \dot{D} \\ b\dot{\psi} \\ \dot{\psi} \end{bmatrix} \tag{4.8}$$

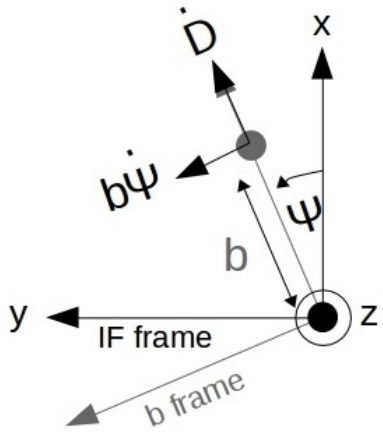
4.1.3 Mobile Manipulator Jacobian

A interesting implementation of mobile manipulator control system is able to track 1) the end effector trajectory and 2) rover trajectory. This allows the system to plan for the rover and arm separately *e.g.* arm can reach the *wasp* while adjusting pose of the rover. Accordingly we seek a Jacobian matrix that can relate composed actuator velocity vector of the *GRA* to its composed task velocity vector:

$${}^{IF}\dot{\mathcal{X}}_{GRA} = J_{GRA} * \dot{q}_{GRA} \tag{4.9}$$



(a) b frame introduction.



(b) Speed element on the b point.

Figure 4.1: look ahead change of coordinates

where

$$ {}^{IF}\dot{\mathcal{X}}_{GRA} = \begin{bmatrix} {}^{IF}\dot{\mathcal{X}}_{ef} \\ {}^{IF}\dot{\mathcal{X}}_{b} \end{bmatrix} \quad , \quad \dot{q}_{GRA} = \begin{bmatrix} \dot{q}_A \\ \dot{q}_R \end{bmatrix} $$

In order to find the J_{GRA} one can combine kinematics of the arm (4.4) and kinematics of the rover (4.7) as follows:

$$ \begin{bmatrix} {}^{IF}\dot{\mathcal{X}}_{ef} \\ {}^{IF}\dot{\mathcal{X}}_{b} \end{bmatrix}_{9\times1} = \begin{bmatrix} J_{IFef\ 6\times n} & J_{c\ 6\times2} \\ [0]_{3\times n} & J_{IFb\ 3\times2} \end{bmatrix} \begin{bmatrix} \dot{q}_{A\ n\times1} \\ \dot{q}_{R\ 2\times1} \end{bmatrix} \tag{4.10} $$

It is obvious that speed of the rover is independent from the joint angles. On the contrary, end effector task velocity vector is coupled with the rover inputs by a Jacobian matrix J_c, of the size 6×1, yet to be found. To find the Jacobian, we can consider the contributions of arm and rover inputs to the end effector task velocity vector separately:

$$ {}^{IF}\dot{\mathcal{X}}_{ef} = {}^{IF}\ \dot{\mathcal{X}}_{efA} + {}^{IF}\ \dot{\mathcal{X}}_{efR} \tag{4.11} $$

The elements of ${}^{IF}\dot{\mathcal{X}}_{ef}$ affected by the rover controls are linear speeds along x and y axis

and angular speed around z axis:

$$ {}^{IF}\mathcal{X}_{efR} = \begin{bmatrix} x_{efR} \\ y_{efR} \\ 0 \\ 0 \\ 0 \\ \psi_{efR} \end{bmatrix} $$

And from Figure 4.2, assuming that the arm base has the same orientation as rover, one can write:

$$ \begin{aligned} x_a &= x_R + {}^{R}x_{ef}\cos\psi_R - {}^{R}y_{ef}\sin\psi_R \\ y_a &= y_R + {}^{R}x_{ef}\sin\psi_R + {}^{R}y_{ef}\cos\psi_R \\ \psi_a &= \psi_R \\ &\quad \tfrac{d}{dt} \Rightarrow \\ \dot{x}_a &= \dot{x}_R - \dot{\psi}_R\,{}^{R}x_{ef}\sin\psi_R - \dot{\psi}_R\,{}^{R}y_{ef}\cos\psi_R \\ \dot{y}_a &= \dot{y}_R + \dot{\psi}_R\,{}^{R}x_{ef}\cos\psi_R - \dot{\psi}_R\,{}^{R}y_{ef}\sin\psi_R \\ \dot{\psi}_a &= \dot{\psi}_R \end{aligned} \tag{4.12} $$

Writing the result of derivatives in the matrix form yields:

$$ {}^{IF}\dot{\tilde{\mathcal{X}}}_{efR} = \begin{bmatrix} 1 & 0 & -({}^{R}x_{ef}\sin\psi + {}^{R}y_{ef}\cos\psi) \\ 0 & 1 & {}^{R}x_{ef}\cos\psi + {}^{R}y_{ef}\sin\psi \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\psi}_R \end{bmatrix} \tag{4.13} $$

Where ${}^{IF}\dot{\tilde{\mathcal{X}}}_{efR}$ is shorten form of ${}^{IF}\dot{\mathcal{X}}_{efR}$, *i.e.* eliminating zero rows. Substituting the rover kinematics from (4.6) in the above equation we get:

$$ \begin{aligned} {}^{IF}\dot{\tilde{\mathcal{X}}}_{efR} &= \begin{bmatrix} 1 & 0 & -({}^{R}x_{ef}\sin\psi + {}^{R}y_{ef}\cos\psi) \\ 0 & 1 & {}^{R}x_{ef}\cos\psi + {}^{R}y_{ef}\sin\psi \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\psi & 0 \\ \sin\psi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{D} \\ \dot{\psi} \end{bmatrix} \\ &= \begin{bmatrix} \cos\psi & -({}^{R}x_{ef}\sin\psi + {}^{R}y_{ef}\cos\psi) \\ \sin\psi & {}^{R}x_{ef}\cos\psi + {}^{R}y_{ef}\sin\psi \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{D} \\ \dot{\psi} \end{bmatrix} \end{aligned} \tag{4.14} $$

From Figure 4.2, one can see the *arm base* is located on the x axes of the rover base with a distance of a from the origin of the *rover base*. Furthermore, the pose of the end effector with respect to the arm base is available thanks to the joint encoders. Thus, it is more convenient to use the end effector position in the arm coordinates rather than rover coordinates *i.e.* :

$$ \begin{bmatrix} {}^{R}x_ef \\ {}^{R}y_ef \end{bmatrix} = \begin{bmatrix} {}^{A}x_ef + a \\ {}^{A}y_ef \end{bmatrix} $$

Finally we can describe ${}^{IF}\dot{\mathcal{X}}_{efR}$ by adding the proper dimension to ${}^{IF}\dot{\tilde{\mathcal{X}}}_{efR}$ which get us to

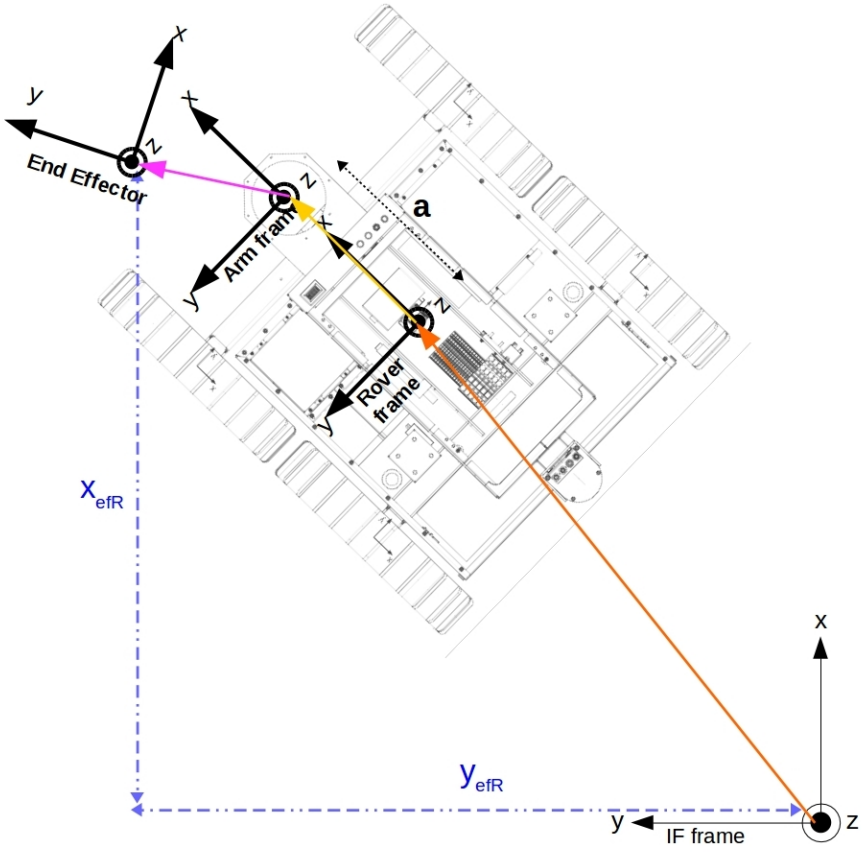


Figure 4.2: Vector representation of the arm base with respect to the rover.

the Jacobian matrix J_c:

$$ {}^{IF}\dot{\mathcal{X}}_{efR} = \begin{bmatrix} \cos\psi & -((a + {}^{A}x_{ef})\sin\psi + {}^{A}y_{ef}\cos\psi) \\ \sin\psi & (a + {}^{A}x_{ef})\cos\psi + {}^{A}y_{ef}\sin\psi \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{D} \\ \dot{\psi} \end{bmatrix} = J_c * \dot{q}_R \tag{4.15} $$

4.2 Kinematic Control of Mobile Manipulator

In the previous section the kinematics of the system was presented. Either it is a mobile or fixed manipulator, it can be expressed in the general form of:

$$ J * \dot{q} = \dot{\mathcal{X}} \tag{4.16} $$

In this section, we attend to the control problem defined as finding actuator velocities that results in the desired task space velocities.

The system understudy is a redundant robotic system *i.e.* the task velocity vector has a lower dimension than actuator velocity vector. As a result, to achieve a desired task velocity vector, *i.e.* ${}^{IF}\dot{\mathcal{X}}_{ef}$, there are more than one valid solution in the space of actuator velocity vector, *i.e.* $\dot{q}$. In other words, the Jacobian matrix has more columns than rows and therefore a non zero null space. This gives us the ability to design a controller that regulates the output in the operational space while performing some optimization on the null space of the system.

In order to find actuator velocity vector from (4.16), Inverse kinematics can be used *i.e.* the Jacobian matrix should be inversed. As mentioned earlier, the null space of (4.16) is not zero allowing for null space optimization while inversion. The null space optimization can be used to address "singularities" and "join limits exceedance" issues that are studied in this section.

4.2.1 Least Norm Solution and Pseudo-Inverse of the Jacobian

From the basics of the linear algebra, one can suggest a Least Norm solution to the undetermined equation of (4.16) using the pseudo-Inverse of the Jacobian matrix. So we get:

$$\dot{q} = J^{\dagger} * \dot{\mathcal{X}} \tag{4.17}$$

where

$$J^{\dagger} = J^{\intercal} * (J * J^{\intercal})^{-1} \tag{4.18}$$

Equation (4.18) is a solution to the following optimization problem:

$$\dot{q}_{LN} = \arg\min_{\dot{q}} ||\dot{q}||, \quad \textit{Subject to: } J * \dot{q} = \dot{\mathcal{X}} \tag{4.19}$$

It is therefore obvious that (4.18) suggests a solution with minimal actuator velocities. However this solution is only optimal as long as the arm is not close to singular configurations.

4.2.2 Singularity Robust Inversion

Intuitively, in the vicinity of the singular configurations, effect of singular joints on the task velocity vector becomes less and less until in the singular configuration the task velocity vector becomes totally independent from the singular joints. Therefore, in the vicinity of a singular configurations, very high joint velocities are needed to produce the desired effect on the task velocity vector leading to undesirable performance and instability. From the mathematical point of view, singularity is interpreted by the Jacobian Matrix loosing rank.

The best way to cope with singularity is avoiding the singular configuration. For a redundant robotic arm, "singular Robust inversion" can be deployed to alleviate the singularity

problem. First, we need to have an indicator reflecting the distance from singularities. This can be done by means of "manipulability measure". Consider the set of all possible velocity task vector that can be achieved by a set of joint velocity vectors satisfying:

$$\sqrt{q_{A1}^2 + \ldots + q_{An}^2} \leq 1 \tag{4.20}$$

This set is an ellipsoid in the m-dimensional Euclidean space representing the *manipulability*. It is obvious that the smaller is the ellipsoid, the closer is the configuration to the singularity. Mathematically, the manipulability ellipsoid can be found by taking a Singular Value Decomposition (SVD) of the Jacobian matrix:

$$J = U\Sigma V^\intercal \tag{4.21}$$

where U is of dimension of $m \times m$ and V is of dimension of $n \times n$, and the two are orthogonal matrices. Furthermore, Σ is a size $m \times n$ matrix containing a $m \times m$ diagonal matrix and $n - m$ zero columns:

$$\Sigma = \left[\begin{array}{ccc|ccc} \sigma_1 & \ldots & 0 & 0 & \ldots & 0 \\ \vdots & \ddots & \vdots & 0 & \ldots & 0 \\ 0 & \ldots & \sigma_m & 0 & \ldots & 0 \end{array}\right] \tag{4.22}$$

The scalar values of σ_i are called singular values of the Jacobian matrix. The manipulability ellipsoid has its principal axes along $\sigma_i u_i$ vectors where u_i is the i^{th} column of matrix U. Finally the manipulability measure can be provided by the volume of the ellipsoid that is:

$$W = \prod_{i=1}^{n} \sigma_i \tag{4.23}$$

The manipulability measure can also be calculated without having to calculate the SVD of the Jacobian using [81]:

$$W = \sqrt{det(J * J^\intercal)} \tag{4.24}$$

Now we have a tool to measure the distance from the singular configuration, we can attend to the *singular robust inversion* method. It can be proven that the following pseudo inversion is a solution to the undetermined equation (4.16) [82]:

$$J^\dagger = J^\intercal * (J * J^\intercal + \beta I_m)^{-1} \tag{4.25}$$

where I_m is an identity matrix of dimension $m \times m$ and β is a scalar factor. At the singular configuration, β is equal to β_s, a tunable value adjusting a safe distance from the singularity. Whereas, in the vicinity of the singular configuration, β has to decrease until a certain margin, *i.e.* W_m, where the configuration is far enough from the singularity and the least norm solution of (4.17) can be used safely. We can then define the β factor as follows:

$$\beta = \begin{cases} \beta_s \left(1 - \dfrac{W}{W_m}\right)^2 & \text{for } W < W_m \\ 0 & \text{for } W \geq W_m \end{cases} \tag{4.26}$$

4.2.3 Weighted Least-Norm Solution

It is important to have control over the joints and task velocity element-wise *e.g.* a join is in its upper/downer limit. In another example the application may put a constraint on movement of the end effect in a specific direction being imposed to avoid an obstacle in the scene or safely interact with another object that has to be grasped. In this case, the least norm solution (4.17) can be elaborated into weighted least norm solution by means of two weighting matrices. In this section we investigate general weighted least norm solution. We will then proceed to special case of joint limits.

Let us define two symmetric, diagonal, positive definite matrices of W_q and W_x. Furthermore, we define the following transformation:

$$\begin{aligned} J_{wt} &= W_x^{\frac{1}{2}} J\, W_q^{\frac{1}{2}} \\ \dot{\mathcal{X}}_{wt} &= W_x^{\frac{1}{2}} \dot{\mathcal{X}}_{wt} \\ q_{wt} &= W_q^{\frac{1}{2}} \dot{q} \end{aligned} \tag{4.27}$$

As a result the least norm solution to (4.18) can be rewritten as:

$$\dot{q}_{wt} = J_{wt}^{\intercal}\, (J_{wt}\, J_{wt}^{\intercal})^{-1}\, \dot{\mathcal{X}}_{wt} \tag{4.28}$$

Substituting (4.27) in (4.28) and some mathematical elaboration raises:

$$\dot{q} = W_q^{-1}\, J^{\intercal}\, W_x^{\frac{1}{2}} \left(W_x^{\frac{1}{2}}\, J\, W_q^{-1}\, J^{\intercal}\, W_x^{\frac{1}{2}} \right)^{-1} W_x^{\frac{1}{2}} \dot{\mathcal{X}} \tag{4.29}$$

(4.29) gives weighted least norm solution to (4.16). Elements of diagonal matrices W_x and W_q can be used to tune the manoeuvrability of the associated parameter. For example 1 causes no limits on the associated joint/task velocity while smaller values leads to limiting the movement.

In practice, in the design of control architecture for a manipulator, avoiding join limits always has be taken into account. As mentioned earlier, weighted least norm solution can be deployed to that purpose. For the sake of simplicity one can rewrite (4.29) assuming the task velocity weighting matrix to be identity:

$$\dot{q} = W_q^{-1}\, J^{\intercal} \left(J\, W_q^{-1}\, J^{\intercal} \right)^{-1} \dot{\mathcal{X}} \tag{4.30}$$

Before proceeding with the joint weighting matrix, it is important to define a criterion to have an understanding of how close the joint is to its limit. To this purpose, [83] suggests the following criterion:

$$C(q) = \sum_{i=1}^{n} \frac{(q_{Ui} - q_{Li})^2}{4(q_{Ui} - q_i)(q_i - q_{Li})} \tag{4.31}$$

Where q_i, q_{Ui} and q_{Li} are respectively current angle, upper and lower joint limits of the joint i.

