

Ivan Šinkarenko

Lunar Rover Motion Planning and Commands

School of Electrical Engineering

Department of Automation and Systems Technology

Thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science in Technology

Espoo, August 20, 2013

Instructor: Eric Halbach
Aalto University
School of Electrical Engineering

Supervisors: Professor Emeritus Aarne Halme
Aalto University
School of Electrical Engineering

Professor Thomas Gustafsson
Luleå University of Technology

Acknowledgements

I would like to express my deepest gratitude to all the people who provided me with the possibility to complete this Thesis. I thank my instructor, Eric Halbach, for his patience and help throughout the entire period of writing. Furthermore, I would also like to thank Tomi Ylikorpi and Professor Emeritus Aarne Halme for their guidance. Special gratitude I give to Veikko Immonen for his assistance and help. I really enjoyed the discussions with Laura Mendoza, my English teacher, and appreciate her contribution into my language skills. I thank my fellow SpaceMasters, who made these two years joyful and unforgettable. In addition, I appreciate all the support and patience provided by my colleagues at DevBridge Group. Finally, nobody has given me as much encouragement and inspiration as my family, who deserved more than just my special thanks.

Espoo, August 20, 2013

Ivan Šinkarenko

Author:	Ivan Sinkarenko	
Title of the thesis:	Lunar Rover Motion Planning and Commands	
Date:	August 20, 2013	Number of pages: 85
Department:	Automation and Systems Technology	
Programme:	Master's Degree Programme in Space Science and Technology	
Professorship:	Automation Technology (AS-84)	
Supervisors:	Professor Emeritus Arne Halme (Aalto) Professor Thomas Gustafsson (LTU)	
Instructor:	Eric Halbach	
<p>Space exploration is moving forward and one of the topics currently being researched is mining. The objective of this thesis is to design and develop software for the autonomous navigation of a wheeled rover that is being built for NASA's Lunabotics Mining Competition. The motion control system is a crucial component of a planetary rover system and its implementation heavily depends on the chassis configuration. The configuration of the rover enables us to use three steering modes: Ackermann, Point-turn and Crab steering. The implementation takes advantages of all the modes and involves algorithms for path planning, path smoothing and path following. In addition, the system offers a feature of automatic steering mode selection. The system can be tuned and controlled by the cross-platform application specifically developed for this purpose. The performance of the implemented system is analyzed by testing in a simulator with a realistic physics engine and 3D visualization capabilities. Our conducted tests confirm that the system is sufficient in the framework of the Lunabotics Mining Competition.</p>		
Keywords:	motion, planning, control, lunar, rover, lunabotics, mining	

Contents

1	Introduction	1
1.1	Competition Rules	2
1.2	Project Description	4
1.3	Outline	5
2	Related Work	6
2.1	Overview of Ground Locomotion	6
2.2	Wheeled Locomotion	9
2.2.1	Wheel Types	11
2.2.2	Suspension	12
2.2.3	Holonomicity	13
2.3	Steering	14
2.3.1	Skid Steering	14
2.3.2	Explicit Steering	15
2.3.3	Articulated Steering	15
2.4	Terrain	17
2.4.1	Regolith Properties	17
2.4.2	Wheel-Soil Interaction	18
2.5	Driving Path	19
2.5.1	Path Generation	19
2.5.2	Path Selection	20
2.5.3	Path Following	21
3	Implementation	24
3.1	Rover Hardware	24
3.1.1	Chassis	24
3.1.2	Electronics	28
3.2	Path Planning Algorithms	28
3.2.1	Search Algorithms Overview	29

3.2.2	Potential Fields	33
3.2.3	Sampling-Based Algorithms	33
3.2.4	Path Planning Solution	34
3.2.5	Path Smoothing	36
3.3	Path Following Algorithms	41
3.3.1	Control Overview	41
3.3.2	PID Controller	42
3.3.3	Ackermann Path Following	44
3.3.4	Point-turn Path Following	46
3.3.5	Crab Path Following	48
3.3.6	Steering Mode Selection	49
3.4	Software Packages	50
3.4.1	Onboard Software Package	51
3.4.2	Graphical User Interface	52
3.4.3	Communication	57
3.5	Simulation	60
3.5.1	Stage Simulator	60
3.5.2	Gazebo Simulator	61
4	Simulation Results	68
4.1	LunArena	68
4.2	Slope	71
4.3	Lunar Surface	72
5	Conclusions and Further Work	74
	References	77
A	Internal Communication of the ROS Nodes	I
B	Shell Scripts of the Onboard Software	III

List of Tables

2.1	Performance evaluation metrics and indexes.	18
3.1	The properties of the individual links of the Gazebo rover model.	65
3.2	The properties of the individual joints of the Gazebo rover model.	65
3.3	The friction coefficients of the polyurethane wheel layer on various surfaces.	65

List of Figures

1.1	LunArena (isometric view).	3
2.1	Simplified kinematics of the Scarab suspension demonstrating inching, with indication of the suspension and wheel motions. . .	8
2.2	Influence of wheel-soil interaction on the wheel arrangement. . .	10
2.3	Influence of wheel-soil terramechanic on the wheel shapes. . . .	10
2.4	Castor wheel.	11
2.5	Mecanum wheels implemented on URANUS omni-directional mo- bile robot.	11
2.6	Ball wheels implemented on MoVille car concept.	12
2.7	The Rocker-bogie suspension of MER.	13
2.8	The articulated suspension of the Marsokhod rover.	13
2.9	Forward-backward maneuver for a non-holonomic wheeled mobile robot.	14
2.10	Skid steering Ackermann and Point-turn examples.	14
2.11	Explicit steering Ackermann and Point-turn examples.	15
2.12	Articulated steering example.	16
2.13	Crab control mode example.	17
2.14	Example terrain assessment and path selection by MER vehicle.	20
2.15	a) Bicycle model. b) All-wheel dynamics model.	22
2.16	Thrust-Cornering Characteristic Diagram.	22
3.1	Three of six 20 inch aluminum wheels on an engineering model of the MSL rover.	25
3.2	The CAD model of the chassis of the rover.	26
3.3	The CAD model of the wheel module.	27
3.4	Mechanical constraints of the steering motors.	27
3.5	The example of the obstacles inflated in the occupancy grid of the LunArena map.	29

3.6	a) Expansion of a node in the 4-directional search. b) Expansion of a node in the 8-directional search.	30
3.7	a) The typical transitions allowed from a node in a uniform grid. b) Field D* available transition in a uniform grid.	31
3.8	Paths produced by classic grid-based planners and Field D* in a 150 × 60 uniform resolution grid.	31
3.9	Expansion of a node in raD* 32 search.	32
3.10	The example of a potential field.	33
3.11	The comparison of the outputs of original and modified A* algorithms.	35
3.12	Construction of the linear Bézier curve.	36
3.13	Construction of the quadratic Bézier curve.	37
3.14	Construction of the cubic Bézier curve.	37
3.15	A quadratic Bézier curve created by the proposed algorithm. . .	38
3.16	Tetragonal concave boundary approach.	39
3.17	Path smoothing analysis conducted by Choi et al.	40
3.18	The smooth trajectory constructed by the tetragonal concave boundary method.	40
3.19	The block diagrams of different control systems.	42
3.20	The block diagram of a PID controller.	43
3.21	Path following using feedback control.	44
3.22	Curvature detection for feedforward control.	45
3.23	The model of the Ackermann steering system.	45
3.24	Control Bicycle model and the ICR point.	46
3.25	The state diagram of the path following routine in Point-turn steering mode.	47
3.26	Point-turn approach of the all-wheel steered rover.	48
3.27	Crab control of the all-wheel steered rover.	49
3.28	Calculation of the maximum curvature.	50
3.29	The main application window of the robot interaction software. .	53
3.30	The example of selecting multiple waypoints.	54
3.31	All-Wheel Control window.	55
3.32	Path following window.	56
3.33	Trajectory Analysis window.	57
3.34	The environment in the Stage simulator.	61
3.35	The six possible lower-pair joints.	62

3.36	The model of the project robot on the simulated lunar surface. .	66
4.1	Test environments created for the Gazebo simulator.	69
4.2	Ackermann steering in the LunArena environment.	70
4.3	Driving test in the LunArena environment.	71
4.4	Path correction test on a slope.	72
4.5	Driving on the Moon-like surface test.	73
A.1	The graph of nodes and topics of the onboard software package.	II

Abbreviations

ATHLETE	All-Terrain Hex-Limbed Extra-Terrestrial Explorer
BP-1	Black Point-1
CAD	Computer-Aided Design
CFM	Constraint Force Mixing
DARPA	Defense Advanced Research Projects Agency
DOF	Degrees of Freedom
DRCSim	DARPA Robotics Challenge Simulator
ESA	European Space Agency
GUI	Graphical User Interface
ICR	Instantaneous Center of Rotation
IMU	Inertial Measurement Unit
JPL	Jet Propulsion Laboratory
LER	Lunar Exploration Rover
KSC	Kennedy Space Center
LPA*	Lifelong Planning A*
LRV	Lunar Roving Vehicle
MER	Mars Exploration Rovers

MSL	Mars Science Laboratory
MSR	Mars Sample Return
NASA	National Aeronautics and Space Administration
ODE	Open Dynamics Engine
OGRE	Object-oriented Graphics Rendering Engine
PCBC	Piecewise Cubic Bézier Curves
PID	Proportional-Integral-Derivative
raD*	Rover Adapted D*
RGB-D	Red-Green-Blue-Depth
ROS	Robot Operating System
SDF	Simulation Description Format
SELENE	SELenological and ENgineering Explorer
SLAM	Simultaneous Localization and Mapping
STEM	Science, Technology, Engineering and Mathematics
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VoIP	Voice over Internet Protocol
XML	Extensible Markup Language

Chapter 1

Introduction

"We can only see a short distance ahead, but we can see plenty there that needs to be done."