(4.31) suggests that when all the joints are in the mid point of their range, *i.e.* $q_i = (q_{Ui} - q_{Li})/2$, contribution of each joint to the criterion is unity. However, the criterion returns a larger number as the joints get closer to their limits. We can then use the criterion $C(q)$ to tune the joint weighting matrix W_q as follows:

$$w_{qi} = 1 + \left| \frac{\partial C(q)}{\partial q_i} \right| = 1 + \frac{(q_{Ui} - q_{Li})^2 (2q_i - q_{Li} - q_{Ui})}{4(q_{Ui} - q_i)^2 (q_i - q_{Li})^2} \tag{4.32}$$

Where w_{qi} are diagonal elements of W_q. It can be seen that $|\frac{\partial C(q)}{\partial q_i}|$ is small when q_i is close to its range middle point. On the contrary, this values grows significantly as the join i get closer to its limits. As a result, a configuration that requires joint i be in close to its limits causes the corresponding weight w_{qi} to be infinity in W_q and, subsequently, zero in W_q^{-1}. Referring back to (4.30), the joint velocity of the joint i approaches zero and then the joint is locked. Yet, it is not desirable to have the joint completely locked in the joint limit as the joint should be able to move back to the range middle point. We, therefore, modify (4.32) as follows:

$$w_{qi} = \begin{cases} 1 + \left| \dfrac{\partial C(q)}{\partial q_i} \right|, & \Delta \left| \frac{\partial C(q)}{\partial q_i} \right| \geq 0 \\ 1, & \Delta \left| \frac{\partial C(q)}{\partial q_i} \right| < 0 \end{cases} \tag{4.33}$$

4.3 Analysis of the Arm Compliance

When there is a need for physical interaction of robot, it is important to have an end effector that is flexible. Obviously, this is because applying too much pressure can cause mechanical damage. For the application understudy, this is even more crucial since there are two robots involved in the physical interaction. Indeed, the arm's gripper has to lock into drone's interface and excessive force may damage the both devices. As explained in section 3.2, to provide a certain level of mechanical compliance, there are two variable stiffness joints considered for the Sherpa arm. The mechanical design of the variable stiffness actuators of shoulder and elbow joints has been explained in sections 3.2.2 and 3.2.5 respectively. In this section, we study the effect of compliance from mathematical point of view.

The compliance of the arm's end effector can be described by the compliance matrix. In fact, infinitesimal twist of the end effector is linearly mapped from its wrench, *i.e.* W, by means of the compliance matrix, *i.e.* C, as follows:

$$\begin{bmatrix} \delta\theta \\ \delta p \end{bmatrix} = \begin{bmatrix} C_o & C_c \\ C_c & C_t \end{bmatrix} \begin{bmatrix} m \\ f \end{bmatrix} = C * W \tag{4.34}$$

Where θ, p, m and f are respectively orientation, position, moment and force vectors. Moreover, C_o, C_t and C_c are respectively rotational and translational and coupling compliance matrices.

Given the joint compliance matrix to be defined as $C_q = -\frac{\partial q}{\partial \tau}$ and τ joint torques, the compliance matrix can also be expressed by means of the Jacobian as follows:

$$C = J(q)C_q J^{\intercal}(q) \tag{4.35}$$

Expectedly, (4.35) implies the compliance matrix is also function of arm pose, *i.e.* q. Consequently, we analyse the compliance of the arm In two poses of special interest for the Sherpa battery exchange experience,(see section 1.3.3).

Grasp: Grasping the *wasp* is a critical pose to be studied. In this pose, the arm is already reached on top of drone's interface. It is therefore crucial for the gripper to be soft so the risk of damaging either gripper or the drone interface is minimized. On the other hand, a certain degree of stiffness is necessary to make the grasping possible.

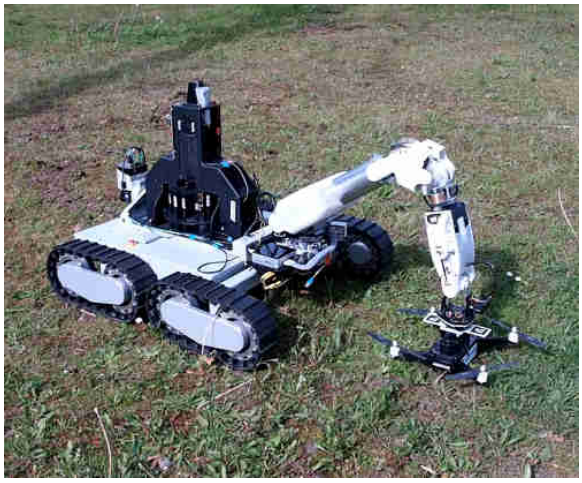
Dock: The *wasp* must be docked to the *Sherpa box* after grasping step. This pose implies more of a challenge as the arm is loaded with the *wasp* with a payload of $2Kg$. Consequently, as soon as the stiffness is decreased, gravity pulls down the *wasp*. Note that in this pose, the drone is rather close to the *sherpa box* and consequently there is a risk damage for both of devices especially wasp's propellers. Yet, a certain degree of compliance is desirable to properly place the drone to the docking station on the Sherpa box.

The translational end effector compliance can be visualized as an ellipsoid obtainable through singular value decomposition of C_t *i.e.* :

$$C_t = U\Sigma V^* \tag{4.36}$$

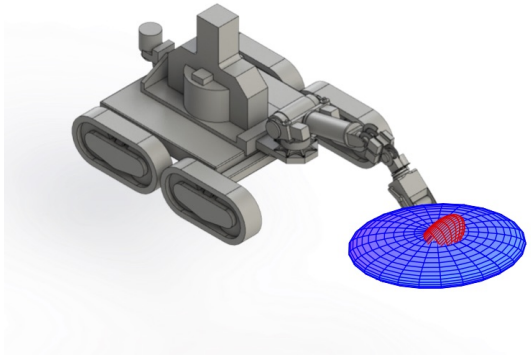
Where the singular values (in Σ) are lengths of semi-axes of the ellipsoid and the columns of U matrix represents orientation of the axes of ellipsoid.

The compliance of the end effector in the poses of interest are demonstrated in Figure 4.3. An interesting feature of the arm's design is the ability to be compliant in one direction while rigid in the other directions. This is especially useful when the arm is loaded *i.e. dock pose*. In this case the end effector's compliance can be tuned to be rigid against the gravity force while a certain degree of compliance can be considered in the horizontal axis to facilitate the docking mechanism.

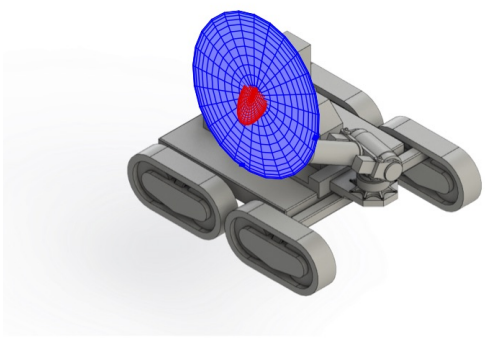


(a) Grasp demo.

(b) Dock demo.



(c) Grasp pose.



(d) Dock pose.

Figure 4.3: End effector's compliance in special poses. Two compliance ellipsoids corresponding the two variable stiffness joints are presented in each pose. In the presented cases the shoulder's and wrist's compliance have been varied, respectively, in the range of $[0.001^{rad/Nm}\ 0.005^{rad/Nm}]$ and $[0.04^{rad/Nm}\ 0.25^{rad/Nm}]$.

Chapter 5

Implementation and Results

This chapter studies detailed design of the system and algorithms that have been developed to perform the *human leashing* and *battery exchange* benchmarks.

We first consider addressing the human leashing using only the rover. To this purpose, we propose a control system able to guarantee tracking of the target in the sensor range. The result of proposed algorithms is evaluated using the simulation and experiment.

Target leashing while rough terrain navigation is more complicated, yet possible taking the advantage of Sherpa arm mounted on the rover. We, therefore, propose a cooperative framework in which arm takes the responsibility of following the target while the rover navigates to its goal optimizing chassis terrain interaction. Moreover, the two robot are coupled using a cooperative framework to provide a superior performance that is evaluated by experiment.

Finally, the battery exchange experiment will be elaborated. In this case, the high level commands come from the *delegation framework* presented in section 1.2.1. The result of battery exchange operation is evaluated using the experiment.

5.1 Human following using the Rover

This section presents a non linear control of a vision-based human following robot. The set up is given by the *rover* equipped with a sensor which provides the human-robot pose in a limited sensory range. The Sensor Operation Field of View (SOFV) is, therefore, treated as a state constraint in the control architecture. An example of SOFV is depicted in Figure 5.1. Furthermore, it is assumed that having the *evader*[1] out of SOFV is not tolerable and the controller must ensure an appropriate human-robot relative pose to avoid losing the track of

[1]The person to follow, followee

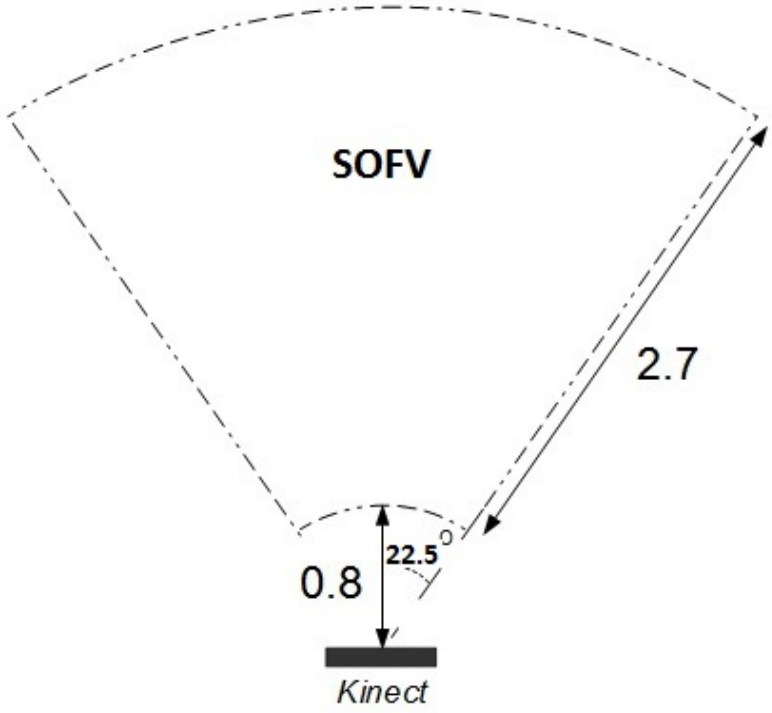


Figure 5.1: Sensor operation field of view of an example vision sensor (Kinect), the measurements are in meters.

evader. Moreover, the actuators limitations have to be considered as control input constraints and they have to be taken into account in the definition of the control law.

Consequently, we aim to design a saturated nonlinear controller to accomplish human following while satisfying the state and input constraints. The upper bound on the control action is dynamically adapted, based on human-robot relative pose, to fully exploit the available actuation power. Given the evader's speed is bounded, the proposed control structure guarantees that the maximum tracking error can be arbitrarily bounded. Notice, as seen in Figure 5.1, we assume that the human recognition is performed via a Kinect[2] sensor, although the analysis of this part are easily expendable to other types of sensor.

5.1.1 Problem formulation

In the analysis that is about to be presented, we consider the inertial and body frames as follows; The inertial frame is a local North-East-Down frame (NED) fixed on an arbitrary point on the terrain whereas the body frame is on the *rover*'s center of rotation (*i.e.* intersection of

[2]A 3D vision sensor invented by Microsoft Cooperation that operates based on projecting an infrared grid patterns on the scene and reading the projection of the grid simultaneously.

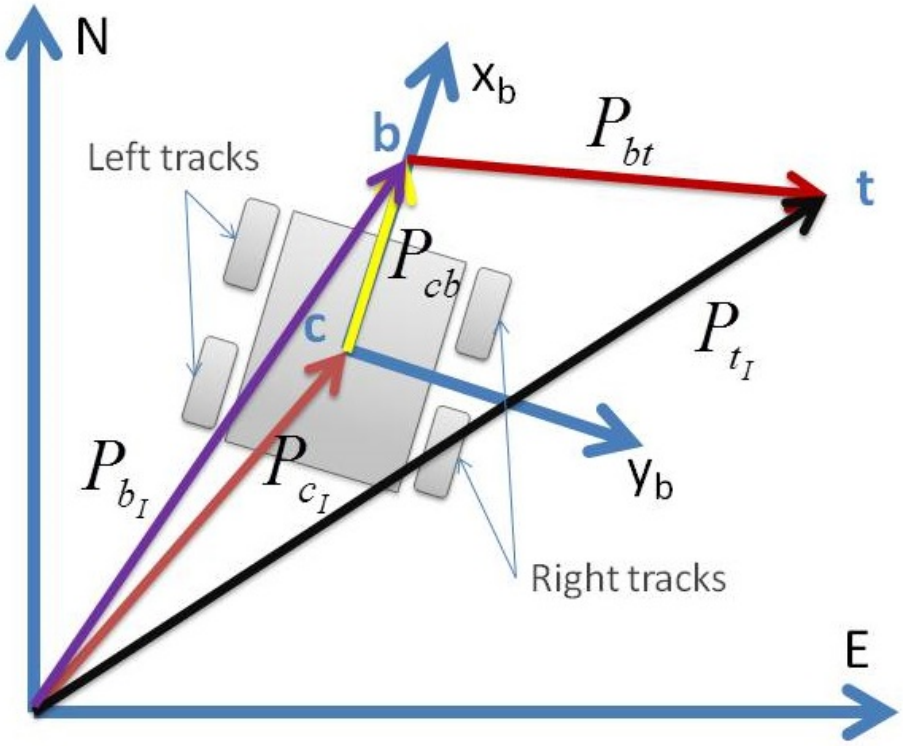


Figure 5.2: Introduction to problem coordinates and vectors.

axes of robot's differential rotation with the robot's upper surface). The x-axes of the body frame is oriented toward the front of the robot, the y-axes points toward the right side of the robot and the z-axes points down. The problem coordinates are illustrated in Figure 5.2. To circumvent the nonholonomic constraints of the rover *look ahead* model is used (see section 4.1.2). Thus, an offset distance, *i.e.* b, is introduced as reference point from which the tracking error is calculated.

The position of b can be described in the inertial frame as sum of two vectors

$$P_{b_I} = P_{c_I} + R_B^I P_{bc_B} \tag{5.1}$$

where P_{b_I} represents the position of b in the inertial frame, P_{c_I} refers to the position rover's center of rotation(c), R_B^I is the rotation matrix from the body frame to the inertial frame and P_{bc_B} describes the position of b relative to c in the body coordinates.

The kinematics of the point b can be obtained through the derivative of Equation (5.1). First let us remember the rotation kinematics being described by the well known relation of:

$$\dot{R}_B^I = R_B^I S(\omega) \tag{5.2}$$

where $S(\cdot)$ is a skew-symmetric matrix and ω represents vector of body angular rates. Furthermore, The rotational matrix R_B^I has the following properties:

$$\det\left(R_B^I\right) = 1 \qquad R_I^B R_B^I = I \tag{5.3}$$

Now we can obtain the kinematics of the b by deriving (5.1):

$$\begin{aligned} \dot{P}_{b_I} &= \dot{P}_{c_I} + \dot{R}^I_B P_{bc_B} + R^I_B \dot{P}_{bc_B} = \\ &= \dot{P}_{c_I} + R^I_B \left[S(\omega) P_{bc_B} + \dot{P}_{bc_B} \right] \end{aligned} \tag{5.4}$$

Here we deal with a 2D control problem where the rigid body is allowed to rotate only around the z axes *i.e.* $\omega = [0, 0, \omega_z]^T$. Since the position of the point b in the body frame is constant, we get $\dot{P}_{bc_B} = 0$. Finally, the kinematics of the point b described in the inertial coordinates is:

$$\dot{P}_{b_I} = \dot{P}_{c_I} + R^I_B S(\omega) P_{bc_B} \tag{5.5}$$

The nonholonomic constraints of the rover allow to describe the linear speed $\dot{P}_{c_I}$ of the rover's mass center as follows:

$$\dot{P}_{c_I} = R^I_B [U_x,\, 0,\, 0]^T \tag{5.6}$$

where U_x is the speed of the rover along the x-body axes in the body coordinates. Furthermore, the rotational kinematics can be written as:

$$S(\omega) P_{bc_B} = S(\omega) [b,\, 0,\, 0]^T \tag{5.7}$$

According to design of the hardware understudy, we suppose presence of low level controllers regulating the actual speed of the tracks to the desired ones. Given U_L and U_R as the left and right track speed respectively, the speed of the point b is described as follows:

$$U_b = \begin{bmatrix} U_{b_x} \\ U_{b_y} \\ 0 \end{bmatrix} = \begin{bmatrix} M \\ 0_{1\times 2} \end{bmatrix} \begin{bmatrix} U_R \\ U_L \end{bmatrix} \tag{5.8}$$

where $0_{1\times 2}$ represents a null matrix with dimensions 1×2 and, given the distance between right and left tracks is ℓ, M is a square matrix as follows:

$$M = \begin{bmatrix} 1/2 & 1/2 \\ -b/\ell & b/\ell \end{bmatrix} \tag{5.9}$$

Up to now, the kinematics of the rover has been presented. However, to address the problem of target tracking, robots kinematics have to be related to the one of the *evader*. The position of evader in the inertial frame is expressed as follows:

$$P_{t_I} = P_{c_I} + R^I_B (P_{bc_B} + P_{tb_B}) \tag{5.10}$$

where P_{t_I} represents position of the *evader* in the inertial frame and P_{tb_B} is the *evader*'s position relative to the point b and described in the body frame. By substituting (5.1) in (5.10) we get:

$$P_{t_I} = P_{b_I} + R^I_B P_{tb_B} \tag{5.11}$$

The kinematics of the *evader* are then derived from the previous equation:

$$\dot{P}_{t_I} = \dot{P}_{b_I} + \dot{R}^I_B P_{tb_B} + R^I_B \dot{P}_{tb_B} \tag{5.12}$$

Assumption 1 *There exists a positive D such that the speed of the evader fulfils:*

$$\|\dot{P}_{t_I}(t)\| \leq D \quad \forall\, t \geq 0$$

In other words, the *edaver*'s speed is assumed to be bounded. We can then proceed with formulating the control problem by presenting the error that has to be regulated. From Figure 5.2, it is easy to see the error to be compensated in the inertial frame can be expressed as follows:

$$e_I = P_{t_I} - P_{b_I} \tag{5.13}$$

The error can also be described in the b frame coordinates:

$$e_b = R^B_I \left(P_{t_I} - P_{b_I}\right) = P_{bt_B} \tag{5.14}$$

Thus, we aim to define a control law on U_b that fulfils

$$\|e_b(t)\| \leq \varepsilon$$

where ε is a bounded parameter such that staying the *evader* inside the SOFV is guaranteed. Furthermore we assume a zero velocity for the *evader i.e.* $\dot{P}_{t_I} = 0$. As a result the controller must ensure:

$$\lim_{t \to \infty} e_b = 0$$

5.1.2 The input constraints

In the context of the problem understudy, there are two types of constraints. The first category includes the limits coming from the nature of the system, *i.e.* actuation constraints, while the second type of constraints has to be fulfilled to guarantee a desired performance *i.e.* state constraints. Exceeding the actuation constraints in the hardware level is usually avoided by a saturation, yet to guarantee the stability of the system, the input constraints have to be taken into account in the design level of control law.

The control vector U_b relates to the commanded track speeds, U_R and U_L, through expression (5.8). Let us suppose the tracks speed is upper bounded by the finite value of V_{max} *i.e*:

$$\begin{array}{rcll} |U_R| & \leq & V_{max} & \\ |U_L| & \leq & V_{max} & V_{max} \in \mathbb{R}^+ \end{array} \tag{5.15}$$

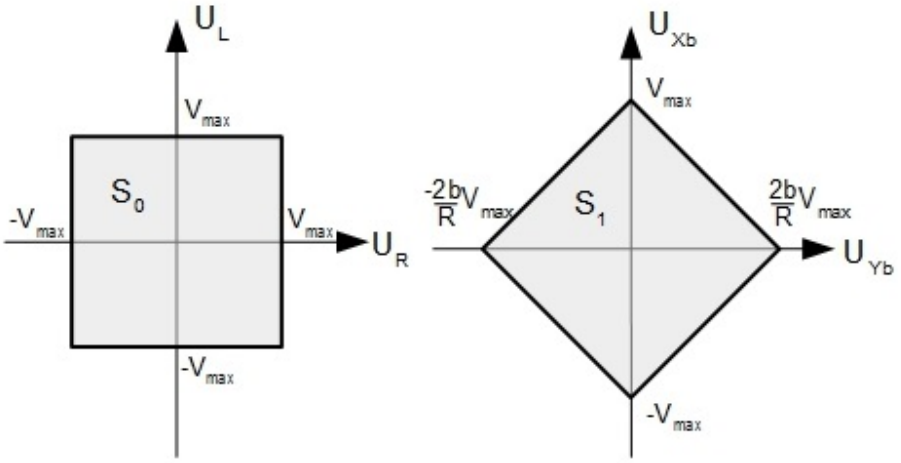


Figure 5.3: Transformation of the input set to the control set

Consequently, the track speeds physically belong to a square defined by the following set:

$$S_0 = \{(U_R, U_L) \in \mathbb{R}^2 : |U_R|,\ |U_L| \leq V_{max}\} \tag{5.16}$$

Equation (5.9) represents a linear transformation that maps the square shape set S_0 to the following diamond shape set:

$$S_1 = S_{1_R} \cap S_{1_L} \tag{5.17}$$

where

$$S_{1_R} = \left\{(U_{b_x}, U_{b_y}) \in \mathbb{R}^2 : U_{b_x} = U_R + (\ell/2b)U_{b_y}, U_R \in S_0\right\}$$

and

$$S_{1_L} = \left\{(U_{b_x}, U_{b_y}) \in \mathbb{R}^2 : U_{b_x} = U_L - (\ell/2b)U_{b_y}, U_L \in S_0\right\}$$

This transformation is shown in Figure 5.3. As long as $U_b \in S_1$ the actuation limits are fulfilled.

5.1.3 Design of the controller

The control law (5.8) is defined as follows:

$$U_b = \begin{bmatrix} U_{b_x} \\ U_{b_y} \\ 0 \end{bmatrix} = \begin{bmatrix} k_x \cos\psi \\ k_y \sin\psi \\ 0 \end{bmatrix} f\left(\|e_b\|\right),\ f\left(\|e_b\|\right) = \frac{K\|e_b\|}{\sqrt{1 + K^2\|e_b\|^2}} \tag{5.18}$$

where k_x, k_y are control parameters yet to be chosen, $\|e_b\|$ is the norm of error, K is a tunable static gain and the angle ψ is defined according to Figure 5.4. It is obvious

$$0 < f\left(\|e_b\|\right) \leq 1$$

and thus implying

$$\begin{array}{rcl} |U_{b_x}| & \leq & k_x \\ |U_{b_y}| & \leq & k_y \end{array} \tag{5.19}$$

To exploit the maximum actuation power in the x-y plane, the coefficients $k_x(\psi)$ and $k_y(\psi)$ can be adapted to the error accordingly. Considering Figure 5.4, the coefficients can be determined by intersecting e_b and the boundary of S_1 *i.e.* solving:

$$\begin{cases} k_x = |\cot(\psi)|\, k_y \\ k_x = V_{max} - (\ell/2b) k_y \end{cases} \tag{5.20}$$

That has the following answer:

$$\begin{aligned} k_x(\psi) &= \frac{V_{max}}{\left[1 + \dfrac{\ell}{2b} |\tan(\psi)|\right]} \\ k_y(\psi) &= \frac{V_{max}}{\left[|\cot(\psi)| + \dfrac{\ell}{2b}\right]} \end{aligned} \tag{5.21}$$

Finally, the control law (5.18) is mapped into left and right wheel control action U_R and U_L by means of (5.8):

$$\begin{bmatrix} U_R \\ U_L \end{bmatrix} = M^{-1} \begin{bmatrix} U_{b_x} \\ U_{b_y} \end{bmatrix} \tag{5.22}$$

5.1.4 Stability Analysis of the Proposed Controller

The dynamics of the tracking error, expressed in body frame, is obtained by deriving (5.14):

$$\begin{array}{rcl} \dot{e}_b & = & \dot{R}_I^B (P_{t_I} - P_{b_I}) + R_I^B (\dot{P}_{t_I} - \dot{P}_{b_I}) \\ & = & S(\omega) R_I^B (P_{t_I} - P_{b_I}) + R_I^B \dot{P}_{t_I} - R_I^B \dot{P}_{b_I} \\ & = & S(\omega) e_b + R_I^B \dot{P}_{t_I} - R_I^B \dot{P}_{b_I} \end{array} \tag{5.23}$$

Obviously

$$R_I^B \dot{P}_{b_I} = U_b$$

Thus, one can rewrite the equation (5.23) as follows:

$$\dot{e}_b = S(\omega) e_b + R_I^B \dot{P}_{t_I} - U_b \tag{5.24}$$

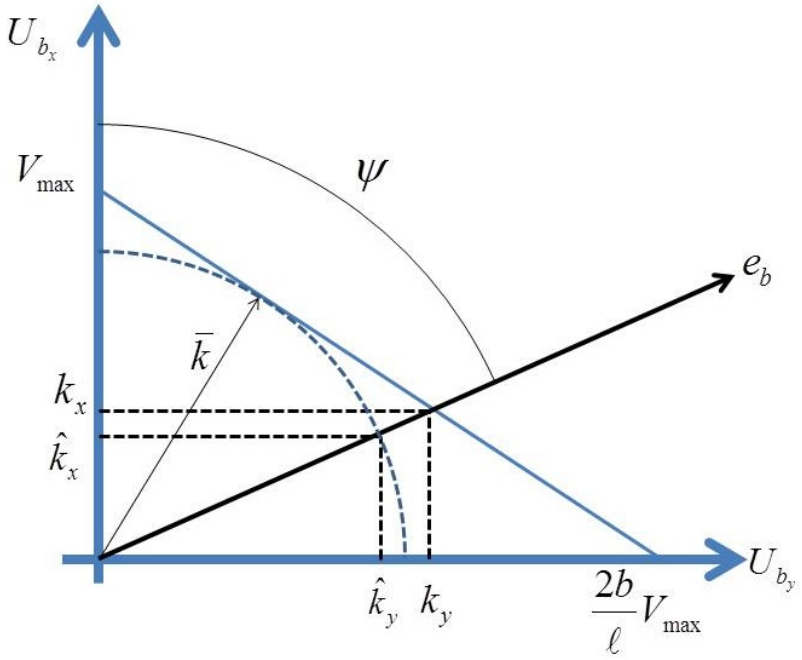


Figure 5.4: Calculation of the control action norm by superimposing error coordinates on control action coordinates

We define with ρ_m the largest value of the radius of the circle $\mathcal{E}$ centered in b and contained in the SOFV (see Figure 5.5). In the forthcoming Proposition 1, we prove that $\mathcal{E}$ is an invariant set for the error. This implies that the *evader* always remains inside the field of view.

Proposition 1 *System (5.24) controlled by (5.18)-*(5.21) *is Input to State Stable (ISS) with respect to $\dot{P}_{t_I}$, i.e. there exists a class-$\mathcal{K}$ function, $\gamma(\cdot)$, and a class-$\mathcal{KL}$ function, $\beta(\cdot)$, such that*

$$\|e_b(t)\| \leq \max\{\beta(\|e_b(0)\|, t), \gamma(\|\dot{P}_{t_I}\|)\}$$

Furthermore, there exist a $D^\star$ and a $K^\star$ such that for every $D < D^\star$ and $K \geq K^\star$ the set $\mathcal{E}$ is forward invariant for system (5.24)*, i.e. if $e_b(0) \in \mathcal{E}$ then $e_b(t) \in \mathcal{E}\ \forall\, t > 0$.*

Proof: Consider the following candidate Lyapunov function:

$$V = \frac{1}{2} e_b^T e_b. \tag{5.25}$$

Its derivative is given by:

$$\dot{V} = e_b^T \dot{e}_b = e_b^T (S(\omega) e_b + R_I^B \dot{P}_{t_I} - U_b). \tag{5.26}$$

Since $S(\omega)$ is a skew symmetric matrix and letting $R_I^B \dot{P}_{t_I} = \dot{P}_{t_B}$ we get:

$$\dot{V} = e_b^T (\dot{P}_{t_B} - U_b) \tag{5.27}$$

Defining

$$e_b^T = [e_{b_x}, e_{b_y}, 0] = \|e_b\|[\cos\psi, \sin\psi, 0]$$

and substituting (5.18) in (5.27) gives:

$$\dot{V} = \|e_b\|d - \frac{K\|e_b\|^2}{\sqrt{1+K^2\|e_b\|^2}}\left(k_x\cos^2\psi + k_y\sin^2\psi\right) \tag{5.28}$$

where

$$d = \cos\psi\dot{P}_{t_{B_x}} + \sin\psi\dot{P}_{t_{B_y}} \leq \|\dot{P}_{t_B}\| \tag{5.29}$$

As mentioned earlier, the maximum control action is identified by k_x and k_y. Let us define a pair of conservative values *i.e.* $\hat{k}_x(\psi)$ and $\hat{k}_y(\psi)$. As illustrated in Figure 5.4, the conservative values are lower bounds for k_x and k_y. To serve this purpose, we define them as follows:

$$\begin{aligned} k_x &\geq \hat{k}_x := \bar{k}\,|\cos\psi| \\ k_y &\geq \hat{k}_y := \bar{k}\,|\sin\psi| \end{aligned} \tag{5.30}$$

where

$$\bar{k} := \frac{V_{max}}{\sqrt{1+\left(\frac{\ell}{2b}\right)^2}} \tag{5.31}$$

(5.30) implies:

$$\begin{aligned} &k_x\cos^2\psi + k_y\sin^2\psi \geq \\ &\bar{k}\left(\cos^2\psi|\cos\psi| + \sin^2\psi|\sin\psi|\right) \geq \frac{\bar{k}}{\sqrt{2}} \end{aligned} \tag{5.32}$$

Finally, incorporating inequalities of (5.29) and (5.32) into (5.28), we obtain:

$$\dot{V} \leq \|e_b\|\|\dot{P_{t_B}}\| - \frac{K\|e_b\|^2}{\sqrt{1+K^2\|e_b\|^2}}\frac{\bar{k}}{\sqrt{2}} \tag{5.33}$$

The function $\gamma(\cdot)$ can be found by basic ISS arguments explained in [84].

Let us define $D^\star := \bar{k}/\sqrt{2}$ and the function $\chi(\cdot)$ as follows:

$$\chi(s) = \frac{\sqrt{2}s}{K\sqrt{\bar{k}^2 - 2s^2}}$$

Then from the ISS argument we get:

$$\|e_b\| \geq \chi\left(\|\dot{P}_{t_B}\|\right) \implies \frac{\partial V}{\partial e_b}\dot{e}_b \leq -\alpha\left(\|e_b\|\right) \quad \forall e_b \in \mathbb{R}^3$$

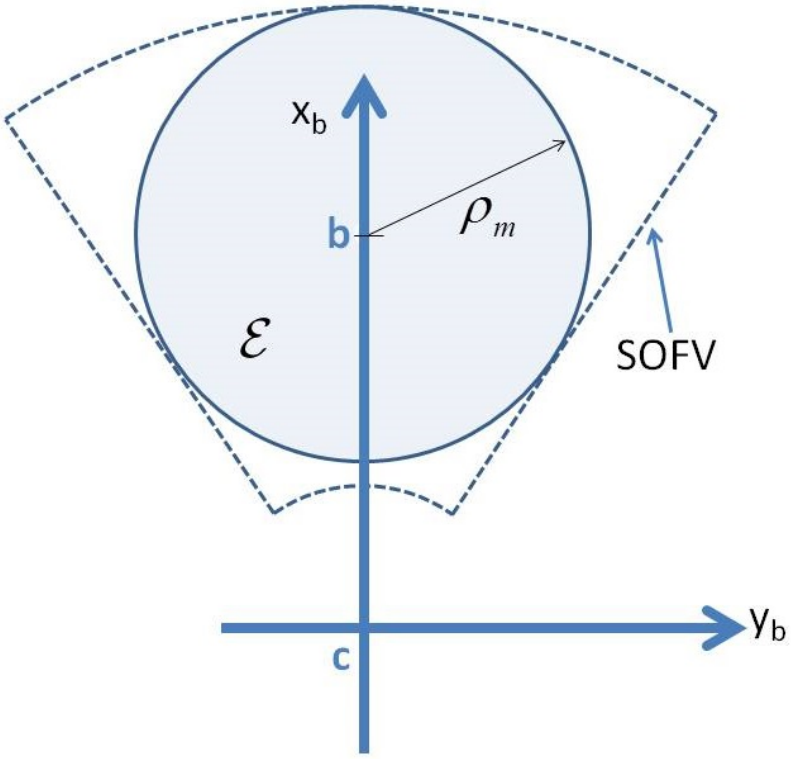


Figure 5.5: The concept of circle shaped invariant set for the error trajectories inside the SOFV.

That holds for all possible $\dot{P}_{t_B}$ fulfilling Assumption 1 and some positive definite $\alpha(\cdot)$. Then, taking into consideration $V = \|e_b\|^2$, the arguments in [84] immediately leads to the ISS inequality in Proposition 1 with $\gamma(\cdot) = \chi(\cdot)$.

The second part of the Proposition follows by ISS by taking $K \geq K^\star$ with $K^\star$ defined as:

$$K^\star = \frac{\sqrt{2}D^\star}{\rho_m\sqrt{\bar{k}^2 - 2D^{\star 2}}} \tag{5.34}$$

In summery, we have proven that given an *evader* complying with the assumption 1 and laying on the circle demonstrated in Figure 5.5, the position of the *evader* in the body coordinates will be attracted to center of the circle by the control law of (5.19). In the inertial frame coordinates, this interpreted as the *rover* following the *evader*. As seen in the analysis, the circle can be tuned to be the largest possible circle that can be fitted in the SOFV. However, the presented analysis only guarantee the stability in a circle.

5.1.5 Simulation Result

To evaluate the proposed structure, the system understudy is simulated in Matlab Simulink®. Figure 5.6a shows the tracking performance of the proposed system in the inertial frame. As seen, the robot is able to track the *evader*'s trajectory. Figure 5.6b demonstrates some sample trajectories of the error in the body frame starting from the boundary of $\mathcal{E}$. As expected, the trajectories are invariant with respect to the circle $\mathcal{E}$ and the state constraints are satisfied. In the case of conducted simulation which assumes a standstill target, the error is attracted to the origin. In the inertial frame, this means the robot follows the evader as soon as an error is introduced. Finally, Figure 5.7 verifies that the actuation constraints are also fulfilled for all $t \geq 0$. As shown in the figure, the linear speeds of actuators are not exceeding the limit (*i.e.* $V_{max} = 3\,m/s$ for this example) while both the x and y components of the error are stable.