- Alan Turing

Space exploration is moving forward and one of the topics currently being researched is mining. It has great potential and can become a cornerstone of space exploration. One of the possible applications is the mining of asteroids to extract useful minerals, such as iron, nickel, titanium, water and oxygen. It is a more attractive way of delivering materials to space stations and planetary bases than launching them from the Earth. This statement becomes more important when space exploration is moving further away from the Earth into deep space. Besides that, several companies have been created to focus on the topic of asteroid mining, for example, Planetary Resources. "There are near-limitless numbers of asteroids and more being discovered every year. More than 1,500 are as easy to reach as the Moon and are in similar orbits as Earth. Asteroids are filled with precious resources, everything from water to platinum" (Planetary Resources, 2013). Another possible application is the excavation of materials on site for building a planetary base. The idea of building a lunar base is quite old and has been considered in the early days of space exploration. Soviet Union has been developing the *Galaktika* project, which aimed to design a Soviet lunar base called *Kolumb*, in 1967 (Harvey, 2007). The proposal of the second Soviet lunar base *Zvezda* was delivered in 1974. On the United States side, there was project *Horizon* created in 1959 that focused on studying the

feasibility of creating a military outpost on the surface of the Moon by 1965 (Encyclopedia Astronautica, 2013). Another project involving construction of an underground base was called *Lunex* and aimed to launch people to the Moon even before Apollo. In the project proposal (U.S. Air Force Systems Command, Space Systems Division, 1961), the base was referred to as Lunar Expedition Facility and it was intended to support the expedition with the ability to be expanded for the military purposes in the future. Unfortunately, neither of the projects was accomplished. Nowadays, space agencies are coming back to the possibility of colonizing the Moon, as recent studies have shown that water might exist in noteworthy quantities at the lunar poles (Spudis and Lavoie, 2011). In 2006, Japan has revealed their plans to have a base on the Moon in 2030. A year later, Russian space agency Roskosmos has declared that they are willing to build a lunar base in the timeframe of 2027-2032.

To speed up the development of the required technologies, some companies are creating challenges available for the vast amount of people to elaborate on solving problems for these missions. One of the widely known challenges is Google Lunar X Prize, which is organized by the X Prize Foundation and sponsored by Google. The goal of the competition is to build a lunar robot that could land on the Moon and travel on its surface. The main objective is to travel 500 m in the lunar surface and transmit the images and video from the rover. Another attractive challenge, which is offered by the National Aeronautics and Space Administration (NASA), is Lunabotics Lunar Mining Competition, held in the Visitor Complex of the Kennedy Space Center (KSC), Florida (National Aeronautics and Space Administration, 2012a). The detailed explanation of the rules of this competition is collected in Section 1.1.

1.1 Competition Rules

NASA's Lunabotics Mining Competition is designed for university level students and is intended to promote interest in areas of Science, Technology, Engineering and Mathematics (STEM). Beforehand, teams have to create a teleoperated or autonomous robotic vehicle capable of excavating the surface of the simulated Moon. There are several categories of the competitions in this event, including onsite mining, outreach project, systems engineering paper, slide presentation

and team spirit. In the onsite mining category, teams are obliged to excavate at least 10 kg of regolith to qualify. In addition, they collect points based on several factors listed below.

- Total mass of collected regolith
- Dust-tolerant design
- Dust-free operation
- Communication bandwidth
- Vehicle mass
- Power consumption
- Level of autonomy

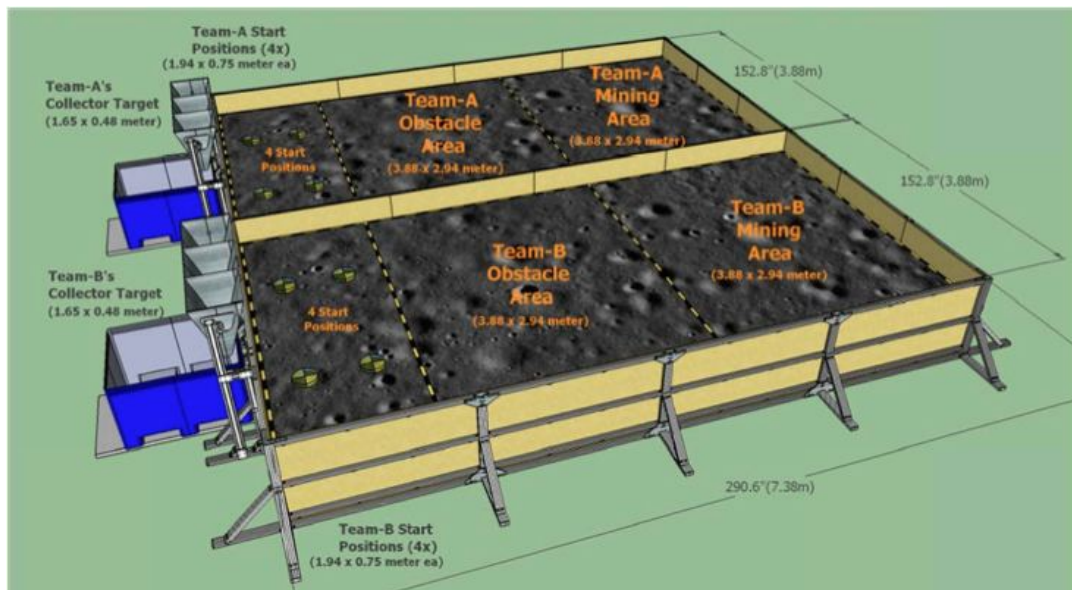


Figure 1.1: LunArena (isometric view). (Source: National Aeronautics and Space Administration (2012a))

Total vehicle mass is allowed to be under 80 kg without a penalty. If it is exceeded, the team gets negative points per kilogram of overweight. Furthermore, the initial dimensions of the vehicle must not exceed 1.5 m length \times 0.75 m width \times 0.75 m height. A rover is allowed to expand after the start, but it may not exceed a 1.5 m height. The simulated area is filled with Black Point-1 (BP-1) lunar regolith simulant and enclosed in a box called *LunArena*. Teams

are required to perform two competition attempts and each attempt occurs with two teams that compete at the same time. During each of the attempts, teams earn cumulative points which are summed up to the total score in the end. Each rover starts in one of four positions, selected arbitrarily, in the starting area. Excavation is allowed only in the mining area, which is separated from the starting area by series of obstacles. The objective of each rover is to get to the mining area avoiding the obstacles, excavate the regolith and bring it back to the starting area. After, it has to dump the regolith into so-called *LunaBin* that is placed 0.5 m above the ground. The obstacle area contains three obstacles placed on top of the compressed BP-1 surface. Each of the obstacles has a diameter of approximately 20 to 30 cm and a mass of 7 to 10 kg. Additionally, there are two craters in the obstacle area no wider or deeper than 30 cm. The sketch of LunArena is shown in Figure 1.1.

1.2 Project Description

This thesis was conducted as a part of the Joint European Master programme in Space Science and Technology. It is based on the lunar rover project specifically being developed for the Lunabotics Mining Competition at Aalto University. The practical part of this thesis was the development of a motion planning and control subsystem of the rover. Since the rover hardware was being built at the same time, the software development was based on simulations. Chapter 3 provides algorithms and simulation description. Along with that, the review of the background study has been conducted, and certain techniques have been applied within the motion subsystem. The motivation for this work is the importance of creating simple and traversable paths and selecting the appropriate steering modes in order to reduce slip and ensure affordable mission time along with the accurate telemetry data. It is proven by the experience of Lindemann and Voorhees (2005), since trajectory errors of Mars Exploration Rovers (MER) described in their paper were directly related to a slope height, slope angles and wheel sinkage.

1.3 Outline

Chapter 2 reviews possible locomotion configurations for lunar/planetary rovers, including different sets of wheels and suspensions. It also provides information about terrain properties and reviews the ways of path generation, selection and following.

Chapter 3 provides an overview of the hardware platform. It also describes the implemented algorithms and the background theory required to understand them. Finally, it presents the software developed for the project and the simulation tools used to test and analyze the implemented algorithms.

Chapter 4 presents the results of various simulation cases.

Chapter 5 summarizes the conducted work and perspectives of this project and overviews the improvements which can be done in the future.

Chapter 2

Related Work

This chapter demonstrates the available configurations for the mechanical structure of different lunar/planetary rovers and the algorithms used in the motion control of those rovers. Section 2.1 discusses the means for the ground motion of the rovers. Section 2.2 presents the more detailed review of the wheeled locomotion. Section 2.3 reviews the steering modes commonly used in wheeled rovers. Section 2.4 discusses the properties of the lunar regolith and its effects on the wheeled motion. Finally, Section 2.5 introduces ways to generate and follow a path and gives the examples of implementations in current rovers.

2.1 Overview of Ground Locomotion

The locomotion subsystem is a very important part of a lunar/planetary rover and should be considered carefully. Particularly, on the Moon, ground force is six times weaker than on the Earth, consequently locomotion mechanism should ensure low ground pressure in order to avoid sinkage and slip, as stated in (Nishida et al., 2011). Many types of locomotion exist in the nature and many of them have been adopted in robotics. Seeni et al. (2008) have classified locomotion types used for extraterrestrial surfaces as wheeled motion, tracks, legged motion and hybrid motion.

- Wheeled locomotion is a commonly used concept for terrestrial and planetary rovers. It ensures static stability with three or more wheels touching

the ground. The drawbacks of this system are sinkage and slip on loose soil and inability to overcome very steep slopes.

- Track locomotion provides static stability with only two crawlers. However, crawler mechanisms can be vulnerable to abrasive particles of planetary soil or lunar regolith (Tao et al., 2006). Also, if the crawler belt is made of rubber, when exposed to vacuum it can be damaged by outgassing and large temperature amplitude (Nishida et al., 2011). In addition, significant power dissipation is a drawback of this mechanism.
- Legged mechanisms are sophisticated and usually are avoided in planetary missions, where robustness is the primary point. However, scientists keep designing concepts with legged locomotion systems as can be seen in (Seeni et al., 2008). Legged motion includes actions, such as running, jumping and walking. In addition, walking can be static or dynamic. Static walking assumes that anytime three or more legs are in contact with the ground, keeping the centre of gravity in the support area (Kyrki, 2012). On the contrary, dynamic walking relies on falling action, which means that less than three legs touches the ground at the same time. As a result, movement is performed due to gravitational forces.
- Hybrid systems combine the multiple locomotion types listed above. For example, Sherpa rover utilizes wheeled legs (Cordes et al., 2011). It takes advantage of features of the system called All-Terrain Hex-Limbed Extra-Terrestrial Explorer (ATHLETE) developed by the Jet Propulsion Laboratory (JPL). It is stated that legged motion is used to overcome large obstacles, whereas wheeled motion is used on flatter terrain. Another example of a hybrid system is a follow-on rover of the SELenological and ENgineering Explorer (SELENE) mission, described by Nishida and Wakabayashi in 2010, which uses crawler modules in the same way as wheels. Moreland et al. (2011) have studied a hybrid push-roll locomotion configuration called *inching*. This type is based on expanding and contracting the wheel base of the vehicle. Drawbar pull is increased approximately by 100% as proven by a drawbar pull test performed with Scarab rover. Moreland et al. have also pointed out that it is important to counter-rotate wheels "in synchrony with the expanding or contracting side frame; if they were to be locked they would rotate in the direction of vehicle displacement and thereby inducing slippage associated with a rotating wheel".

The hybrid locomotion vehicle can potentially overcome steeper slopes or carry payload of greater weight than the regular vehicles. Schematic of the inching motion of Scarab rover is shown in Figure 2.1. The ExoMars rover, developed by the European Space Agency (ESA), also has a hybrid locomotion system. It can crawl by moving one wheel at a time. Similarly, WorkPartner robot described by Ylonen and Halme (2002) can take advantage of hybrid locomotion. Ylonen and Halme refer to its motion as *rolking* (rolling-walking). Hakenberg (2008) has studied the capabilities of hybrid locomotion of the Marsokhod rover. This six-wheeled rover is able to *rolk* by contracting and expanding its front and rear axles.

Many lunar/planetary rover projects (e.g., Biesiadecki and Maimone (2006), Kulkarni et al. (2006), Seegmiller and Wettergreen (2011), Peynot and Lacroix (2003)) implement wheeled locomotion because of its robustness, reliability and simplicity. In fact, no successful space missions with legged or crawling locomotion mechanisms are known. Nevertheless, attempts to build a device which utilizes legged locomotion have been made, and the concepts can be found in (Seeni et al., 2008) and (Bares and Whittaker, 1990).

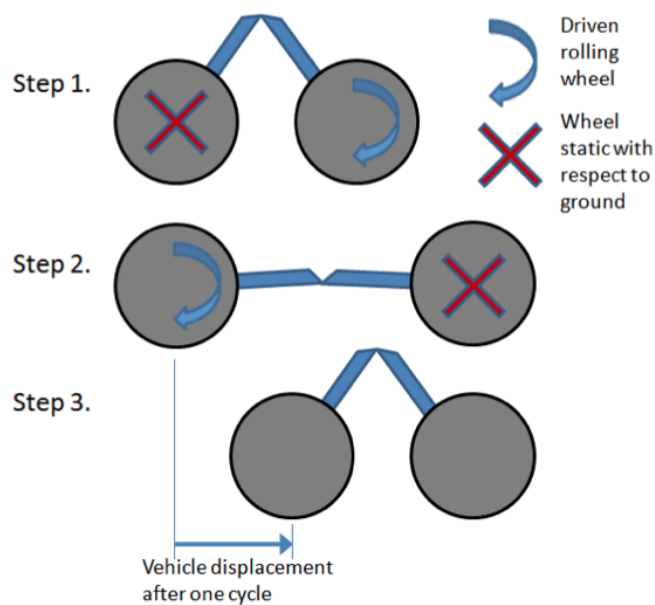


Figure 2.1: Simplified kinematics of the Scarab suspension demonstrating inching, with indication of the suspension and wheel motions. (Source: Moreland et al. (2011))

Fuke et al. (1995) have specified the following requirements as the key factors affecting locomotion design of a lunar exploration rover:

- **Environmental survivability**, that is, design issues, such as thermal control, lubricants containment.
- **Long term reliability**, since rovers usually are meant to operate from several months to several years.
- **Terrainability**, since a rover has to be able to overcome as many obstacles as possible without a priori knowledge.
- **Reduction of body oscillation**, since an exploration rover has to point its communication antennas very precisely.

2.2 Wheeled Locomotion

Zhang et al. (2008) have pointed out that there is no unified evaluation system for the design of wheeled planetary rovers. Usually, they are classified by the number of wheels.

- A two-wheel configuration is hard to imagine, but this concept has been proposed by Tao et al. (2006). This rover has "bilateral symmetry in configuration". The author considers using several such rovers in a junction to create the different wheel base configurations for different purposes.
- Four-wheel rovers can be found in many applications on Earth but are less popular in planetary exploration. The very well known four-wheel rover is Lunar Roving Vehicle (LRV) from Apollo mission. JPL Nanorover, described by Tao et al., also belongs to this group.
- A five-wheel configuration is an extraordinary concept and currently has been developed under SELENE-II mission on the Micro5 rover (Seeni et al., 2008).
- A six-wheel configuration is one of the most popular lunar exploration rover concepts (Zhang et al., 2008). It has also been extensively used in

JPL Martian missions. Their rover, Sojourner, was the first one to drive on Mars autonomously. However, it relied purely on reactive behavior and did not have any permanent terrain map, according to Bajracharya et al. (2008). MER rovers and Mars Science Laboratory (MSL) rover are the descendants of Sojourner and have inherited its six-wheel base (National Aeronautics and Space Administration, 2012b). Future mission, such as Mars Sample Return (MSR) also consider using six-wheeled rovers. Even an asymmetric configuration of a six-wheeled rover was proposed and studied by Du et al. (2010).

- An eight-wheel rover example is Lunokhod, developed in early days of planetary missions. Nowadays this configuration has little use.

Wheeled locomotion can sometimes be referred to by the wheel formula. Seeni et al. (2008) define wheel formula as *Wheel formula = Total no. of wheels \times no. of actuated wheels \times no. of steered actuated wheels*. For instance, the wheel formula of the MSL rover is $6 \times 6 \times 4$. Also, wheeled rovers can vary in wheel arrangement or wheel shapes. Possible configurations are shown in Figures 2.2 and 2.3.

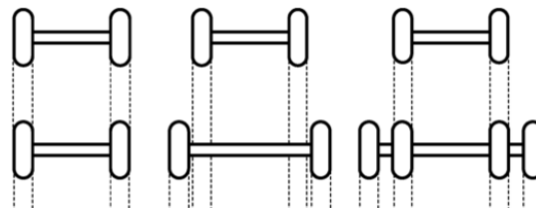


Figure 2.2: Influence of wheel-soil interaction on the wheel arrangement. (Source: Deng et al. (2011))

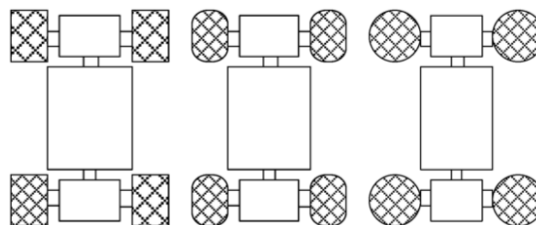


Figure 2.3: Influence of wheel-soil terramechanic on the wheel shapes. (Source: Deng et al. (2011))

2.2.1 Wheel Types

Kyrki (2012) has listed common wheel types used in field and service robotics. Robots can use combinations or distinct types of wheels, namely:

- **A fixed standard wheel**, which has two Degrees of Freedom (DOF): rotation around wheel axle and contact point.
- **A steerable standard wheel**, which is the same as the fixed standard wheel with steering capability.
- **A castor standard wheel**, which has three DOF (See Figure 2.4).
- **A Mecanum wheel**, developed by the Swedish company Mecanum, is omnidirectional (See Figure 2.5).
- **A ball wheel**, which is omnidirectional (See Figure 2.6).



Figure 2.4: Castor wheel.



Figure 2.5: Mecanum wheels implemented on URANUS omni-directional mobile robot. (Source: Carnegie Mellon University (2011))

Furthermore, some planetary rovers use wheels with grousers. According to Sutoh et al. (2012), recent studies have proved that grousers increase traveling performance of planetary rovers. Even though they have little influence on heavy machines, grousers are efficient for lightweight vehicles, such as planetary rovers. Moreover, Sutoh et al. (2012) has evaluated influence of grousers and concluded that increasing their number helps only to a certain extent.



Figure 2.6: Ball wheels implemented on MoVille car concept. (Source: Woo-Ram Lee (2013))

2.2.2 Suspension

Suspension is an important part of the locomotion system especially for lunar/-planetary rovers. It helps to absorb shocks stimulated by rough terrain and damp oscillations by buffering part of the energy transferred between surface and rover body (Deng et al., 2011). Suspension types found in existing rovers are:

- **Rocker-bogie** is the most well known suspension system as it has been adopted in many rovers, including Sojourner, MER, MSL. It has successfully demonstrated its high mobility performance in Martian missions. The suspension is able to passively keep all the wheels in contact with the surface on any kind of terrain (Seeni et al., 2008). Rocker-bogie suspension is explained in detail by Lindemann and Voorhees (2005) and can be seen in Figure 2.7. Another example, which uses Rocker-bogie suspension, is Lunar Exploration Rover (LER), developed by the Chinese Academy of Space Technology in 2006 (Zhang et al., 2008).
- **Articulated** suspension is common for cylinder-conical rovers, such as Marsokhod or Lama rover from (Peynot and Lacroix, 2003). In this configuration, wheels are mounted on the axles which can roll relatively to each other. The research on cylinder-conical locomotion control has been done by Yu et al. in 2006 and Hakenberg in 2008. This setup is shown in Figure 2.8.
- Other interesting suspension concepts include SR2, SOLERO, CRAB-II, CRAB-S, CRAB-8, RCL-E, RCL-C, Double Spring and are all described

by Seeni et al. (2008).