5.1.6 Experimental Result

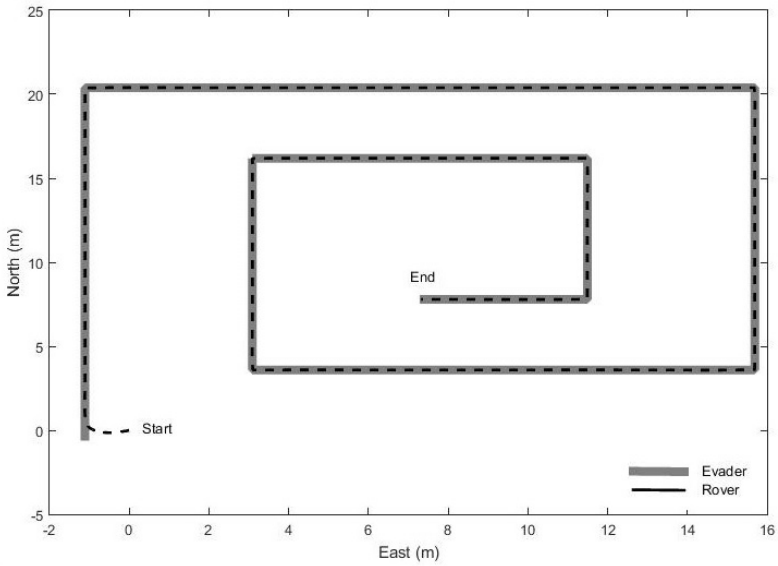
In order to evaluate performance of the proposed structure in practice, the developed control law is implemented on the *donkey rover* which is a skid driven robot with four tracks. The system is equipped with a XtionPRO ASUS Kinect [85] as the vision sensor. For the specific model that has been used in this work the operation range is between $0.8m$ and $3.5m$ and field of view is $70°$.

The implementation of the proposed control law has led to the definition of the origin of the error in the body coordinates, b, is equal to $2.23\,m$ and radius of the maximal invariant circle C_m inside the SOFV is $\rho_m = 1.27\,m$.

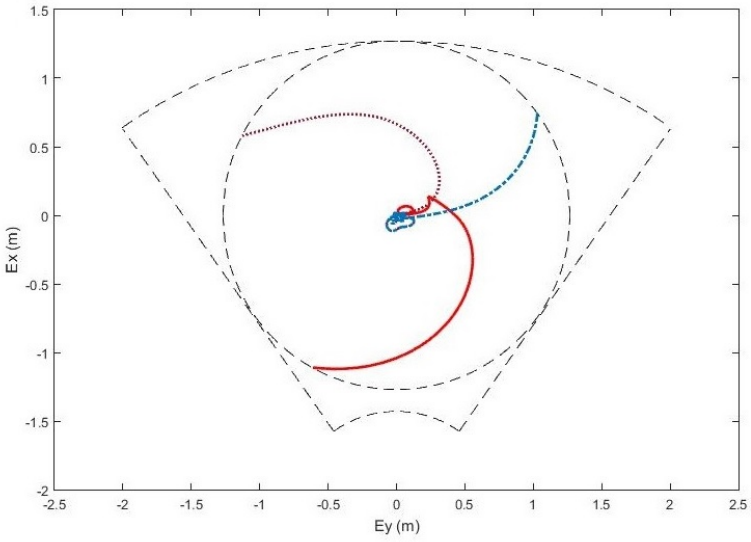
Figure 5.8 illustrates the experimental set up of this study. As mentioned earlier, the developed system in this section is not capable of rough terrain navigation and obstacle avoidance. To perform a safe navigation we have selected our laboratory in University of Bologna, "CASY", as the experiment site.

The algorithm behind the experiment is implemented in ROS indigo environment and "openni tracker" [86] is used to find the position of the *evader*. *Openni tracker* recognizes and tracks 15 benchmarks of the human body and one of them has to be selected as representative of the *evader* (*i.e.* head in this experiment). The tracking software is initialized by a specific body gesture demonstrated in Figure 5.8. It then supplies the error to the control algorithm described earlier in this section. Figure 5.9 demonstrates the result of experiment that was conducted for about 43 seconds. As can be seen, both components of the error are stable while actuation limits are respected. A recorded footage of the experiment is also available on our on-line repository[3].

[3] `https://www.youtube.com/watch?v=tO3RHHmhxxQ`



(a) Tracking trajectory in the inertial frame.



(b) Tracking trajectory in the body frame.

Figure 5.6: Simulation result of the proposed system a) Trajectories of the *evader* versus the rover in the inertial frame b) Some example trajectories of the error in the body frame inside the SOFV with different initial conditions on the boundary of C_m.

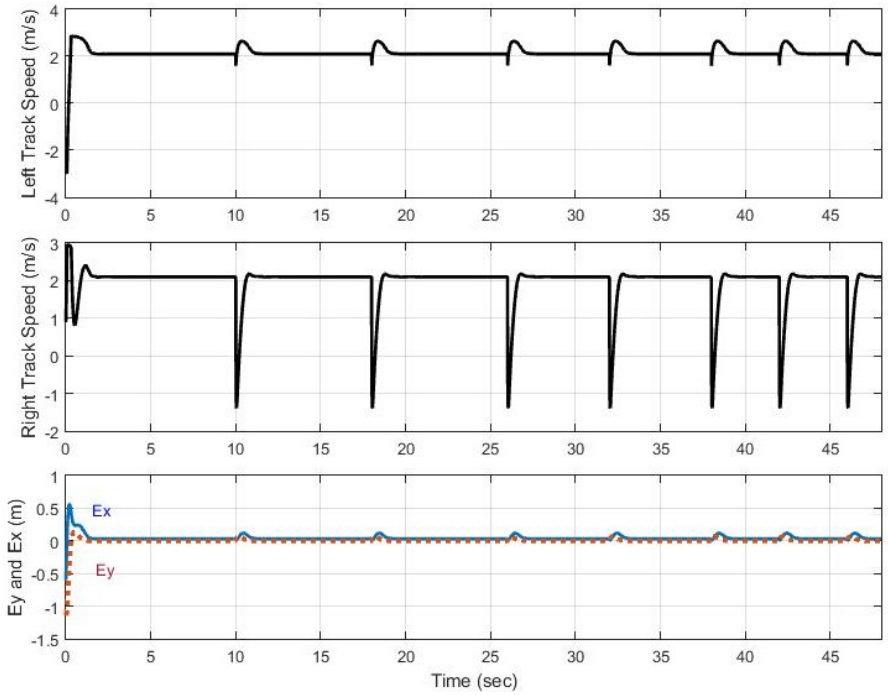


Figure 5.7: Actuators speeds and error components during the simulation. A limit of $V_{max} = 3\,m/s$ is considered in this example.

Figure 5.8: Experimental setup of this study.

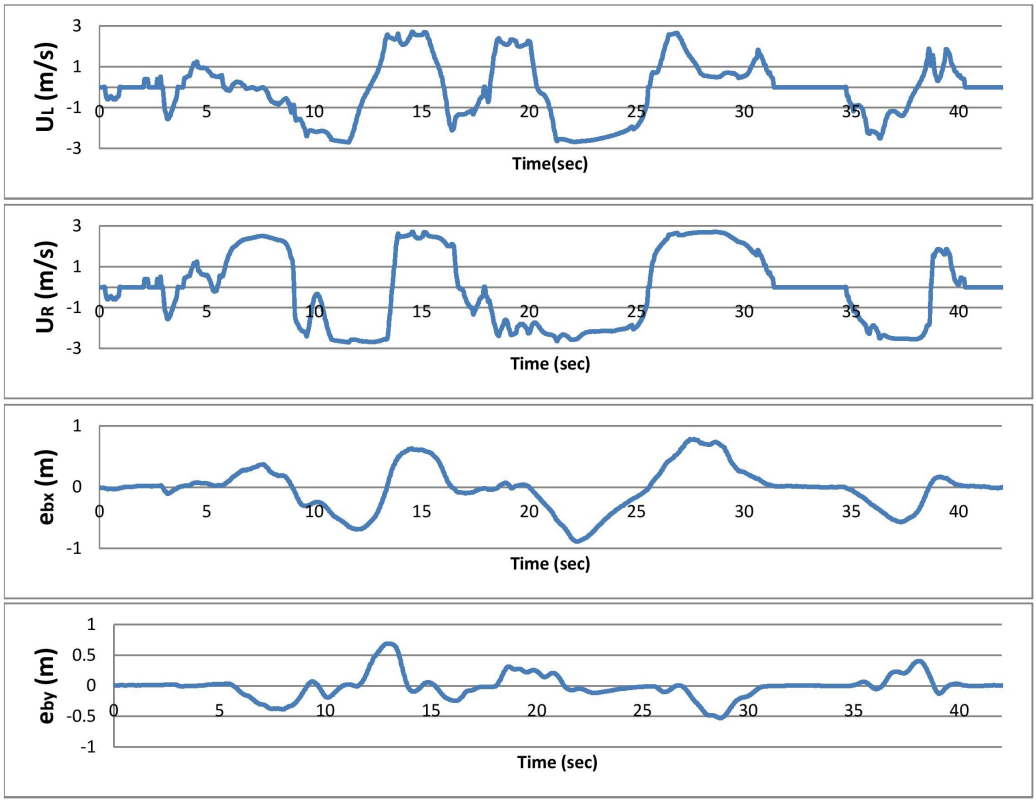


Figure 5.9: Result of experimentation that has been conducted for 43 seconds, the figure demonstrates speed of right and left actuators and error components in the body frame

5.2 Cooperative Following using the Arm and the Rover

The previous section investigated implementation of an *evader* following by means of a sensor that is fixed on the robot. The main drawback of such an implementation is the SOFV rotating with the robot limiting the robot's capability to optimize chassis terrain interaction. In this section, we seek a solution to the target following problem that involves both rover and arm. The main motivation to involve the arm is endowing the sensor with a flexible platform and ultimately increase the *rover*'s manoeuvrability.

To pursue this purpose, we approach the problem by means of a cooperative framework based on the "cooperative game theory" [87, 88] to couple the *arm*'s and *rover*'s behaviour. Similar to the previous section, the system has the same goal of following the target. Although in this case, safe navigation is a priority to be respected. To achieve a superior performance, the two robots have to consider strategies taken by their partners and bargain upon a solution in the pursuit of achieving the mutual interest while respecting the limits of the partners.

Here we choose a strategy based on Model Predictive Control (MPC) enabling the robot to predict the results of control actions. Thus, the outcomes of different control inputs can be predicted while the system's kinematics and dynamics are taken into account. As a result, the output trajectory obtained by a Model Predictive Planner (MPP) is kinematically and dynamically feasible [89–91].

5.2.1 Components Multi-Robot Interaction

Let us introduce the main components of the desired system, namely the ground rover and robotic arm, and lays out the framework used for the interaction between these systems. We formulate the task, *i.e.* continuously tracking a target with the arm while the rover approaches it, as a cooperative game in which each player maximizes their profit through cooperation.

5.2.1.1 Ground Rover

Here we use a simple representation of the *rover*'s kinematics without considering the look ahead change of coordinates. The kinematics of this differentially driven robot can be described as follows;

$$\begin{bmatrix} \dot{N} \\ \dot{E} \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} \cos(\theta_r) & 0 \\ \sin(\theta_r) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V_l \\ \omega_r \end{bmatrix} \tag{5.35}$$

where N, E and θ_r describe the 2D pose of the vehicle in the world reference frame referring to north, east and yaw respectively. Furthermore, V_l is linear speed and ω_r is the rotational speed of the rover around its center of differential rotation.

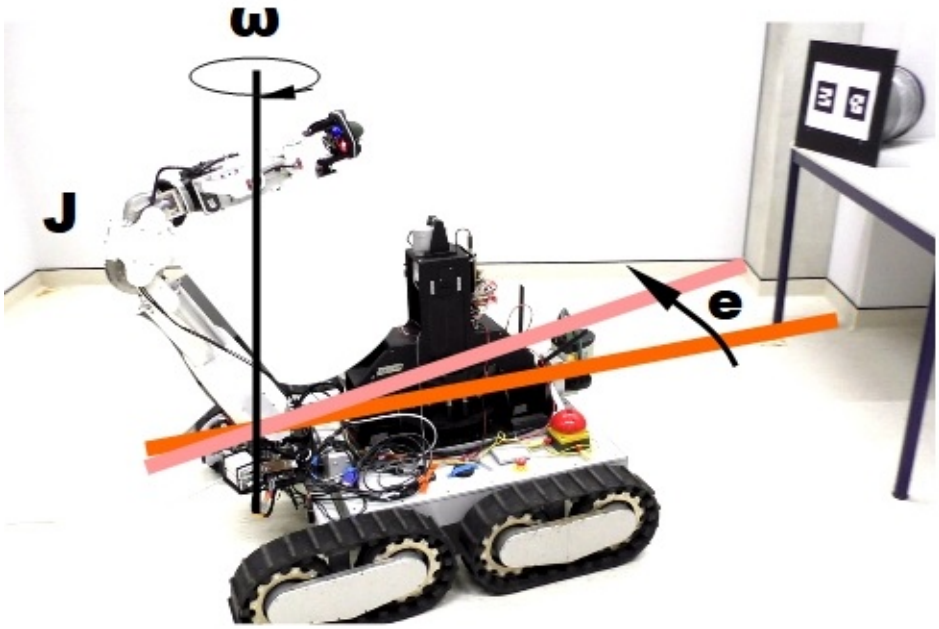


Figure 5.10: The arm with angular velocity ω and inertia J is controlled to track the target.

The task of the rover is to find and execute a path to the goal that is not only safe and feasible, but also facilitates the tracking of the target by the robotic arm.

5.2.1.2 Robotic Arm

As described in section 3.2, the *arm* mounted on the *rover* is a 7-DoF robotic manipulator that is equipped with a camera on its end-effector. Apart from manipulating objects, it can also be used to improve the overall perception of the system by offering a better and adaptable point of view, than is attainable from the chassis of the rover.

The task of the arm is to continuously track the target, for which it only uses one DoF in the shoulder which is PD-controlled with the angular error between the arm and the target, as shown in Figure 5.10. This simple scheme not only reduces the system complexity, but more importantly avoids collisions of the arm with itself or the rover. The arm's dynamics can thus be described as a controlled second order system:

$$\dot{\omega} = {}^{\delta}/_{J}\,\omega + {}^{\tau}/_{J} \tag{5.36}$$

with the arm's angular velocity ω, damping δ, rotational inertia J, and control torque τ.

5.2.1.3 Cooperative Game Theory

A dominant strategy $\hat{\gamma}$ that guaranties the lowest cost for all players cannot be achieved in the context of cooperative navigation and obstacle avoidance, as the rover prioritizes a safe path over target tracking. Two concepts are important to instead find a strategy that leads to the lowest overall cost, *Pareto efficiency* and *Bargaining* [92].

Pareto efficiency *In a cooperative game, a set of player strategies, $\hat{\gamma}$, is called Pareto efficient if it is impossible to find a different set of strategies, i.e. γ, that improves the cooperative performance of all the players (i.e. J_i). In other words, a set of inequalities $J_i(\gamma) \leq J_i(\hat{\gamma})$ with at least one strict inequality in the solution space of the game (i.e. Γ) cannot be found.*

Bargaining problem *A situation in which a number of players have to agree upon a set of strategies to achieve a superior performance through cooperation at the cost of inferior individual performances due to conflicting interests.*

Pareto efficient strategies are not unique and the set of all Pareto strategies is referred to as "Pareto frontier" [92]. The selection of the proper Pareto solution out of the Pareto frontier is the bargaining problem. In other words, bargaining is a way of sharing profit of the cooperation among the players. Whereas the economics literature presents solutions such as Nash bargaining, Kalai-Smorodinsky, and the egalitarian bargaining solution, we solve the bargaining problem by the introduction of constraints. Finally, the lower bound of the profit share that each player can accept is the player's "disagreement point". The different concepts in the context of game theory are illustrated in Figure 5.11.