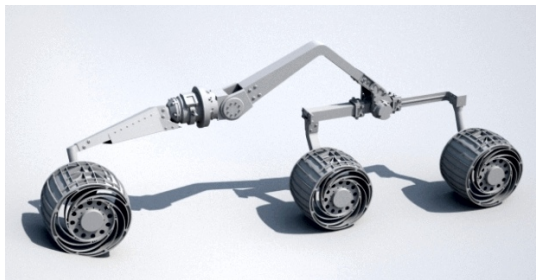


Figure 2.7: The Rocker-bogie suspension of MER. This is a modified version of the original Rocker-bogie suspension that can fold. Folding allowed to pack the rovers inside the spacecraft. (Source: ExploreMars.org (2012))



Figure 2.8: The articulated suspension of the Marsokhod rover. It is fully articulated in pitch, roll and yaw. (Source: Jean-Marc Maclou (2011))

2.2.3 Holonomicity

When designing a robot, its steering mode characterizes whether the system is holonomic or non-holonomic. Holonomicity is a relationship between controllable DOF and total DOF of the robot. If the number of controllable DOF is greater or equal to the total number of DOF, the robot is said to be holonomic. As in (Society of Robots, 2013), a non-holonomic robot cannot move in any direction but has to perform series of maneuvers to achieve the correct heading. The typical example of a non-holonomic system is a vehicle with Ackermann steering geometry. Since it can not move laterally, it has to follow the trajectory shown in Figure 2.9. Holonomic robot, on the other hand, can move in any direction. An example of a holonomic robot is URANUS robot that uses Mecanum wheels (See Figure 2.5). Holonomicity is important to remember when creating a path.

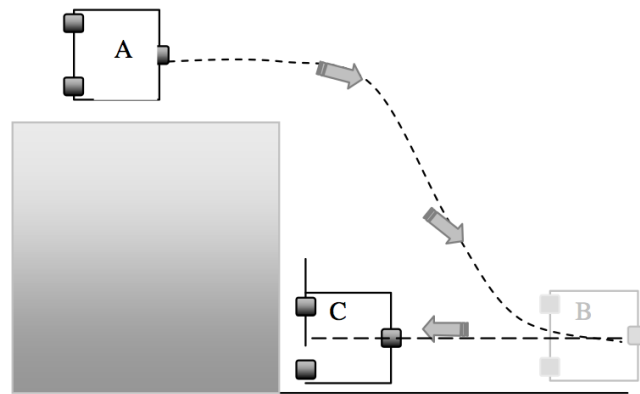


Figure 2.9: Forward-backward maneuver for a non-holonomic wheeled mobile robot. Path starts from point A and ends at point B. (Source: Khoukhi et al. (2007))

2.3 Steering

2.3.1 Skid Steering

The simplest and, thereby, very popular steering type is skid steering. As Fuke et al. (1995) have pointed out, skid steering does not require any specific steering mechanism. Instead, the rover has fixed wheels and the steering is done by varying velocity difference between individual wheels. As a result, the rover sweeps out the soil in a lateral direction. In turn, the lateral sweep causes resistance, which creates a high turning moment. According to Fuke et al., this turning moment grows when wheels start to sink in the regolith or collide with an object. Moreover, as stated by Shamah (1999), skid steering consumes

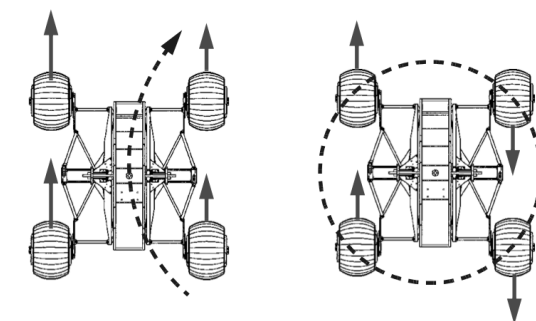


Figure 2.10: Skid steering Ackermann and Point-turn examples. (Source: Shamah (1999))

more power and demands higher torque than other steering configurations, for

instance, explicit steering. It happens due to higher sideslip angles produced by skid steering. Figure 2.10 shows the example of the skid steering. Another drawback of the skid steering is its poor accuracy. Since the turning motion of a skid steered rover highly depends on the grip with the surface, wheel odometry becomes unreliable. Instead, other means, such as Inertial Measurement Unit (IMU) or visual odometry, should be used.

2.3.2 Explicit Steering

Explicit steering is another popular type. It allows rotation of some wheels individually so that all wheels follow the arc around the same central point that is called Instantaneous Center of Rotation (ICR). As a result, sideslip is eliminated and driving efficiency is at its peak. In fact, explicit steering is the most used type, as most of the conventional automotive vehicles belong to it. However, they are able to steer only front wheels, while it is possible to

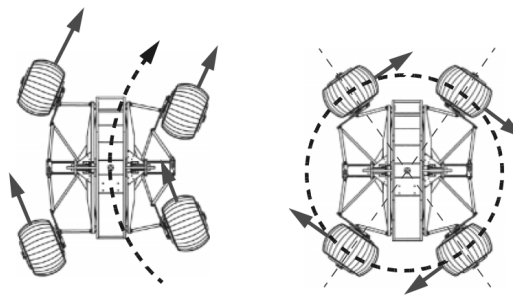


Figure 2.11: Explicit steering Ackermann and Point-turn examples. (Source: Shamah (1999))

steer all wheels in an all-wheel steering configuration. Most of the planetary rovers, including Martian rovers use an explicit steering configuration (National Aeronautics and Space Administration, 2012b). Figure 2.11 shows the explicit steering example.

2.3.3 Articulated Steering

In an articulated steering configuration, a four-wheel vehicle is split into two parts: the front and rear. The steering is done by changing the angle between front and rear halves. The advantage of this type is that it does not require

central differential because both halves follow the same trajectory. Still, changing the angle between halves requires much power. Therefore, hydraulics is a common mechanism for this steering, but, due to its bulkiness, it is not used in planetary rovers. This is the reason why articulated steering is usually omitted while designing a planetary rover. Figure 2.12 shows the articulated steering example.

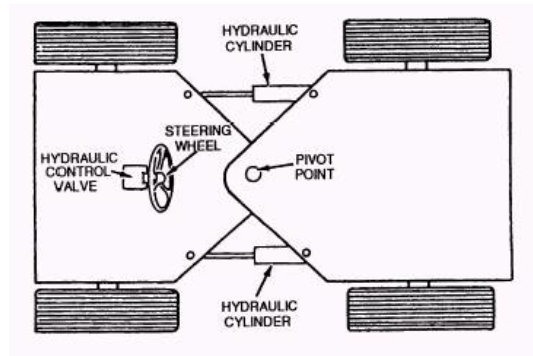


Figure 2.12: Articulated steering example. (Source: Integrated Publishing (2003))

The common steering modes of the planetary rovers are:

- Ackermann or *arc-turn*, as it is called by Lindemann and Voorhees (2005). In this mode rover wheels are steered so that the rover can follow a trajectory around ICR (Hoepflinger et al., 2008).
- Point-turn or *turn-in-place* as it is called Lindemann and Voorhees, is also called *Turn on Spot* in (Hoepflinger et al., 2008) and (Shamah, 1999). It uses straight paths, which connect multiple waypoints. After reaching a waypoint, the rover turns in place to adjust its heading toward next waypoint.
- Crab steering is a mode, when all wheels are pointing in the same direction. It enables a vehicle to move sideways. Figure 2.13 shows the example of the Crab steering.

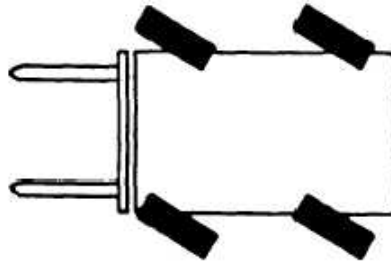


Figure 2.13: Crab control mode example. (Source: Headquarters, Department of the Army (1993))

2.4 Terrain

2.4.1 Regolith Properties

It has been explored that the lunar surface contains little steep craters and rocks and generally is flat (Zhang et al., 2008). Furthermore, lunar regolith is distributed uniformly. Therefore, properties of the surface in different places of the Moon have similar characteristics. Regolith particles are usually angular and sharp, as stated by Nishida et al. (2011), because weather phenomena that could affect those particles are not present on the Moon. Consequently, regolith particles act as abrasives and can damage unprotected systems. In addition, the lunar regolith is a loose substance, which creates some challenges in motion control.

Since it is not possible to use real lunar regolith in the Lunabotics Mining Competition, BP-1 regolith simulant is used instead. In the description by National Aeronautics and Space Administration (2012a), it is stated that BP-1 is a "crushed lava basalt aggregate with a natural particle size distribution similar to that of lunar soil". Moreover, BP-1 aggregate may contain naturally occurring rocks. The compressed simulant has the density of $1.5 - 1.8 \text{ g/cm}^3$, while 2 cm fluffy powder layer on the top of the surface is raked to the density of approximately 0.75 g/cm^3 . Moreover, it is emphasized that BP-1 behaves similar to actual lunar soil and differently from usual sand.

Performance evaluation metrics	Performance evaluation indexes
Obstacle surmounting capability: the capability of surmounting obstacle and gradeability while driving on the lunar surface without losing maneuverability	Slope angle Positive capability Negative capability
Trafficability performance: the capacity of solid to support a lunar rover and provide sufficient traction for locomotion system	Nominal ground pressure Sinkage ratio Drawbar pull ratio Resistance coefficient
Stability: capability to prevent overturning on the slope both downhill and cross-hill	Downhill angle Cross-hill angle Mass center height Ground clearance
Energy consumption: capability of minimum energy consumption working in diversified conditions	Energy efficiency Unit tractive force coefficient Wheel torque required Wheel power consumption

Table 2.1: Performance evaluation metrics and indexes. (Source: Zhang et al. (2008))

2.4.2 Wheel-Soil Interaction

Scientists are constantly trying to improve the existing configurations to increase their performance and efficiency. Many papers have been written on the wheel-soil interaction, including (Zhang et al., 2008), (Bauer et al., 2005), (Sutoh et al., 2012), (Deng et al., 2011), (Grecenko, 1992), (Andrade et al., 1998). According to Zhang et al., the performance of a mobility system is defined by "thrust available from the soil and resistance to motion". The best mobility platform is the one which requires the lowest friction coefficient to drive over an obstacle. Therefore, Deng et al. suggest selecting the diameter of the wheels with respect to the sinkage, tire width, load and considered slope angle. Usually, performance analysis considers wheel parameters, such as the diameter of the wheel or height of grousers; soil parameters; slip ratio and wheel loads. Performance evaluation metrics are listed in Table 2.1.

The most important metrics of the analysis are trafficability and terrainability, where trafficability defines the physical configuration along with environment and task parameters; terrainability is the ability of the rover to overcome rough

terrain features while keeping its stability and forward progress. Therefore, the position of the center of mass is important for good terrainability. For example, Zhang et al. have mentioned the ability to dynamically alter the mass center as a factor that could increase the capability of climbing steep slopes. One problem about the wheel-soil interaction is that it is hard to model. Subsequently, simulation of the rover behavior is usually generalized. Bauer et al. have listed possible modeling approaches, including:

- Modeling of every grain and its interaction with neighboring grains. This approach requires enormous amount of processing power.
- Modeling of cells, consisting of a finite number of grains, and their interaction with neighboring cells, as in (Andrade et al., 1998).

2.5 Driving Path

2.5.1 Path Generation

The Ackermann control mode requires smooth paths which can be created using different techniques. Choi et al. (2008) have mentioned circular arcs to construct turns as one of the options for trajectory generation. Another good alternative is Bézier curves, which are a popular tool used in many applications including object shape description, computer-aided design and path planning (Sohel et al., 2005). Liang et al. (2012) have mentioned cubic polynomial B-spline as a possibility but they state that the Bézier curve has smaller curvature and fast curve fitting rate. Usually, curvature is referred as a value inversely proportional to the turning radius, as shown in Equation 2.1, where k is curvature and R is the turning radius.

$$k = \frac{1}{R} \quad (2.1)$$

Hence, a path with higher curvature requires a smaller turning radius. Smaller path curvature is preferred because not any non-holonomic rover can follow a path with sharp curves. In other words, the highest possible angular velocity, which is achievable by the rover, should be greater than the changing range of

the path curvature. Otherwise, the rover will not be able to follow the trajectory, as described by Liang et al.. Additionally, some Bézier curves tend to have many control points. Choi et al. claim that it is undesirable to have a curve with many control points; instead, one should "join low-degree Bézier curves together in a smooth way for path planning". Hwang et al. (2003) have studied Piecewise Cubic Bézier Curves (PCBC) usage within an on-line smooth trajectory generation algorithm along with an on-line modification of those curves. This modification is crucial when the robot is put into dynamic environment, since obstacles can be discovered suddenly. It is based on the dodging behavior, described by Arkin in 1998, coupled with a trajectory generation algorithm. The trajectory is modified by recalculating the control points of the PCBC. In contrast, Montes et al. (2007) have studied Rational Bézier Curves and Clothoid curves for obstacle avoidance.

2.5.2 Path Selection

MSL and MER rovers are the good candidates to look at before trying to invent path generation workflow. MER rovers have proven the concept, which they are based on, by operating long beyond their estimated mission times. MER

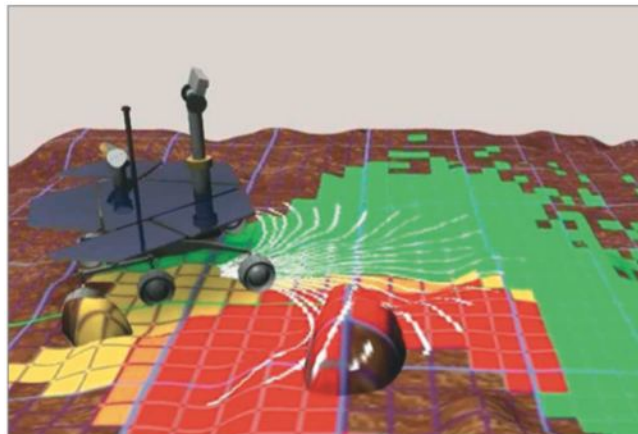


Figure 2.14: Example terrain assessment and path selection by MER vehicle. The terrain model is built up from stereo imagery. The colored grid represents the traversability map, and the white lines show the considered paths. (Source: Bajracharya et al. (2008))

and MSL rovers use the same navigation systems with only one exception that MSL can plan longer routes than MER rovers (National Aeronautics and Space

Administration, 2012b). The navigation system generates the surrounding terrain map from the stereo cameras. The map has grid representation, where each cell contains information about its traversability. While navigating, the rover generates several possible paths and assesses their path traversability and length. The shortest path, which has affordable traversability is selected. This process is shown in Figure 2.14.

2.5.3 Path Following

A common way to control the trajectory of a rover is called *bicycle model* (Ishigami and Yoshida, 2005). This model approximates a four-wheel vehicle base to a two-wheel base, when the front wheel is located in the middle of front wheel axle, and the rear wheel is located in the middle of the rear wheel axle. This approximation is useful when slip motion can be neglected. However, it is inaccurate on the loose soil. The example of the bicycle model is shown in Figure 2.15. In contrast, the all-wheel dynamics model, presented in the same figure, is a complex model consisting of two models: wheel contact model and dynamics model of the vehicle. The equation of motion for this model is as in Equation 2.2, where H is the inertia matrix of the vehicle, C is the velocity depending term, v_0 is the translational velocity, w_0 is the angular velocity, q is the angle of each joint of the vehicle, F_0 is the vector of forces acting on the center of gravity of the body, N_0 is the vector of torques acting on the center of gravity of the body, τ is the vector of torques acting on each joint of the vehicle, J is the Jacobian matrix, F_e is the vector of the external forces acting on the center of gravity of each wheel and N_e is the vector of torques acting on the center of gravity of each wheel.

$$H \begin{bmatrix} \dot{v}_0 \\ \dot{w}_0 \\ \ddot{q} \end{bmatrix} + C = \begin{bmatrix} F_0 \\ N_0 \\ \tau \end{bmatrix} + J^T \begin{bmatrix} F_e \\ N_e \end{bmatrix} \quad (2.2)$$

As stated by Lindemann and Voorhees (2005), loose soil on a cross slope contributes mostly to the side slip and has a little effect on the change of the yaw of the rover. MER rovers are known to sink down by 1-2 *cm* in soft soil, such as dry loose sand. However, sinkage would increase to 6-12 *cm* when traversing rocks taller than 15 *cm*. It happens because the rover needs more effort to

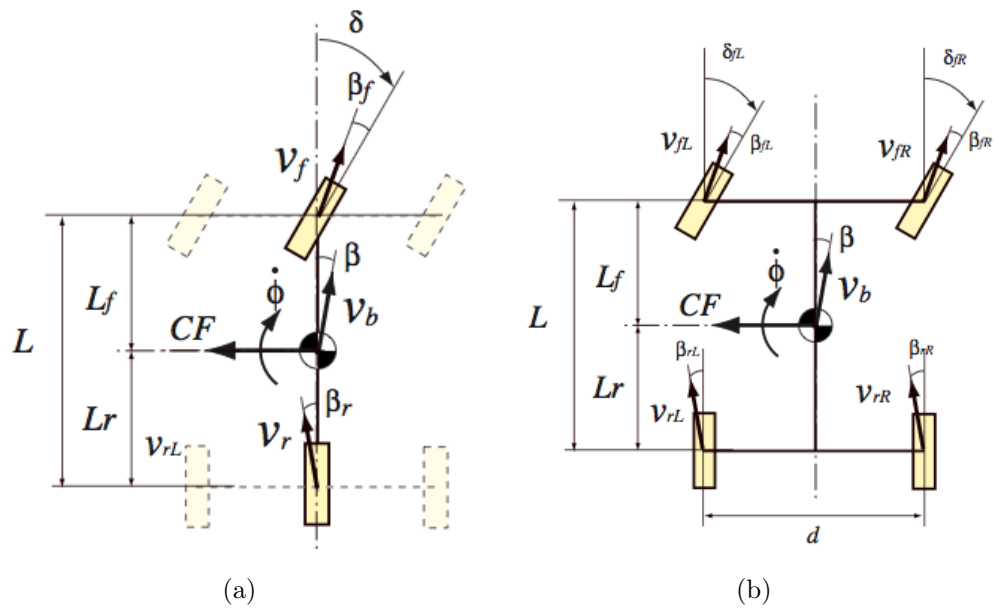


Figure 2.15: a) Bicycle model. b) All-wheel dynamics model. (Source: Ishigami and Yoshida (2005))