5.2.1.4 Educative Interaction

The players of our system affect each other in different ways. The strategy that the arm chooses does not affect the rover, while the rover's path, on the other hand, has a direct influence on the arm's strategy. According to [93] an interaction between players can either have a *master-slave* characteristic, when the dominant player ignores the cost incurred by the slave. Or have an *educative* nature, when the *teacher* chooses a strategy that maximizes the overall profit. We have thus chosen a *educative* framework for the set of non linear time variant systems with a single teacher and N students described below:

$$\begin{cases} \dot{x}_t(t) = f_t(t, x_t) + g_t(t, x_t)u_t \\ \dot{x}_{si}(t) = f_{si}(t, x_{si}) + g_{si}(t, x_{si})u_{si}, \quad i = 1, ..., N \end{cases} \tag{5.37}$$

where the subscripts t and s, respectively, show *teacher* and *student* systems. The quadratic cost functions of the subsystems are defined as follows:

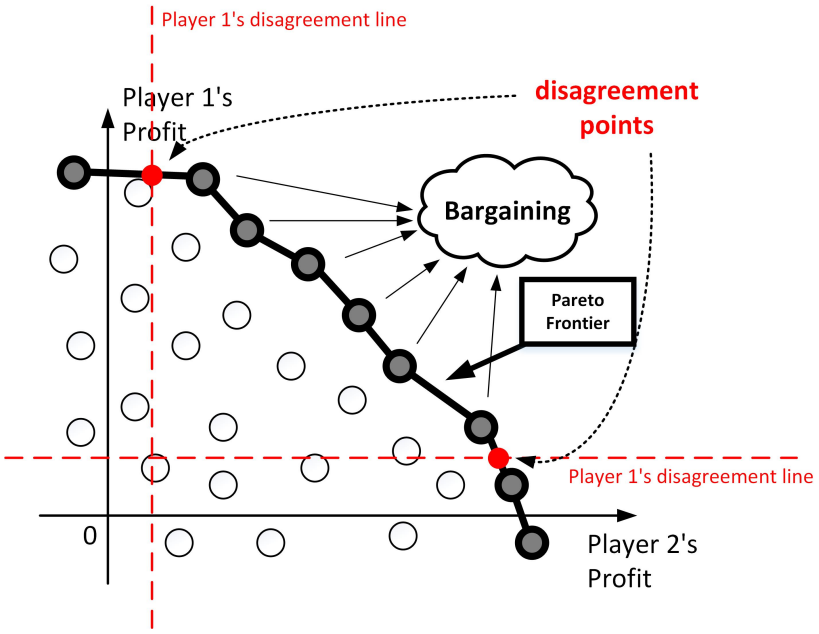


Figure 5.11: Illustration of different concepts of the cooperative game theory in a simple two player game. The horizontal and vertical axis indicate players' profit. In this example, the players need a minimum profit to be motivated to take part in the game. This concept is depicted by the disagreement lines for each agent.

$$\begin{cases} J_{si} = \int_0^T (x_{si} Q_{si} x_{si}^\intercal + u_{si}^\intercal R_{si} u_{si}) dt, \quad i = 1, ..., N \\ J_t = \int_0^T \left(\alpha_t (x_t Q_t x_t^\intercal + u_t^\intercal R_t u_t) + \sum_{i=1}^{N} \alpha_i x_{si} Q_{si} x_{si}^\intercal \right) dt \end{cases} \tag{5.38}$$

where Q_{si}, R_{si}, Q_t and R_t are square symmetric matrices of the proper dimensions and the weights α are some positive scalar coefficients satisfying

$$\alpha_t + \sum_{i=1}^{N} \alpha_i = 1$$

The cooperative optimization problem, then, is only attended to by the *teacher* while each *student* tries to minimize its own cost in a non cooperative way:

$$\begin{aligned} \hat{u}_t &= \arg\min_{u_t \in \Gamma_t} J_t(x_t, u_t) \quad s.t. \quad x_t \in S_t \\ \hat{u}_s &= \arg\min_{u_s \in \Gamma_s} J_s(x_s, u_s) \quad s.t. \quad x_s \in S_s \end{aligned} \tag{5.39}$$

where Γ and S represent constrained input and state spaces respectively. Note that in an *educative* game framework, the *students* are not compromising for the *teacher's* performance, as the *students'* effort u_{si} is absent in the teacher's cost definition in (5.38). Furthermore, the bargaining problem is formulated in (5.38) by the α coefficients. We define the disagreement point of a player as the exceedance of its constraints. The bargaining can then be treated as a degree of freedom for tuning the tracking performance as long as the constraints are handled properly.

5.2.2 Model Predictive Planing

This section describes the MPP approach used to predict the outcomes of control actions of the players in order to find an optimal solution to the *bargaining* problem.

5.2.2.1 MPP formulation

Our approach is motivated by [90], in which the general optimal control problem is transformed to a parametric optimization problem by parametrization of the input state. We can reformulate both systems of the rover and arm from Equations (5.35) and (5.36) in the form

$$\dot{x}(t) = f(x(t), u(q, t)) \tag{5.40}$$

where q refers to the vector of parameters that will be discussed in details in the following section. The parametrized optimization problem can therefore be formulated as finding a candidate q^* that minimizes a cost function of the form $J(x(t), u(q, t))$ under the condition that the boundary conditions are met.

5.2.2.2 MPP solution with Particle Swarm Optimization

The non-linearity of (5.40) implies a non-linear MPP problem which can be approached by a heuristic method. In this work we solve the problem by means of Particle Swarm Optimization (PSO) where a swarm of solutions S is evolved through an iterative method. In each iteration, a set of candidate solutions, also known as particles x^k, is tried and evaluated using a cost function C. The evolution of the swarm is accomplished using a velocity term v^k which attracts the next particle set x^{k+1} to the best solution of the iteration P^{best} and the best solution in the entire swarm G. In other words:

$$\begin{aligned} x^{k+1} &= x^k + v^k \\ v^k &= \omega x^k + c_1 r_1 (P^{best} - x^k) + c_2 r_2 (G - x^k) \end{aligned} \tag{5.41}$$

Where ω, c_1 and c_2 are tunable PSO parameters, and r_1 and r_2 are randomly generated values in the range of $[0, 1]$. The MPP problem of (5.40) can be formalized and solved as a PSO problem. To be able to approach the constrained problem by PSO, we define a penalty factor in the cost function as explained in [94], causing the solutions which do not satisfy the boundary conditions to present a high cost and eventually be avoided. The cost function of the PSO algorithm is therefore defined as follows:

$$C = J(x, u) + h(x, u) \tag{5.42}$$

5.2.3 Path Planning

The path planning problem of the rover is formulated and solved as MPP problem, in which the input space of the system given in Equation (5.35) is searched to find a optimal solution satisfying the constraints. The chosen trajectory leads the rover to the goal position while avoiding obstacles and minimizing the combined cost of the arm and rover.

5.2.3.1 Input Parametrization

Polynomial spirals can be used to parametrize the input state of the path and to ultimately represent any feasible vehicle motion through a small number of parameters [89]. Considering (5.35) and describing the curvature $\kappa(t)$ in a third degree polynomial form, we obtain the following representation:

$$\begin{cases} \kappa(s) = a + bs + cs^2 + ds^3 \\ \omega(t) = \kappa(t)v(t) \end{cases} \tag{5.43}$$

where a, b, c and d are coefficients of the polynomial and s is an independent variable. Thus, a vector of parameters can be defined as follows:

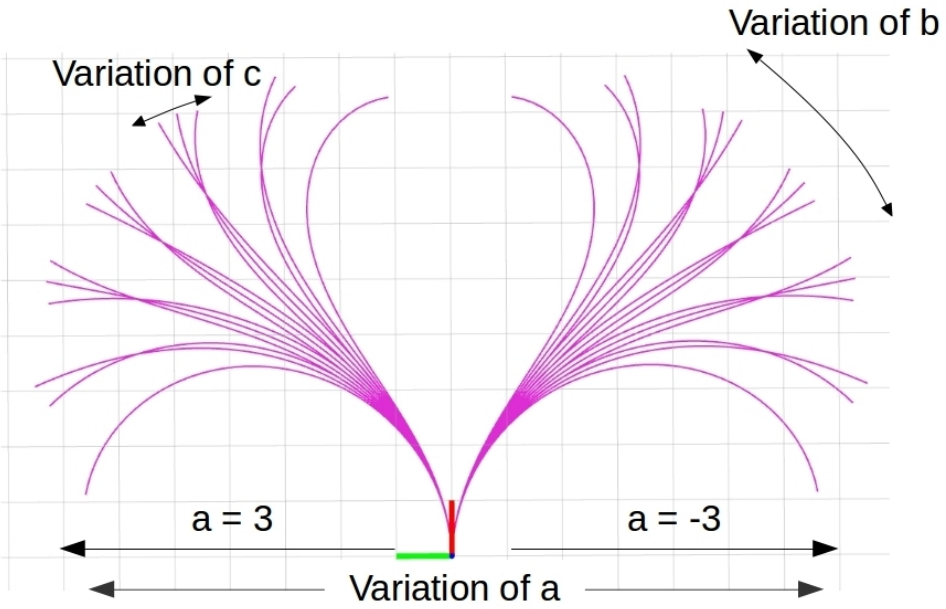
$$q = \begin{bmatrix} a & b & c & d & l \end{bmatrix}^T \tag{5.44}$$

where l is the length of the path. Note that (5.43) suggests one more degree of freedom which is the norm of variable s, although in this paper we considered a fixed constant value for this parameter.

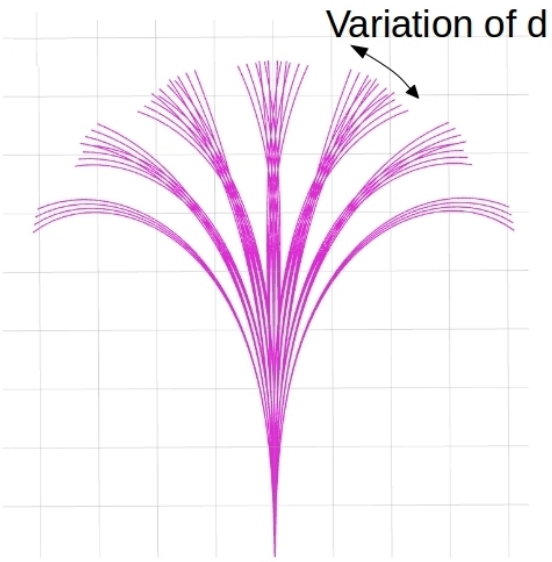
Figure 5.12 illustrates the influence that the elements of q have on the shape of the path. Changing lower order coefficients, i.e. a and b, causes substantial changes in the path structures, while neat adjustments to the goal of the trajectory can be made with the higher order coefficients c and d.

5.2.3.2 Initialization of PSO algorithm

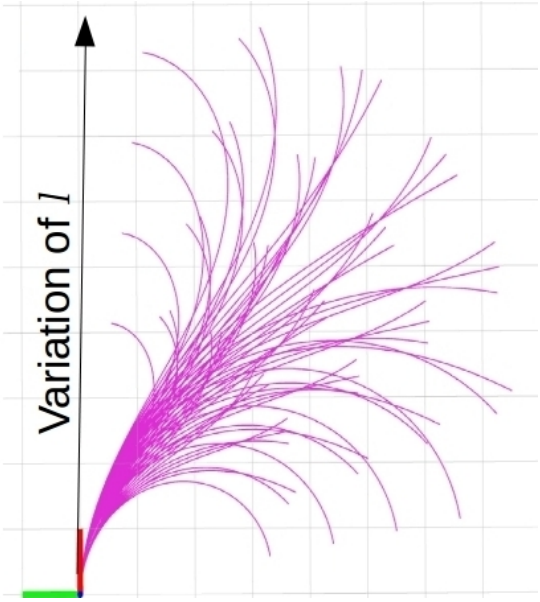
Solving the PSO optimization problem characterized by the cost function (5.42) is challenging not only because of the infinite possible solutions for q. More importantly, many local minima need to be avoided when approaching the constrained optimization problem by a heuristic penalized optimization method. In the worst case the swarm may contain no solution that complies with the boundary conditions. It is therefore crucial to initialize the algorithm to achieve a desirable performance. This can be accomplished by a set of predefined solutions stored in a Look-Up Table (LUT). Given a navigation goal, the system makes an inquiry to the LUT for a number of possible solutions. The solutions acquired from the LUT are used as the first set of particles of the optimization algorithm and help the convergence to the optimal solution.



(a) Effect of variation of a, b, and c;

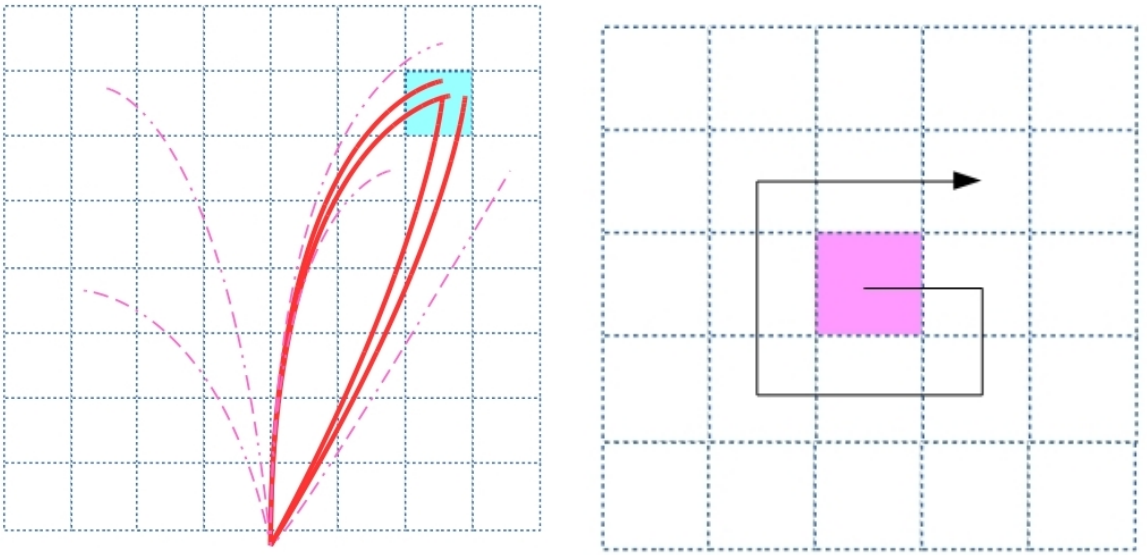


(b) Variation of d.



(c) Variation of l.

Figure 5.12: Different patterns that are used in the look up table, variation of elements of q can create a variety of shapes



(a) Path ordering (mapping) (b) Path acquisition (search)

Figure 5.13: Selection of viable paths from the look up table. Trajectories that end in the goal cell (solid) are selected as initial particles for the optimization algorithm, while other paths (dotted) are discarded, as shown in Figure 5.13a. If not enough initial candidates are found, the range of goal cells is expanded in an outward spiraling pattern, as shown in Figure 5.13b.

The LUT contains a set of possible solution consisting of: 1) The parameter vector q, 2) A path associated to q, 3) A travel cost representing how straight the path is, and 4) The destination of the path. The LUT keeps an order of the predefined paths based on discretized values of their destinations. In other words:

$$[d_x, d_y]^T = floor(s[c_x, c_y]^T) \tag{5.45}$$

where d and c are respectively discrete and continuous values and s is a scaling factor of the unit $cell/meter$.

Those paths that end up to the goal cell are selected from the LUT as initial candidates, as shown in Figure 5.13a. If the number of available paths in one cluster is not sufficient for the initialization purpose, a simple path acquisition method is used. In this case, paths from surrounding cells will be invoked by the mechanism illustrated in Figure 5.13b.

5.2.3.3 Cost Function

The cost function of (5.42) is composed of two parts, the MPP cost function $J(x, u)$ and the penalty factor $h(x, u)$. These terms are composed of the following components:

- Chassis cost: Having a candidate path $\mathbb{P}$, the chassis simulator can estimate the result of the rover terrain interaction as a set of orientations $\mathbb{O} \triangleq \{\mathfrak{F}_{ch}(x) | x \in \mathbb{P}\}$. The variance

of $\mathbb{O}$ is, therefore, used in the cost function to represent the energy required to traverse $\mathbb{P}$.

$$J_{ch}(x) = \mathrm{Var}(\mathbb{O}) \tag{5.46}$$

- Inflation cost: In order to avoid potential hazards associated with passing in the vicinity of obstacles, an inflation layer, $\mathbb{INF}$, is considered around the occupied cells in the costmap. A constant cost of C_{inf} is then considered for each member of $\mathbb{P}$ that is located in an inflated cell.

$$J_{inf}(x) = \mathrm{size}(\mathbb{P} \cap \mathbb{INF}) \times C_{inf} \tag{5.47}$$

- Arm cost: Referring to cooperative game theory equations (5.38) and (5.39), a simple arm simulator is needed to calculate the error that arms controller has to compensate if a candidate path is taken. Thus, the arm simulator calculates a set of end effector orientation that should be applied to track the target $\mathbb{A} \triangleq \{\mathfrak{F}_{arm}(x, G_{arm}) | x \in \mathbb{P}\}$. The arm contribution to the cost function can be then described as:

$$J_{arm}(x, G_{arm}) = \mathrm{Var}(\mathbb{A}) \tag{5.48}$$

- Goal distance penalty: Design of the proposed system penalizes big and small errors with different gains as the former implies an inferior and yet an acceptable solution while the latter causes violating the boundary conditions. Thus, we consider a threshold, Err, to distinguish constraint violating solutions. It is obvious that the continuity of this factor has a great impact on convergence of the algorithm. Therefore, having the error Δx between the desired goal and the candidate path destination, a cost value J_{Err} associated to Err and a penalty gain K for unacceptable solutions, we define the goal distance penalty as follows:

$$h_{goal}(x) = \begin{cases} \frac{J_{Err}}{Err}\Delta x & \Delta x \leq Err \\ (\frac{2\Delta x}{Err} - 1)^K \frac{J_{Err}}{Err}\Delta x & \Delta x > Err \end{cases} \tag{5.49}$$

- Obstacle penalty: Similar to inflation cost, a constant cost of C_{obs} is considered to penalize passing through the set of occupied cells $\mathbb{OBS}$. It is obvious that $C_{obs} > C_{inf}$ as passing through the inflated layer is acceptable while occupied cells are not traversable.

$$h_{obs}(x) = size(\mathbb{P} \cap \mathbb{OBS}) \times C_{obs} \tag{5.50}$$

- Chassis penalty: It is necessary to penalize unacceptable and hazardous attitudes that can result in a flip over. We therefore define a set of hazardous attitude $\mathbb{HAZ}$ to penalize paths that implies such attitudes.

$$h_{att}(x) = size(\mathbb{O} \cap \mathbb{HAZ}) \times C_{haz} \tag{5.51}$$

To summarize, the cost function of (5.42) can be rewritten as (5.52). Note that even though the constraints are treated as an added cost value, the algorithm can still reject the solutions

which their penalty factor exceeds a threshold and ultimately do not satisfy the boundary conditions. The cooperative cost function of the rover is formulated in (5.52). As mentioned earlier, the bargaining coefficients $\alpha_{\mathbf{t}}$ and $\alpha_{\mathbf{s}}$ can be used to tune the level of cooperation among the two agents.

$$\begin{aligned} C(x) = \alpha_{\mathbf{t}} \left(J_{ch}(x) + J_{inf}(x) \right) + \alpha_{\mathbf{s}} J_{arm} \\ + \left(h_{goal}(x) + h_{obs}(x) + h_{ch}(x) \right) \end{aligned} \tag{5.52}$$

5.2.4 Experimental Results

This section presents the experimental results validating the cooperative navigation of the rover with simultaneous tracking of the target by the arm. Figure 5.14 shows the set-up of this experiment including the rover, the mounted arm, the target and the environment including foam cushions mimicking uneven terrain and obstacles.

The proposed algorithm is implemented in the Robot Operation System (ROS) [95]. An open source marker detection package [96] is used to localize and track the target, while the rover-world localization is provided by Hector SLAM [27,97] which incorporates LiDAR and IMU data to provide 2D mapping with a limited 3D localization. Finally, the 3D Point cloud analysis is facilitated by the open source PCL library [98, 99]. The experiment is executed as follows:

1. The target is localized by the camera of the arm.
2. The rover performs a tilted scan to acquire data to be used by the navigation.
3. The algorithm solves the cooperative path planning problem.
4. The rover starts navigating the solution path, while the localization of the rover is performed using a fixed configuration of laser scanner and IMU.
5. The arm localizes and tracks the target while the rover is approaching the target.

The performance of the proposed algorithm is demonstrated in Figure 5.15. The path finder algorithm is ready to start as soon as the target is localized. After a navigation goal is received, elevation data is calculated from a single tilt scan (Figure 5.15a). To initialize the optimization algorithm, some precalculated paths are acquired from the LUT (Figure 5.15b). Based on (5.41), the optimization algorithm evolves a swarm of paths which is growing in each iteration (Figure 5.15c). Furthermore, a swarm of best particles (G) is evolved in the course of the algorithm run-time that is demonstrated in Figure 5.15d.

Finally, result of the experiment is demonstrated in Figures 5.16 and 5.17. Figure 5.16 illustrates the target pose, initial rover pose, rover goal pose and a 2D map of the environment acquired by the SLAM algorithm that serves for real-time localization of the system. The

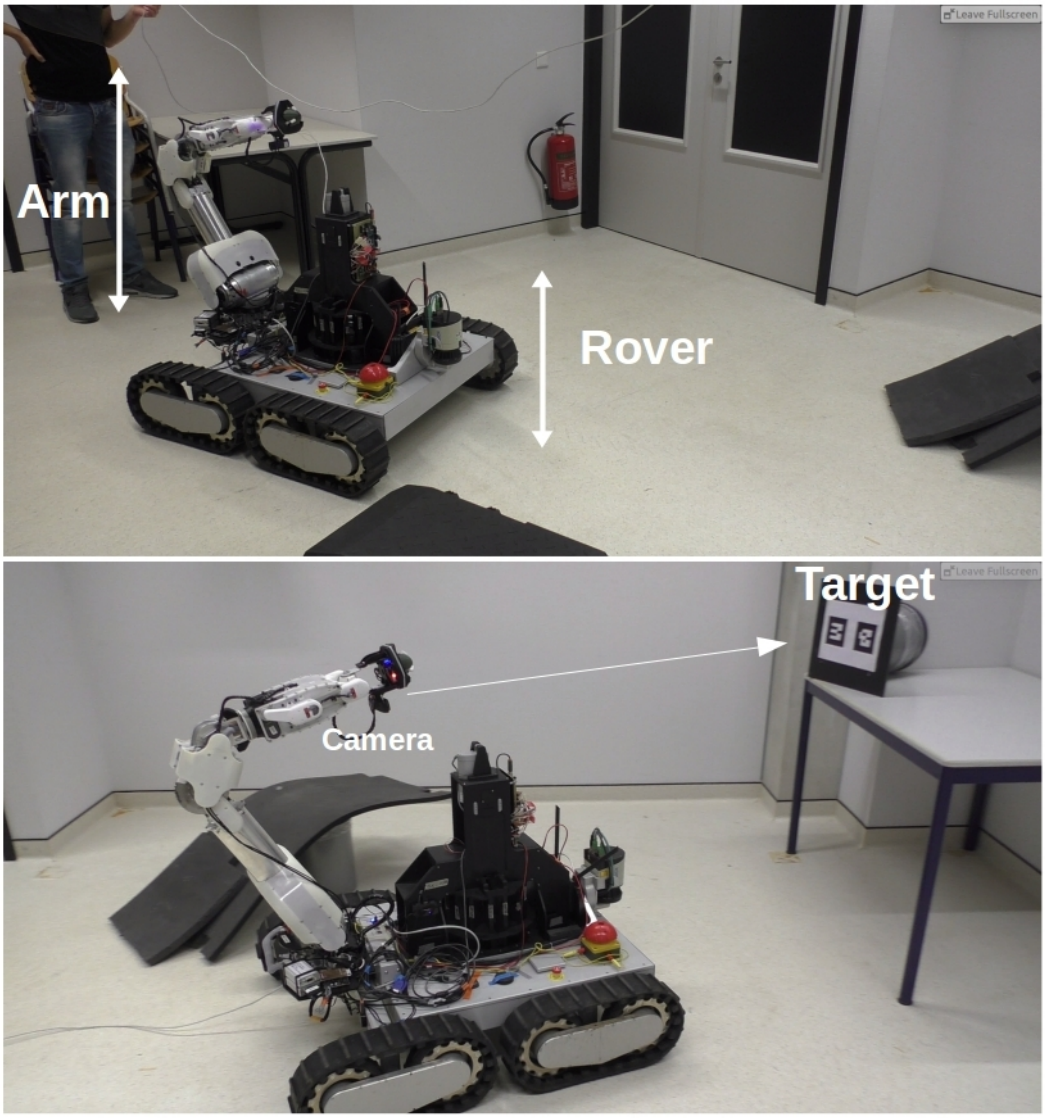
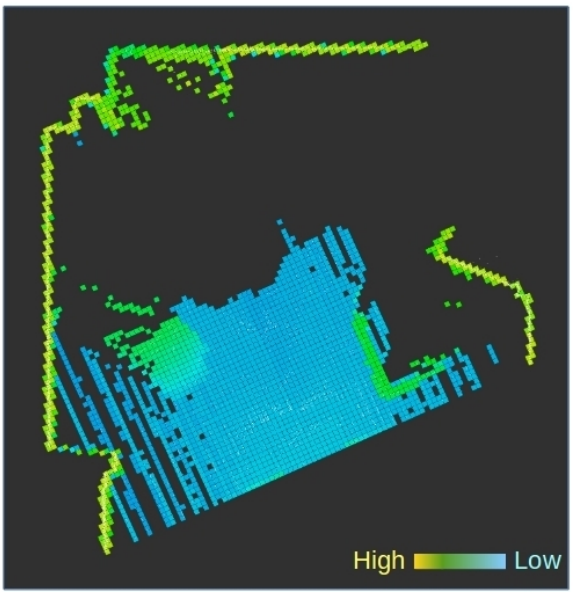
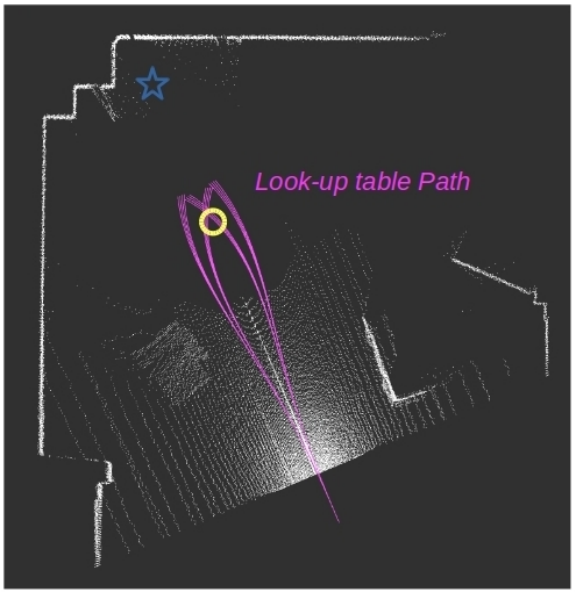


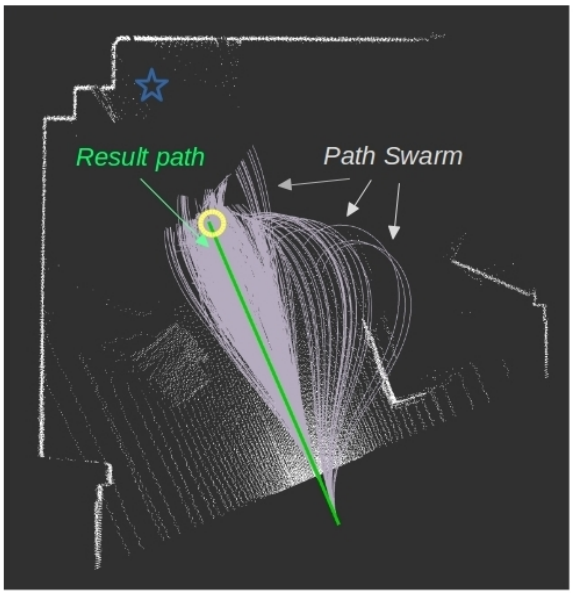
Figure 5.14: The experimental set-up including the rover, arm, target, and obstacles.



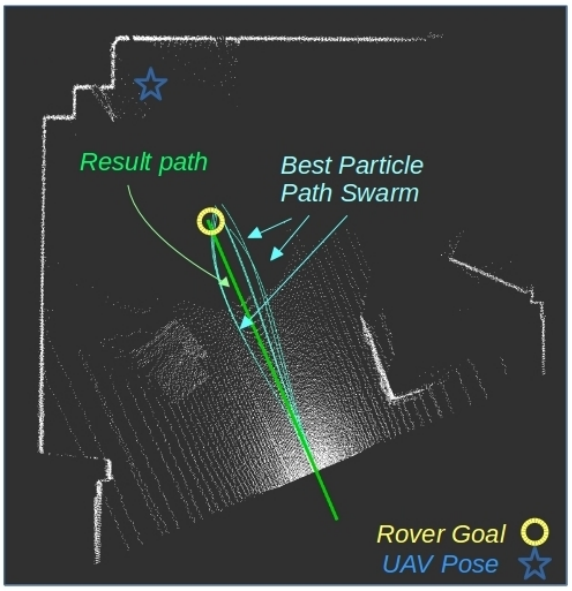
(a) Elevation data acquired in a single scan.



(b) Pre-calculated solutions proposed by the LUT.



(c) Swarm of the particles and the result solution.



(d) Best solutions of the particles(G) with and the result solution.

Figure 5.15: Performance of the proposed path finder algorithm.

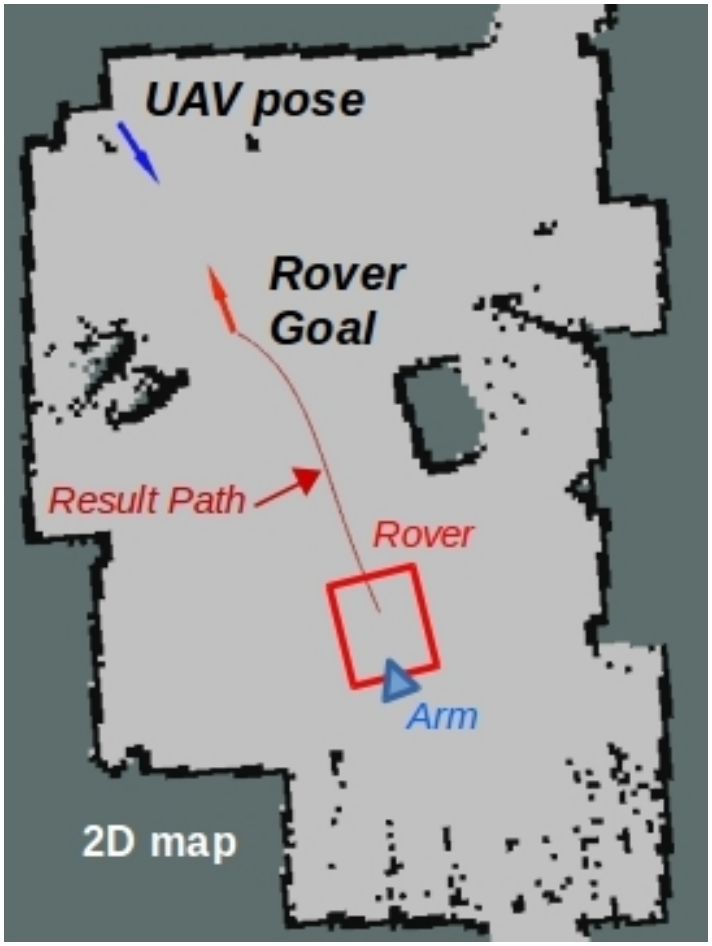


Figure 5.16: The path planned by the rover with initial and goal positions in the map of the environment.

resulting path is also demonstrated in this figure while a video of the experiment can be found in our online repository[4]. Figure 5.17 shows the position of the arm's shoulder joint while tracking the target during the approach.

5.2.5 Discussion

The main contribution of the presented approach is to define the concept of cooperative planning for navigation tasks. Even though the experimental results presented in this paper could easily have been achieved with a simple centralized planner, it was our aim to demonstrate cooperative navigation in practice. Comparing to a centralized approach, our strategy is more versatile and expandable. More players can be added to the system to perform more complex tasks that cannot be completed by a single player with centralized planning. For example, map can take part in the game by inviting the rover to navigate through well discovered areas

[4]https://youtu.be/3B41FBIsO2g

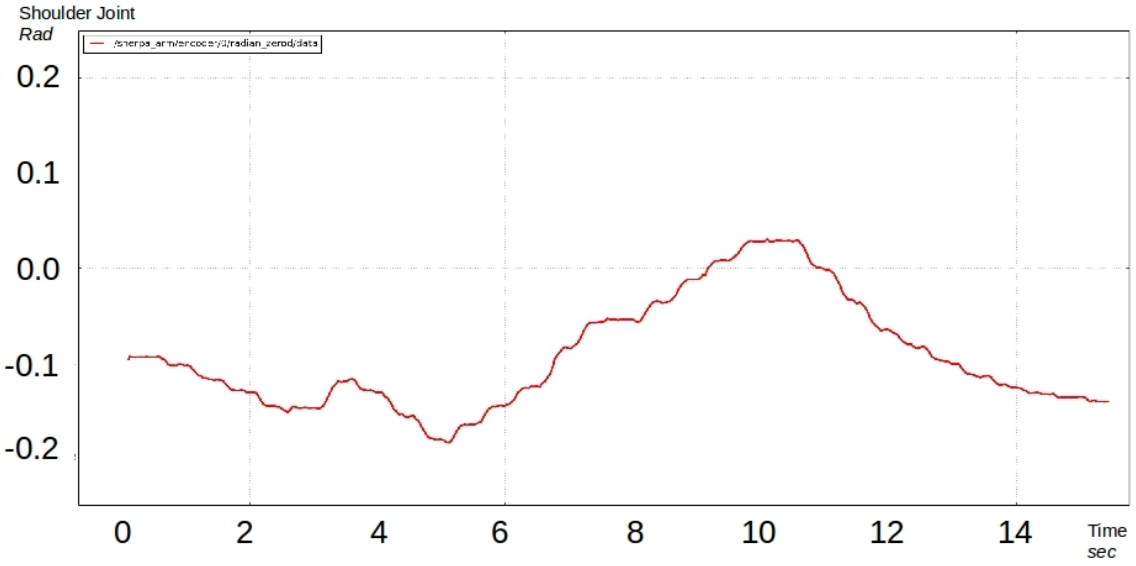


Figure 5.17: Experimental result, arm's shoulder joint movement to track the wasp.

to ultimately achieve a more robust localization. Another important advantage of the presented approach is the possibility to integrate heterogeneous players with different abilities and sub-tasks.

In comparison to most of the modern path planning algorithms available in the literature, the proposed pathfinder algorithm is not complete. As seen in Figure 5.15b, a considerable part of the space is not being searched and in case of complicated maze-like environments the algorithm fails to find a solution. However this algorithm is sufficient to solve an unstructured outdoor navigation problem where a complex obstacle pattern is not usually the case. Instead, the main issue in such an environment is the unevenness of the ground which we addressed by incorporating the chassis simulator in the cost function. In summary we claim that the proposed pathfinder platform for implementation of cooperative navigation suggests a trade off among completeness, ease of implementation and computational complexity. Furthermore, as the cooperation in the proposed algorithm is only defined in the cost function, our system can be easily extended to be implemented on a state lattice path planner platform that uses any standard path finding search algorithm such as Dijkstra, A-star, etc.