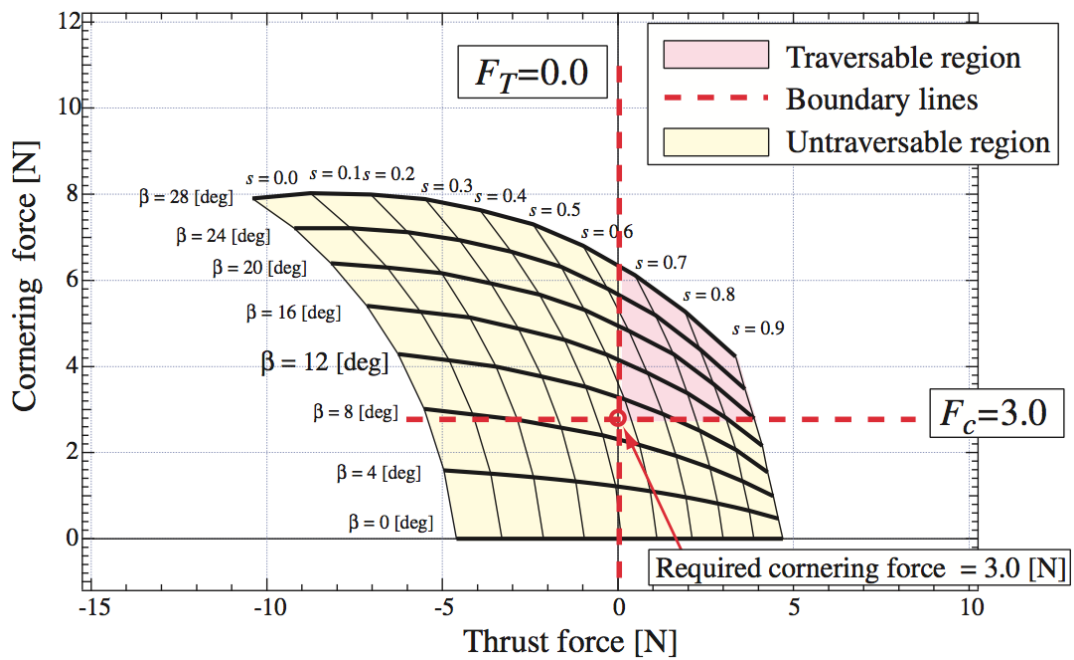


Figure 2.16: Thrust-Cornering Characteristic Diagram. (Source: Ishigami et al. (2008))

climb over the obstacle. Consequently, more skidding occurs, which results in a greater sinkage. Because sinkage and sideslip in the loose soil make it hard to predict actual motion, Bajracharya et al. (2008) have proposed ability to learn terrain properties on-the-fly as a solution. While driving, the rover would

augment its knowledge about the terrain and use it to correlate learned situations with current terrain geometry and appearance. Alternatively, Ishigami et al. (2006) have proposed a control algorithm with slip compensation, which calculates both steering and driving maneuvers. The simulation was carried out on a slope, uniformly covered with lunar regolith simulant, with an angle of 10° . Eventually, the experiment has shown the success of the proposed algorithm. Ishigami et al. define two ways of using their algorithm in a real system:

- Control the path following in real time.
- Simulate motion offline and generate maneuvers based on this simulation.

In addition, Ishigami et al. have suggested to use their thrust-cornering characteristic diagram for calculation of appropriate steering angles when traversing a slope. This diagram represents the relationship between the thrust force and cornering force, as shown in Figure 2.16.

Chapter 3

Implementation

This chapter reveals all the work that has been done to implement the motion control subsystem. First of all, Section 3.1 gives an overview of the mechanical structure of the rover and the electronics setup. Section 3.2 describes the existing search algorithms and the final implementation based on the A* search. The techniques used while developing a path following control system, are presented in Section 3.3, followed by Section 3.4, which makes the detailed overview of the developed software packages for both the onboard computer of the rover and the control computer. Finally, Section 3.5 explains the simulation that has been done and the models that have been used.

3.1 Rover Hardware

The mechanical design, described below has been selected in order to meet the competition rules and fit in a budget at the same time. The total weight of the vehicle is estimated to be approximately 78 kg , whereas a rover is allowed to be up to 80 kg without being penalized, as explained in Section 1.1.

3.1.1 Chassis

Among all the possible locomotion configurations described in Section 2.1, wheeled locomotion was selected, as it is durable and easy to use. Although

six-wheeled platforms are the most common in planetary missions, four-wheeled type was considered sufficient for this competition. In addition, a higher number of wheels would require more effort during the development phase, while the time was extremely limited for this project. Selected four-wheeled platform configuration has a $4 \times 4 \times 4$ wheel formula (Wheel formula is explained in Section 2.2). It gives the highest possible flexibility and allows to use Crab steering along with other steering modes.



Figure 3.1: Three of six 20 inch aluminum wheels on an engineering model of the MSL rover. Steering axes are crossing the centers of the wheels. (Source: Tann (2012))

Usually, the mechanical design of the wheels is done in a way that the wheel is aligned with the rotation axis of the steering motor, as shown in Figure 3.1. Thus, the steering action does not change the displacement of a wheel with respect to the center of a rover. However, this design was not affordable due to the following reasons:

- It increases the height of the chassis, while dimensions are very limited for our project.
- It requires driving motors to be placed inside the wheels. Such motors are costly and are not affordable in the framework of our project.

As a result, the wheels are displaced from the steering axes by the distance needed to accommodate driving motor and gearbox assemblies, namely 13 cm .

Other dimensions are as follows:

- The steering pivot points are placed 53 cm in the longitudinal direction and 20.5 cm in the lateral direction away from the geometrical center.
- Wheel axes are 4.66 cm below the geometrical center.
- Wheels have 10 cm width and 29.9 cm diameter.

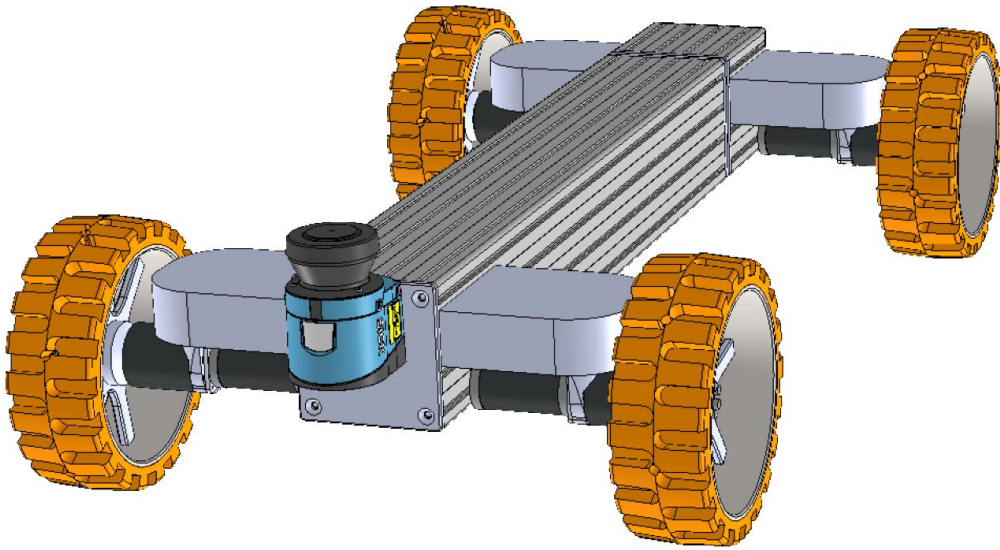


Figure 3.2: The CAD model of the chassis of the rover.

The suspension system implies the rear axle that can passively pivot up to $\pm 30^\circ$ around the longitudinal axis of the rover. Due to the limitation of the steering motors, they can rotate 90° outwards and 65° inwards. It creates dead zones for the ICR-based steering that is described in Section 3.3.3. These dead zones are referred to in Figure 3.4. The Computer-Aided Design (CAD) model of the chassis is shown in Figure 3.2. The setup of a wheel module is shown in Figure 3.3. As seen from the figure, the wheel is positioned on a certain distance from the steering axis. Hence, when the steering motor is rotating, it requires a certain rotation of the corresponding wheel in order to avoid skidding. Also, as the wheel can already be rotating when the robot is moving, the rotation rate required by the steering is considered a compensation to the current driving velocity. Adding the compensation velocity to the current driving velocity helps to avoid skidding, even when the robot is moving. The relation of the linear

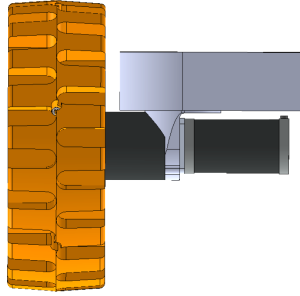


Figure 3.3: The CAD model of the wheel module. The gray box on the top is the steering mechanism containing a steering motor and a worm drive transmission, the black cylinder on the right is the driving motor and black cylinder in the center is the gearbox.

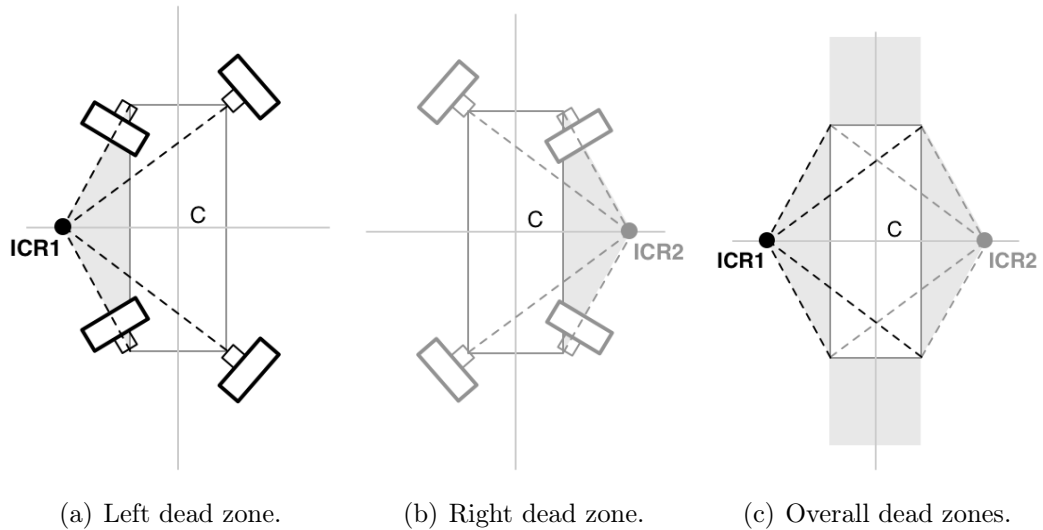


Figure 3.4: Mechanical constraints of the steering motors. Light gray areas are the dead zones where ICR point cannot be placed. Left (a) and right (b) triangles are created by mechanical constraints. Upper and lower rectangles are restricted by the algorithm because the ICR point is always placed on the side of the rover for the curve following or inside the rover for Point-turn rotation. Black lines represent the wheel states when driving around ICR1, while gray lines represent the wheel states for the driving around ICR2.

and angular velocities is shown in Equation 3.1, where v is the linear velocity, w is the angular velocity and R is the radius of a turn. Therefore, the tangential velocity of a wheel can be calculated from Equation 3.2 and then converted to the angular velocity, as shown in Equation 3.3. w_d is the angular velocity of the driving motor; w_s is the angular velocity of the steering motor; v_w is the

tangential velocity of the wheel; l is the shoulder between the steering motor and the wheel; r is the wheel radius.

$$w = \frac{v}{R} \quad (3.1)$$

$$v_w = lw_s \quad (3.2)$$

$$w_d = \frac{v_w}{r} = \frac{lw_s}{r} \quad (3.3)$$

3.1.2 Electronics

The electronics node consists of two FitPC3 computers, running Ubuntu 12.04. The computers are interfaced via the Ethernet cable and act as a single processing unit. They are also connected to the wireless router to be able to receive remote commands from the controlling computer. The chassis has its own electronics board which receives high level commands from the FitPC3 computers and dispatches them among low level components. For example, the computers may request to steer a particular wheel by providing a steering angle, whereas the chassis will select and perform motor routines to reach the desired angle. All the subsystems are powered by the battery pack, which can output voltages of 5 and 24 V. The sensor collection includes two infrared proximity sensors for precision driving while excavating regolith; four load cells to measure the weight of the carried regolith; IMU, Red-Green-Blue-Depth (RGB-D) Camera, wheel encoders and a lidar for Simultaneous Localization and Mapping (SLAM) and odometry; bumpers for safety.

3.2 Path Planning Algorithms

Path planning is meant to find a way from the current position of a robot to the goal. This problem requires knowledge about the environment and the current state of the robot. A common way to represent the environment is an occupancy grid map. It is a two-dimensional matrix, where each cell contains occupancy information. This information can be treated as traversability option or a surface type. For example, Oniga et al. (2009) have three distinct types of the occupancy grid cells: a road, traffic isles and obstacles. In our project,

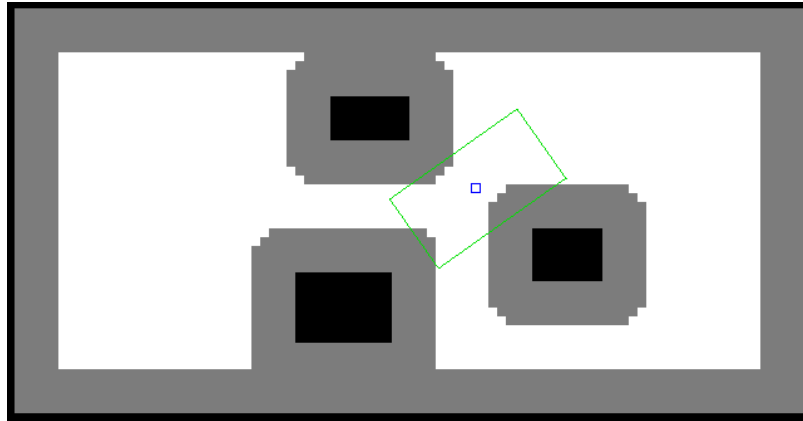


Figure 3.5: The example of the obstacles inflated in the occupancy grid of the LunArena map. The black cells represent the areas occupied by the real obstacles, the gray cells represent the inflated obstacles, the blue square is the cell occupied by the rover and the green rectangular frame around it is the real dimensions of the rover.

three values are used for the sake of simplicity: obstacles, safety zone and unoccupied area. Obstacles are inflated in our maps. It means that the occupied size in the map does not correspond to the real size of an obstacle and is much bigger. It is inflated in all directions by the distance that is safe for the rover. As the rover can occupy only one cell in the map, this occupied cell corresponds to the geometrical center of the rover. If the position of the rover in the map was adjacent with an obstacle cell, without inflation, it would collide with an obstacle unless the dimensions of the rover are smaller than the cell resolution. Figure 3.5 represents the described configuration. Of course, techniques described in Sections 3.2.2 and 3.2.3 help to avoid inflated obstacle models. However, it was decided to leave it as a convenient way to constrain path smoothing algorithm described in Section 3.2.5.

3.2.1 Search Algorithms Overview

Search algorithms are used to find a solution in a tree or graph (closed tree) of the state space. The occupancy grid mentioned previously is also a representation of a graph, when each grid cell corresponds to a node in the graph. Therefore, a search algorithm is the core component of the path planning problem. Grid-based search algorithms can be 4-directional and 8-directional, as

shown in Figure 3.6. Also, Russell and Norvig (2003a) split search algorithms into two categories, namely: informed and uninformed search algorithms.

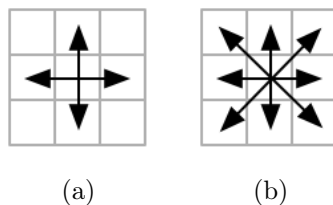


Figure 3.6: a) Expansion of a node in the 4-directional search. b) Expansion of a node in the 8-directional search.

Uninformed Search Algorithms

Uninformed search is also called a blind search. It does not contain any additional information besides the information provided by the problem definition. These algorithms cannot distinguish which one of several states is more efficient or desirable. Common uninformed search algorithms explained by Russell and Norvig include Breadth-first search, Depth-first search, Uniform-cost search, Depth-limited search, Iterative deeping depth-first search and Bidirectional search.

Informed Search Algorithms

In contrast to an uninformed search, informed search uses problem-specific knowledge together with problem definition to find a solution. The general approach of the informed search is called *best-first* search. A key component of the best-first search algorithms is a heuristic function, which denotes "the estimated cost of the cheapest path from node n to a goal node", as explained in (Russell and Norvig, 2003b). Informed search algorithm family includes the following algorithms:

- **Greedy best-first search** tries to expand the node which is likely the closest to the goal. In this sense, it is similar to the depth-first search since it tries to follow the same branch to the goal. If the node does not

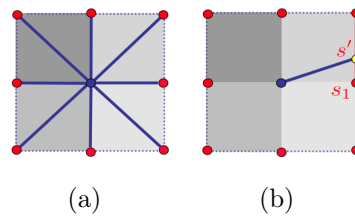


Figure 3.7: a) The typical transitions (in blue) allowed from a node (shown at the center) in a uniform grid. Notice that only headings of 45° increments are available. b) Using linear interpolation, the path cost of any point s' on an edge between two grid nodes s_1 and s_2 can be approximated. This can be used to plan paths through grids that are not restricted to just 45° heading transitions. (Source: Carsten et al. (2007))

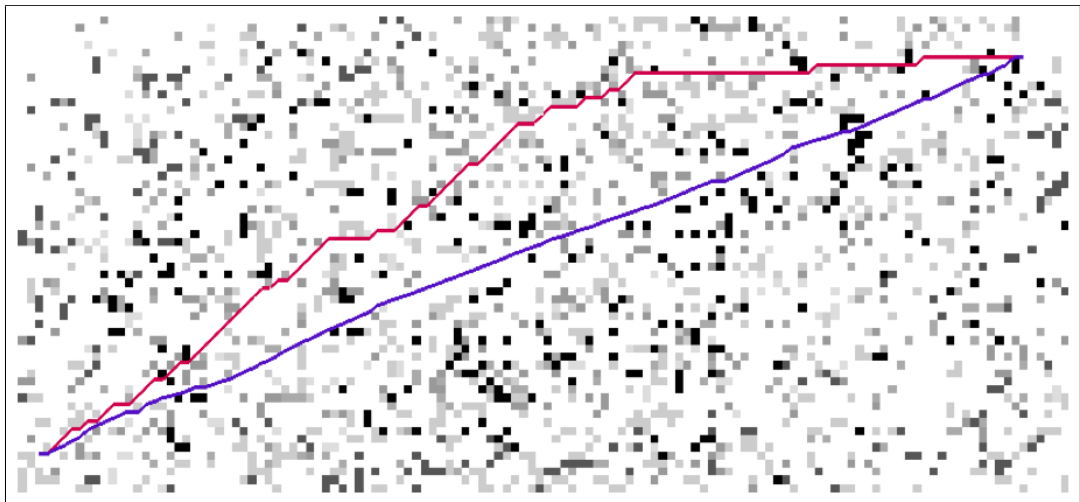


Figure 3.8: Paths produced by classic grid-based planners (red/top) and Field D* (blue/bottom) in a 150×60 uniform resolution grid. Darker cells represent higher-cost areas. (Source: Carsten et al. (2007))

have any successors, and the goal was not reached then greedy best-first search will back-up and switch to another branch.