5.3 Battery exchange operation

Overall architecture of the SHERPAsystem was presented in section 1.2. Yet the experimental results of the SHERPAbattery exchange operation is to be presented. As mentioned in section 1.2.1, the execution of a collaborative plan TST, *i.e.* the executor, is the second phase of the delegation process. When the battery replacement executor, shown in Figure 5.18, is requested, it is delegated to the *GRA*[5] agent and expands into several nodes that are delegated to the *arm*, *wasp*, *rover*, and *Sherpa box* agents.

The *change_batt* executor is typically triggered by either the human operator, or fully autonomously by the *wasp*, which will request the battery exchange after landing when its power is running low. After accepting the delegation, the *GRA* expands the executor and commands the *arm* to localize the *wasp* through the *find_wasp* executor. This executor triggers the *arm* to search for the *wasp* with its end-effector camera through coordinated motions of the *arm* and the *rover*, after which the location and pose of the *wasp* is published to the other agents. Next, the *wasp* is commanded to disarm its motors, and the *GRA* to dock the *wasp*. The *dock_wasp* executor then in turn expands into the *move_to*, *pick*, and *dock* executors, which command the *rover* to approach the *wasp*, and the *arm* to grasp and move the *wasp* into the docking positing on the *Sherpa box*. The following *lock_wasp*, *switch_batt*, and *release_wasp* executors are delegated to the *Sherpa box*, and lock the *wasp* in the docking position, replace the depleted battery for a charged one, and release the *wasp* respectively. Finally the *wasp* is deployed again by placing it on the ground, and moving the *GRA* away from it, such that the *wasp* and *GRA* can continue their respective missions.

When the *GRA* is delegated to exchange the landed *wasp*'s batteries, the *arm* determines its precise position relative to the rover. This is happens by putting the arm in specific pose, *i.e.* "scan pose", in which the camera installed on the end effector has a proper field of view. Figure 5.19a demonstrate the *scan pose* considered for the detection phase.

Furthermore, detection of the special markers that are placed on the mechanical interface of the *wasp* is shown in Figure 5.19b where the still image of the camera feed is overlaid with the detected location of the marker, and the virtual collision object of the plan scene. The virtual drone is included the plan scene of the arm, as shown in Figure 5.20, and published.

The *rover* then plans a collision-free trajectoryand approaches the *wasp*, such that the wasp is located within the arm's workspace and can easily be grasped. Figure 5.21 shows the rover and wasp locations in the map generated by the rover, and the rover's trajectory.

Figure 5.22 shows the arm docking and deploying the wasp as a sequence of still images. After approaching the wasp, the arm picks up, grasps, and docks it on the Sherpa box, where the wasp's battery is exchanged according to Section 3.3. The *arm* then places the wasp back on the ground and the *rover* moves away from the wasp so that it can continue its mission with a replenished battery. The tunable passive compliance of the arm has proven to be

[5] Ground Rover and Arm

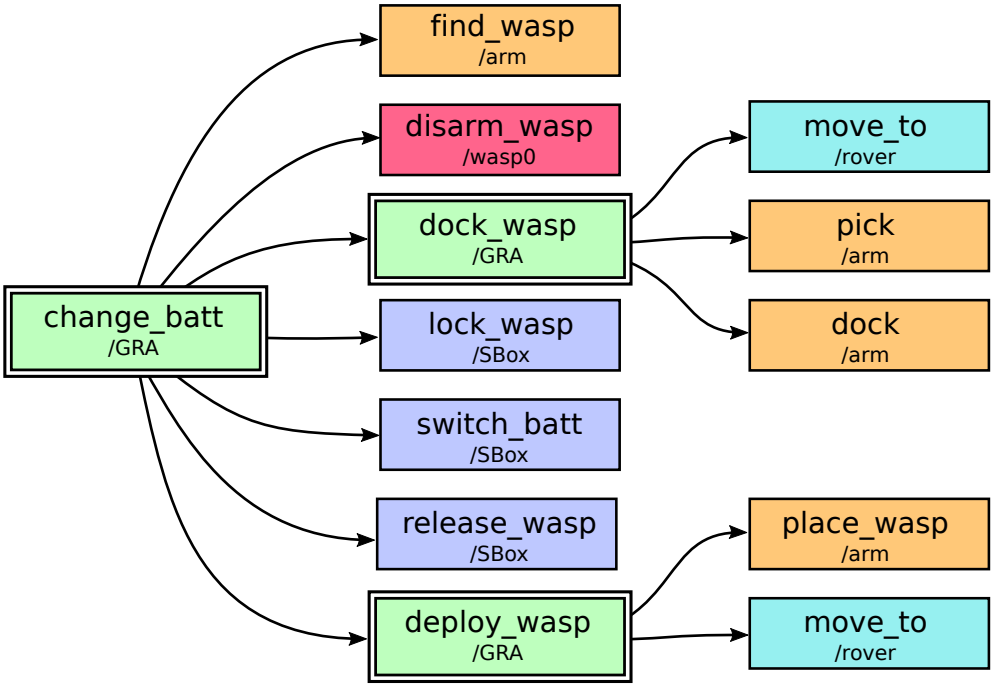
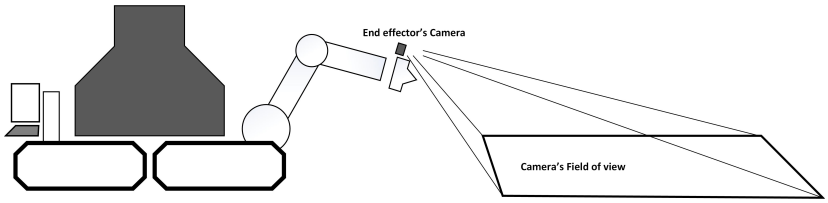


Figure 5.18: The battery change executor (expanded collaborative plan TST) is delegated to the *GRA* agent. It expands into other executors, which are delegated to the *arm*, *wasp*, *rover*, and *Sherpa box* agents. Internal nodes of the executor represent control statements, leaf nodes represent domain specific tasks. The different colors denote which agent the task is delegated to.

instrumental in the successful execution of the intricate interaction tasks. During grasping and docking of the *wasp*, the arm's compliance in the shoulder and wrist joints is controlled to make the operation more robust and reliable.

Lastly, Figure 5.23 gives an overview of the execution of the battery exchange by presenting key variables of the *rover*, *arm*, and *Sherpa box* over the course of the operation.



(a) Scan pose of the arm.

(b) Camera's view.

Figure 5.19: The wasp is detected from the video feed of the arm's camera by means of the markers placed on the interface. The position of the markers is denoted by the two circles, while the orientation is shown through the superimposed coordinate frames. The virtual model of the wasp (shown in green) is also overlaid onto the live video stream.

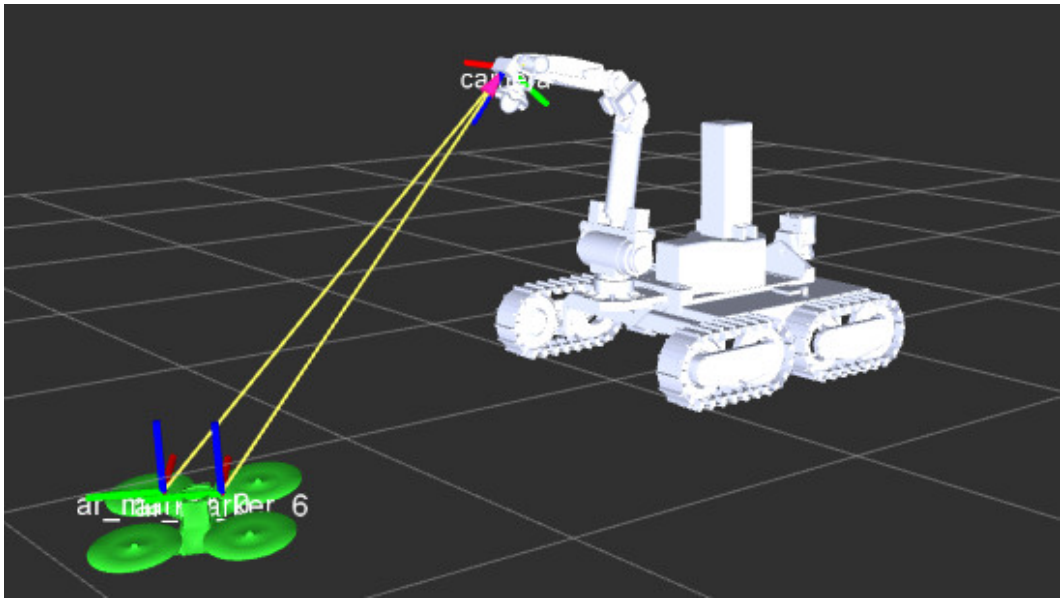


Figure 5.20: The virtual collision object of the wasp is placed in the plan scene of the arm once it is detected. The wasp's position and orientation are also published to the other agents.

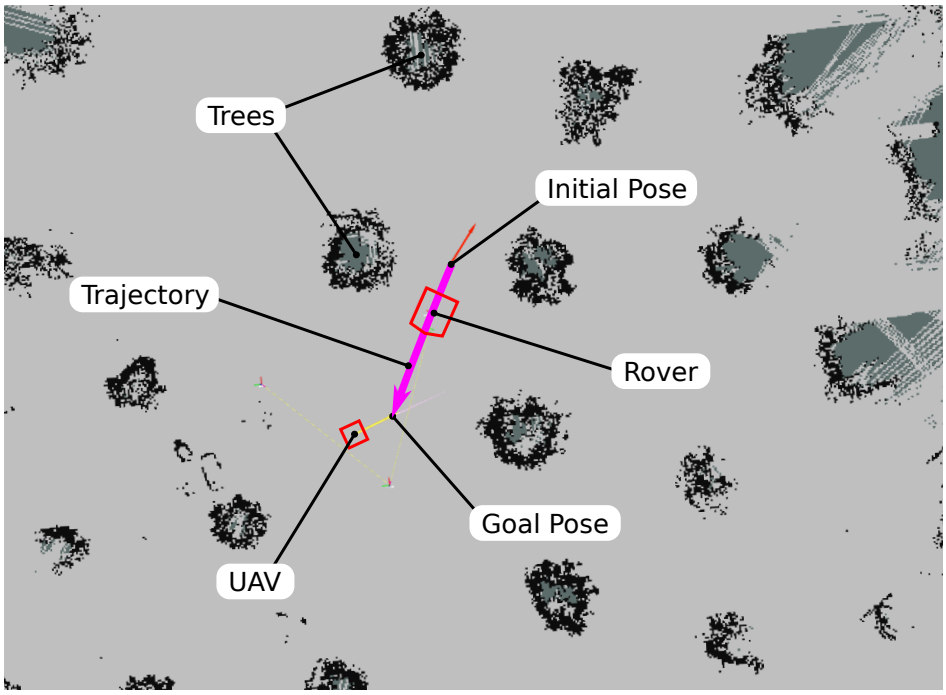
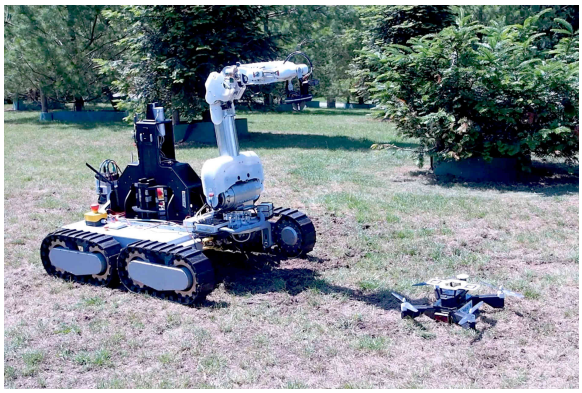
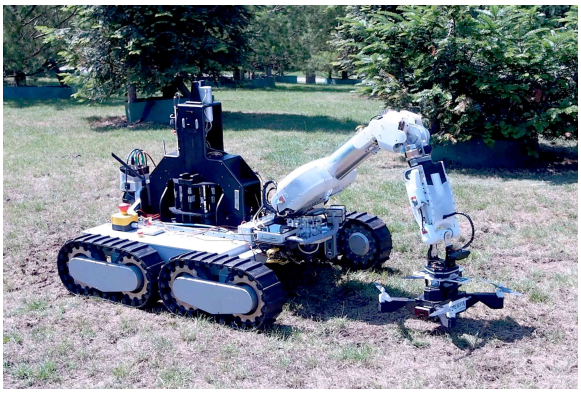


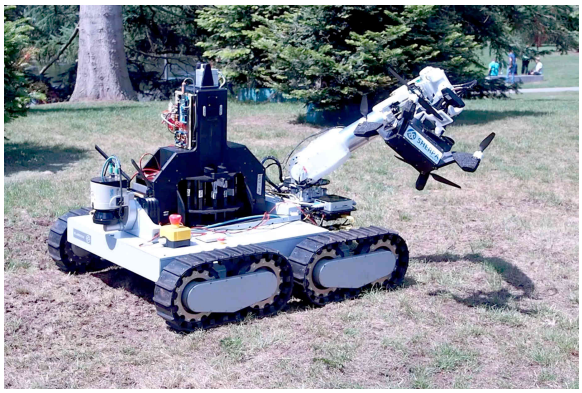
Figure 5.21: The map generated by the rover during the experiments. Free cells are shown in light gray, occupied cells in black, and unknown cells in dark gray. The small trees in the rover's environment are clearly visible as obstacles. The map shows the initial pose of the rover and its goal pose, at an offset from the UAV position, as well as the generated trajectory connecting the two.



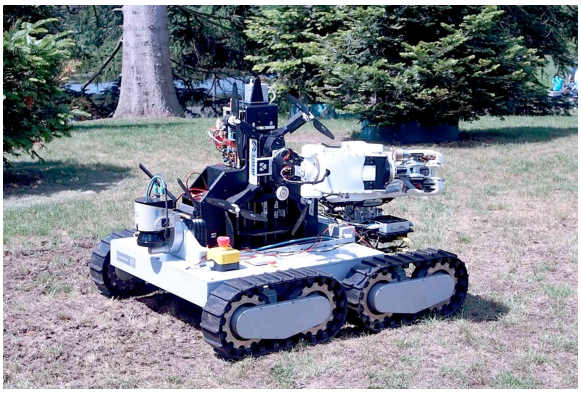
(a) Scan and approach



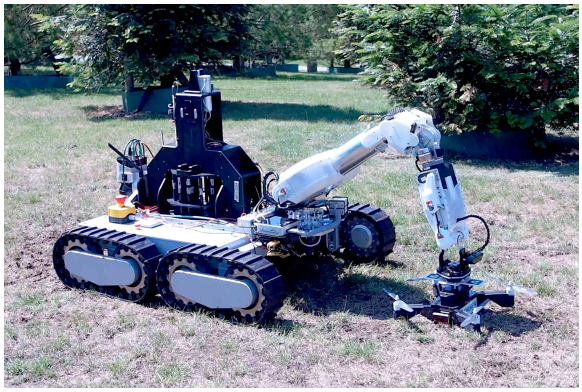
(b) Grasp



(c) Docking



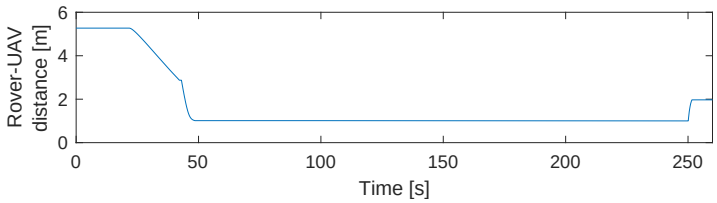
(d) Battery exchange



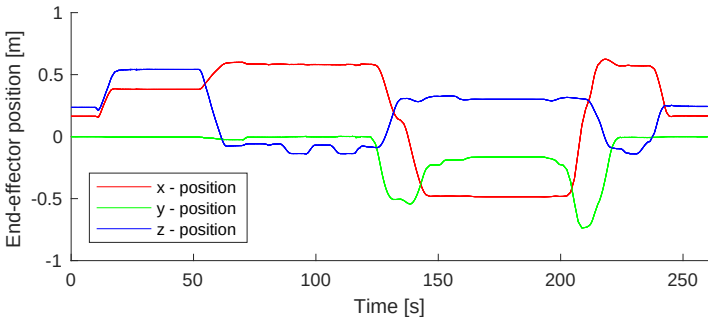
(e) Deploy

(f) Retreat

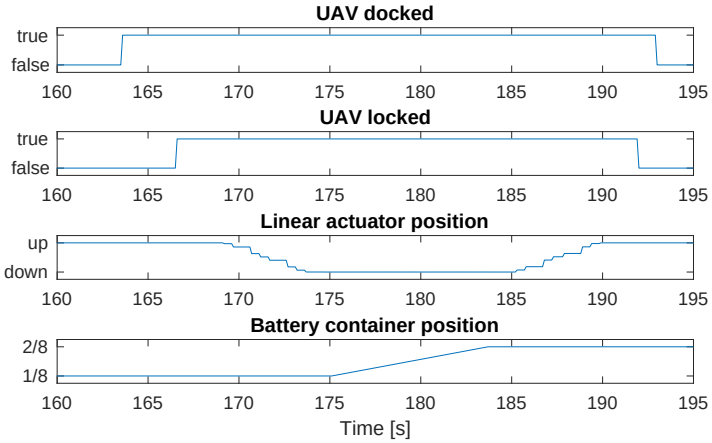
Figure 5.22: After the ground rover approaches the landed *wasp* (Figure 5.22a) the arm picks it up (Figure 5.22b) and docks it on the Sherpa box (Figure 5.22c) where its battery is exchanged (Figure 5.22d). The arm then places it back on the ground (Figure 5.22e) and retreats away from the *wasp* (Figure 5.22f).



(a) Distance between *rover* and *wasp*.



(b) End-effector position of the *arm*.



(c) Status of the *SBox*.

Figure 5.23: Overview of the battery exchange operation through the status of the involved agents. Figure 5.23a shows the distance between the *rover* and *wasp*, as the *rover* approaches it during the execution of the *dock_wasp*-executor, and moves away from the *wasp* along the trajectory generated by the rover's path planner during the *deploy_wasp*-executor. Figure 5.23b shows the end-effector position of the robotic *arm*. The *arm* moves first into a scanning pose during *find_wasp* and the start of *dock_wasp*, before it grasps and docks the *wasp*, and places it back on the ground during the *deploy_wasp* executor. Figure 5.23c shows the status of the *Sherpa box*. The *Sherpa box* registers the *wasp* as docked when the *arm* successfully places the *wasp* on the *Sherpa box*. The *wasp* is then locked by the docking clamps while the linear actuator moves the empty battery into the battery container, which then moves a charged battery into position that is inserted into the *wasp* by the linear actuator.

Chapter 6

Conclusion

This thesis focused on development of the Sherpa ground rover introduced by the SHERPA project. The SHERPA project seeks to present the values that a heterogeneous robotic team can add to a search and rescue mission in the Alps. To this purpose, the project introduces a set of particular quadrotors, called "wasp", that helps the rescuer to increase their awareness of the situation quickly and efficiently. However, the limited power capacity and high energy consumption of the wasps limit the autonomy of these robots. To address this problem the Sherpa ground rover has been introduced by the SHERPA project. Together with the Sherpa robotic arm and Sherpa box, the ground rover provides a mobile power replenishment station for the wasp.

In this document, the architecture of the system understudy was motivate. To provide a world model and organize different automation tasks among the multi-robot system of the SHERPA, the SHERPA system is introduced. Here, therefore, different elements of the SHERPA system were presented. Moreover, the two important SHERPA software, *i.e.* "Delegation framework" and "Sherpa World Model", were explained.

Navigation on an unstructured outdoor environment implies challenging localization, mapping and planning tasks. Thus, a comprehensive overview of the modern localization, mapping and planning methods were presented. The basics and characteristics of different localization strategies were studied. Furthermore, to perform an accurate navigation, a discrete event planner platform able to consider kinematics and dynamics of the robot was investigated.

The Sherpa rover, Sherpa arm and Sherpa box are three principal elements of the wasps' mobile power replenishment station which are to be integrated in one system. Therefore, the hardware of the ground rover, Sherpa arm and Sherpa box were presented and the specification and capabilities of the hardware of the systems were studied.

Moreover, integration of the Sherpa ground rover and Sherpa arm from the theoretical point of view were studied. To this purpose, firstly modelling of the rover and the arm were

covered. Furthermore, the control issues of the robotic arm were presented and addressed. Lastly, the modelling and control of the integrated system, as a mobile manipulator, were presented.

Finally benchmarks of the SHERPA project concerning the Sherpa rover were addressed. To this purpose, two target following strategies were presented. In the first case, a fix vision based sensor on the rover were considered to provide the target localization. Then a controller was designed to reject the disturbance while respecting the state and actuation constraints. In the second case, the vision sensor was considered on the arm and the human following were provided through a cooperation between the arm and the rover. Finally, the battery exchange operation experiment were presented.

Bibliography

[1] F. Fraundorfer and D. Scaramuzza, "Visual odometry: Part ii: Matching, robustness, optimization, and applications," *IEEE Robotics & Automation Magazine*, vol. 19, no. 2, pp. 78–90, 2012.

[2] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.

[3] Sherpa project offcial website. [Online]. Available: http://www.sherpa-project.eu/

[4] Sherpa page on eu community research and development information service(cordis). [Online]. Available: http://cordis.europa.eu/project/rcn/106964_en.html

[5] The sherpa box. [Online]. Available: http://www.sherpa-project.eu/sherpa/content/sherpa-box

[6] The sherpa animals. [Online]. Available: http://www.sherpa-project.eu/sherpa/content/sherpa-animals

[7] B. D. Song, J. Kim, J. Kim, H. Park, J. R. Morrison, and D. H. Shim, "Persistent UAV service: an improved scheduling formulation and prototypes of system components," in *International Conference on Unmanned Aircraft Systems*, 2013, pp. 915–925.

[8] R. Godzdanker, M. J. Rutherford, and K. P. Valavanis, "Islands: a self-leveling landing platform for autonomous miniature UAVs," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2011, pp. 170–175.

[9] D. R. Dale, "Automated ground maintenance and health management for autonomous unmanned aerial vehicles," Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2007.

[10] Y. Mulgaonkar, "Automated recharging for persistence missions with multiple micro aerial vehicles," Master's thesis, Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, 2012.

[11] F. P. Kemper, K. A. O. Suzuki, and J. R. Morrison, "UAV consumable replenishment: design concepts for automated service stations," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1, pp. 369–397, 2011.

[12] N. K. Ure, G. Chowdhary, T. Toksoz, J. P. How, M. A. Vavrina, and J. Vian, "An automated battery management system to enable persistent missions with multiple aerial vehicles," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 1, pp. 275–286, 2015.

[13] K. A. Swieringa, C. B. Hanson, J. R. Richardson, J. D. White, Z. Hasan, E. Qian, and A. Girard, "Autonomous battery swapping system for small-scale helicopters," in *IEEE International Conference on Robotics and Automation*, 2010, pp. 3335–3340.

[14] K. A. O. Suzuki, P. Kemper Filho, and J. R. Morrison, "Automatic battery replacement system for uavs: Analysis and design," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1, pp. 563–586, 2012.

[15] K. Fujii, K. Higuchi, and J. Rekimoto, "Endless flyer: a continuous flying drone with automatic battery replacement," in *IEEE International Conference on Ubiquitous Intelligence & Computing and IEEE International Conference on Autonomic and & Computing*, 2013, pp. 216–223.

[16] F. Bourgault, T. Furukawa, and H. F. Durrant-Whyte, "Decentralized bayesian negotiation for cooperative search," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004, pp. 2681–2686.

[17] J. Gancet, G. Hattenberger, R. Alami, and S. Lacroix, "Task planning and control for a multi-UAV system: architecture and algorithms," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 1017–1022.

[18] P. Doherty, J. Kvarnström, P. Rudol, M. Wzorek, G. Conte, C. Berger, T. Hinzmann, and T. Stastny, *A Collaborative Framework for 3D Mapping Using Unmanned Aerial Vehicles*. Springer, 2016, pp. 110–130.

[19] J. Kim, B. D. Song, and J. R. Morrison, "On the scheduling of systems of UAVs and fuel service stations for long-term mission fulfillment," *Journal of Intelligent & Robotic Systems*, vol. 70, no. 1, pp. 347–359, 2013.

[20] S. Blumenthal and H. Bruyninckx, "Towards a domain specific language for a scene graph based robotic world model," *arXiv preprint arXiv:1408.0200*, 2014.

[21] S. Blumenthal, N. Hochgeschwender, E. Prassler, H. Voos, and H. Bruyninckx, "An approach for a distributed world model with qos-based perception algorithm adaptation," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 1806–1811.

[22] S. Blumenthal, B. Brieber, N. Huebel, F. Yazdani, M. Beetz, and H. Bruyninckx, "A case study for integrating heterogeneous knowledge bases for outdoor environments," in *Integrating Multiple Knowledge Representation and Reasoning Techniques in Robotics (MIRROR-16)*, 2016.

[23] E. Barrett, M. Reiling, G. Barbieri, M. Fumagalli, and R. Carloni, "Mechatronic design of a variable stiffness robotic arm," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.

[24] "MoveIt! motion planning framework," http://moveit.ros.org/, accessed: 2010-08-01.

[25] G. Grisettiyz, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 2432–2437.

[26] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[27] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, November 2011.

[28] D. F. Wolf and G. S. Sukhatme, "Semantic mapping using mobile robots," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 245–258, 2008.

[29] A. Pronobis, "Semantic mapping with mobile robots," Ph.D. dissertation, KTH Royal Institute of Technology, 2011.

[30] B. Kuipers and Y.-T. Byun, "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations," *Robotics and autonomous systems*, vol. 8, no. 1, pp. 47–63, 1991.

[31] M. Pivtoraiko and A. Kelly, "Efficient constrained path planning via search in state lattices," in *International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, 2005, pp. 1–7.

[32] R. A. Knepper and A. Kelly, "High performance state lattice planning using heuristic look-up tables." in *IROS*, 2006, pp. 3375–3380.

[33] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 4889–4895.

[34] M. Rufli, D. Ferguson, and R. Siegwart, "Smooth path planning in constrained environments," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 3780–3785.

[35] J. Borenstein and L. Feng, "Measurement and correction of systematic odometry errors in mobile robots," *IEEE Transactions on robotics and automation*, vol. 12, no. 6, pp. 869–880, 1996.

[36] K. W. Eure, C. C. Quach, S. L. Vazquez, E. F. Hogge, and B. L. Hill, "An application of uav attitude estimation using a low-cost inertial navigation system," 2013.

[37] C. C. Ward and K. Iagnemma, "Model-based wheel slip detection for outdoor mobile robots," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 2724–2729.

[38] J. Yi, J. Zhang, D. Song, and S. Jayasuriya, "Imu-based localization and slip estimation for skid-steered mobile robots," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. IEEE, 2007, pp. 2845–2850.

[39] A. Censi, "An icp variant using a point-to-line metric," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 19–25.

[40] J. Minguez, F. Lamiraux, and L. Montesano, "Metric-based scan matching algorithms for mobile robot displacement estimation," in *IEEE International Conference on Robotics and Automation*, vol. 4. Citeseer, 2005, p. 3557.

[41] L. Montesano, J. Minguez, and L. Montano, "Probabilistic scan matching for motion estimation in unstructured environments," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE, 2005, pp. 3499–3504.

[42] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[43] Opencv camera calibration tool in ros. [Online]. Available: http://wiki.ros.org/camera_calibration

[44] G. Nützi, S. Weiss, D. Scaramuzza, and R. Siegwart, "Fusion of imu and vision for absolute scale estimation in monocular slam," *Journal of intelligent & robotic systems*, vol. 61, no. 1, pp. 287–299, 2011.

[45] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3d reconstruction in real-time," in *Intelligent Vehicles Symposium (IV)*, 2011.

[46] B. Kitt, A. Geiger, and H. Lategahn, "Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme," in *Intelligent Vehicles Symposium (IV)*, 2010.

[47] W. Förstner, "A feature based correspondence algorithm for image matching," *International Archives of Photogrammetry and Remote Sensing*, vol. 26, no. 3, pp. 150–166, 1986.

[48] C. G. Harris and J. Pike, "3d positional integration from image sequences," *Image and Vision Computing*, vol. 6, no. 2, pp. 87–90, 1988.

[49] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Computer vision–ECCV 2006*, pp. 430–443, 2006.

[50] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[51] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," *Computer vision–ECCV 2006*, pp. 404–417, 2006.

[52] M. Agrawal, K. Konolige, and M. Blas, "Censure: Center surround extremas for realtime feature detection and matching," *Computer Vision–ECCV 2008*, pp. 102–115, 2008.

[53] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[54] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustmentâĂŤa modern synthesis," in *International workshop on vision algorithms*. Springer, 1999, pp. 298–372.

[55] J. Aulinas, Y. R. Petillot, J. Salvi, and X. Lladó, "The slam problem: a survey." *CCIA*, vol. 184, no. 1, pp. 363–371, 2008.

[56] E. W. Nettleton, P. W. Gibbens, and H. F. Durrant-Whyte, "Closed form solutions to the multiple platform simultaneous localization and map building(slam) problem," in *PROC SPIE INT SOC OPT ENG*, vol. 4051, 2000, pp. 428–437.

[57] G. Huang, A. Rad, and Y. Wong, "Online slam in dynamic environments," in *Advanced Robotics, 2005. ICAR'05. Proceedings., 12th International Conference on*. IEEE, 2005, pp. 262–267.

[58] Y.-D. Jian and F. Dellaert, "ispcg: Incremental subgraph-preconditioned conjugate gradient method for online slam with many loop-closures," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 2647–2653.

[59] V. C. Guizilini and J. Okamoto, "Solving the online slam problem with an omnidirectional vision system," in *International Conference on Neural Information Processing*. Springer, 2008, pp. 1110–1117.

[60] P. Moutarlier and R. Chatila, "An experimental system for incremental environment modelling by an autonomous mobile robot," in *Experimental Robotics I*. Springer, 1990, pp. 327–346.

[61] ——, "Stochastic multisensory data fusion for mobile robot location and environment modeling," in *5th Int. Symposium on Robotics Research*, vol. 1. Tokyo, 1989.

[62] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous robots*, vol. 4, no. 4, pp. 333–349, 1997.

[63] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[64] T. Duckett, S. Marsland, and J. Shapiro, "Learning globally consistent maps by relaxation," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 4. IEEE, 2000, pp. 3841–3846.

[65] ——, "Fast, on-line learning of globally consistent maps," *Autonomous Robots*, vol. 12, no. 3, pp. 287–300, 2002.

[66] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit *et al.*, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *Aaai/iaai*, 2002, pp. 593–598.

[67] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 2014.

[68] J. Guivant, E. Nebot, and S. Baiker, "Autonomous navigation and map building using laser range sensors in outdoor applications," *Journal of robotic systems*, vol. 17, no. 10, pp. 565–583, 2000.

[69] D. H. D. Fox, W. Burgard, and S. Thrun, "A highly efficient fastslam algorithm for generating cyclic maps of large-scale environments from raw laser range measurements," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[70] Slma gmapping, implementation of particle filter based slam in ros. [Online]. Available: http://wiki.ros.org/slam_gmapping

[71] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, November 2011.

[72] Hector slam, implementation of slam in ros. [Online]. Available: http://wiki.ros.org/hector_slam

[73] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2564–2571.

[74] ——, "Orb: An efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2564–2571.

[75] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," *Computer Vision–ECCV 2010*, pp. 778–792, 2010.

[76] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *arXiv preprint arXiv:1610.06475*, 2016.

[77] W. T. Higgins, "A comparison of complementary and kalman filtering," *IEEE Transactions on Aerospace and Electronic Systems*, no. 3, pp. 321–325, 1975.

[78] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[79] M. Furci, "Mobile robots control and path planning strategies," Ph.D. dissertation, Faculty of Engineering, Alma Mater Studiorum - University of Bologna, 2016.

[80] E. Barrett, M. Fumagalli, and R. Carloni, "Elastic energy storage in leaf springs for a lever-arm based variable stiffness actuator," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 537–542.

[81] T. Yoshikawa, *Foundations of robotics: analysis and control*. MIT press, 1990.

[82] Y. Nakamura, *Advanced robotics : redundancy and optimization*. Addison-Wesley, 1991.

[83] T. F. Chan and R. V. Dubey, "A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 2, pp. 286–292, 1995.

[84] A. Isidori, "Nonlinear control systems ii," 2000.

[85] ASUS XtionPRO kinect sensor. https://www.asus.com/3D-Sensor/Xtion_PRO/.

[86] OpenNI Tracker kinect open source interface software. http://wiki.ros.org/openni_tracker.

[87] B. An, Z. Shen, C. Miao, and D. Cheng, "Algorithms for transitive dependence-based coalition formation," *IEEE Transactions on Industrial Informatics*, vol. 3, no. 3, pp. 234–245, 2007.

[88] A. Ghazikhani, H. R. Mashadi, and R. Monsefi, "A novel algorithm for coalition formation in multi-agent systems using cooperative game theory," in *Electrical Engineering (ICEE), 2010 18th Iranian Conference on*. IEEE, 2010, pp. 512–516.

[89] A. Kelly and B. Nagy, "Reactive nonholonomic trajectory generation via parametric optimal control," *The International Journal of Robotics Research*, vol. 22, no. 7-8, pp. 583–601, 2003.

[90] T. M. Howard and A. Kelly, "Optimal rough terrain trajectory generation for wheeled mobile robots," *The International Journal of Robotics Research*, vol. 26, no. 2, pp. 141–166, 2007.

[91] T. M. Howard, M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Model-predictive motion planning: Several key developments for autonomous mobile robots," *IEEE Robotics & Automation Magazine*, vol. 21, no. 1, pp. 64–73, 2014.

[92] J. Engwerda, *LQ dynamic optimization and differential games*. John Wiley & Sons, 2005.

[93] N. Jarrassé, T. Charalambous, and E. Burdet, "A framework to describe, analyze and generate interactive motor behaviors," *PloS one*, vol. 7, no. 11, p. e49945, 2012.

[94] K. E. Parsopoulos, M. N. Vrahatis *et al.*, "Particle swarm optimization method for constrained optimization problems," *Intelligent Technologies–Theory and Application: New Trends in Intelligent Technologies*, vol. 76, no. 1, pp. 214–220, 2002.

[95] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[96] I. I. Saito. Ar track alvar, open source marker tracker. [Online]. Available: http://wiki.ros.org/ar_track_alvar

[97] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf. Hector slam, open source ros software. [Online]. Available: http://wiki.ros.org/hector_slam

[98] Point cloud library, open source c++ library. [Online]. Available: http://wiki.ros.org/hector_slam

[99] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.