- **A* search** is a very well known in the area of computer science and robotics. As described by Russell and Norvig (2003b), it assesses the nodes by introducing a complex heuristic function, which consists of a sum of the cost to reach the node $g(n)$ and the cost to reach the goal from this node $h(n)$. Hence, the heuristic function of the A* search algorithm is defined by the Equation 3.4. A* search is proven to be optimal if $h(n)$ is an admissible heuristic, that is, it never overestimates the cost to reach

the goal.

- **D* search** is named after Dynamic A* and is an extension of A* search algorithm. Its main advantage is that it can efficiently work within dynamically changing environment, whereas A* is a static search. It is very useful when used in field robotics because environment is continuously changing and is complemented by the newly observed obstacles. There exist several versions of D* algorithm and D* Lite is the most popular among them. D* Lite is based on Lifelong Planning A* (LPA*), which is an iterative version of A*, as stated by Koenig and Likhachev (2002).
- **Field D* search** is a modification of D* search algorithm used in field robotics. It is implemented onboard MER and MSL rovers as a part of AutoNav path planner subsystem. As Carsten et al. (2007) have explained, in contrast to the grid-based planning algorithms like A*, Field D* allows transition through any point of an neighboring cell edge. Grid-based algorithms allow transitions only through the cell centers and corners (in case of a 8-directional search). This difference can be seen in Figure 3.7. The difference of the resulting paths can be seen in Figure 3.8.
- **Rover Adapted D* (raD*) search** was presented by Würgler and Sukkarieh (2010). Instead of considering just adjacent cells, this algorithm can expand the search space of each node depending on the order of the algorithm. Authors presented versions with orders of 8, 16 and 32. The example of raD* 32 is shown in Figure 3.9.

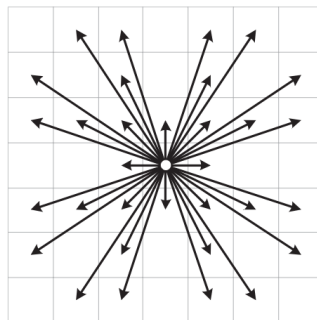
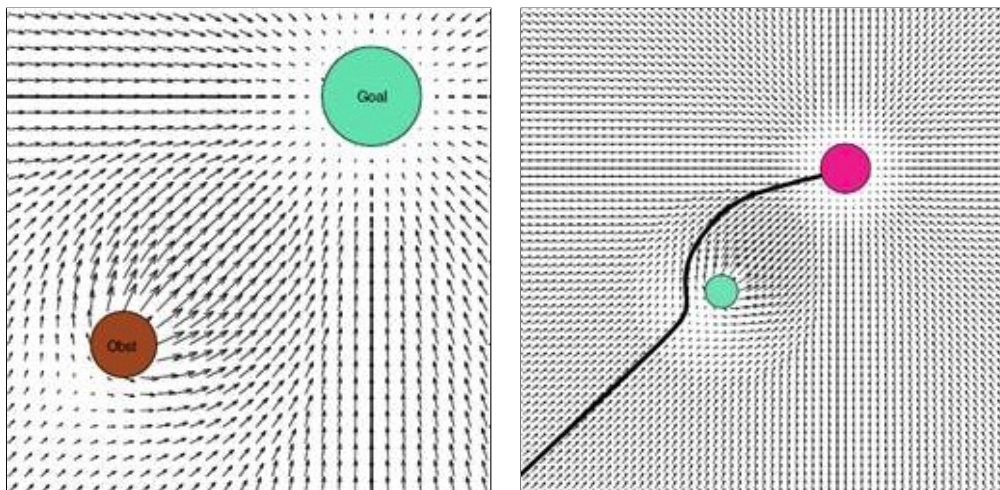


Figure 3.9: Expansion of a node in raD* 32 search. (Source: Würgler and Sukkarieh (2010))

$$f(n) = g(n) + h(n) \quad (3.4)$$

3.2.2 Potential Fields

Motion planning based on potential fields uses the principle of charged particles. In nature, particles with opposite charges attract each other, whereas particles of the same charge repel. Likewise, the robot is repelled from the obstacles and attracted to its goal. By combining attractive and repulsive vector fields, it is possible to model the motion of the robot. The motion is derived by calculating the local force vector at each point, as shown in Figure 3.10. The main advantage of this approach is computational cheapness. However, potential fields usually suffer from local minima problems. Local minima are the spots where the sum of attractive and repulsive forces is zero. It happens under certain circumstances, for example, when a robot enters a U-shaped obstacle. When trapped, it cannot move anywhere. Thus, alternative approaches have to be used for these particular cases.



(a) Attractive and repulsive fields.

(b) Resulting trajectory.

Figure 3.10: The example of a potential field. Small arrows are the vectors of the local forces derived from the combination of attractive and repulsive fields. (Source: Safadi (2007))

3.2.3 Sampling-Based Algorithms

Sampling-based algorithms are often used with the configuration space. The configuration space or *C-space* is a set of configurations, where each configura-

tion consists of variables that fully define the state of a robot. "Probabilistic planners build an approximate model of the valid C-space (C_{free}) by selecting random samples of configurations and transitions between them according to some strategy" (Morales et al., 2007). Usually, the output of the planner is a graph or tree called *roadmap* that contains vertices representing possible configurations and edges representing feasible transitions between these configurations. C-space planning is very popular with the high-dimensional models, such as the motion of a robotic manipulator.

3.2.4 Path Planning Solution

The resulting algorithm consists of the grid-based search, potential field approach and sampling-based strategy. For the grid-search, A* search algorithm was used, as it is sufficient for the application. D* was considered an optional objective, as more effort was directed towards the path following research rather than finding a path. 8-direction A* was used in a grid, where the transitions have costs equal to the Euclidian distances, that is, 1 for transition between adjacent cells and $\sqrt{2}$ for the diagonal transition. Later, when all-wheel rover control was undergoing the path following testing, it became clear that reduction of path nodes has to be done. As a result, the A* algorithm was enhanced to exclude intermediate nodes that can be safely removed. An intermediate node can be safely removed if:

- It has not been explicitly selected in the multiple waypoints selection mode.
- The line connecting its parent and child nodes does not intersect any obstacles.

This condition is checked every time when adding a new node to the final graph. The comparison of the outputs of original and modified A* algorithms can be seen in Figure 3.11. The modified A* algorithm produces a path similar to the Field D* result.

Later, the grid-based search was updated to reflect the properties of the potential field approach. Thus, the cells that are close to an obstacle get the

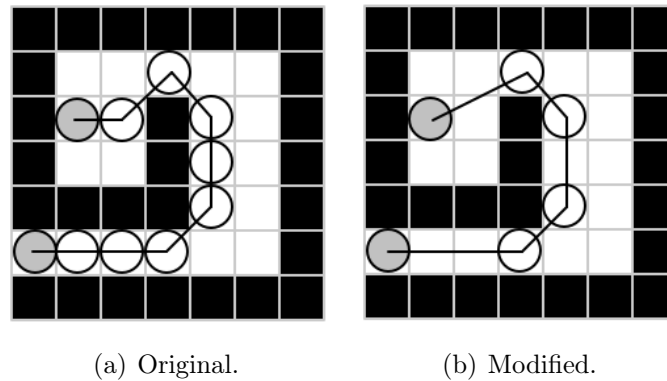


Figure 3.11: The comparison of the outputs of original and modified A* algorithms. Black cells are obstacles, white circles connected by solid lines are waypoints and the gray circles are initial and the goal cell.

penalty which is added to their heuristic. Consequently, the algorithm tries to stay away from the obstacles to a certain extent, just like repulsive force was acting on it. This repulsive penalty is inversely proportional to the distance between the node and the closest obstacle. In addition, it is applied only at a distance not larger than half width of the robot. Hence, the robot will not go too far away from the obstacle in a wide open area. The attractive force was not added, because the A* heuristic already contains information about the distance to the goal.

Finally, the C-space sampling was added to provide an even safer path. The configuration space of the rover consists of the position and orientation of the rover. The position of the wheels is not considered, as the rover is switching several states when rotating in place. Therefore, the maximum dimensions of the rover are used for collision detection (the maximum length is achieved when the wheels are aligned longitudinally, whereas the largest width is when the wheels are aligned laterally). When the grid-based search is adding a new node, it checks whether a collision can occur when:

- The rover will try to rotate towards a new waypoint.
- The rover will arrive at the new waypoint with this orientation.

3.2.5 Path Smoothing

The aim of path smoothing is to ensure the continuous trajectory of a rover by removing sharp turns. In addition, good path smoothing algorithms should seek minimization of path curvature, as it allows a rover to keep a higher linear velocity. Depending on the steering configuration, not any vehicle can perform turns of high curvature. For example, automotive vehicles that use conventional Ackermann steering geometry can only perform turns with a radius of at least few meters. Consequently, they can not follow a trajectory with very sharp turns.

Path smoothing algorithms are often based on Bézier curves. As described in (About.com, 2012), "a Bézier curve is a curved line or path defined by mathematical equations". A French mathematician and engineer Pierre Bézier developed this method when he was working for Renault company in the late 1960s. At first, it was intended for computer drawing software but nowadays has got a broader use. The shape of the Bézier curve can vary dramatically depending on the position of its control points and the order of the equation which describes it. Common types are:

- **First order** - linear Bézier curve, which contains two control points (See Figure 3.12).
- **Second order** - quadratic Bézier curve, which contains three control points (See Figure 3.13).
- **Third order** - cubic Bézier curve, which contains four control points (See Figure 3.14).

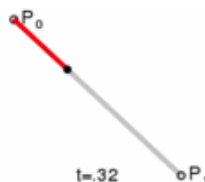


Figure 3.12: Construction of the linear Bézier curve. P_0 and P_1 are its control points. t is the parameter for constructing the curve, given that the curve is defined by $B(t)$, $t \in [0, 1]$.

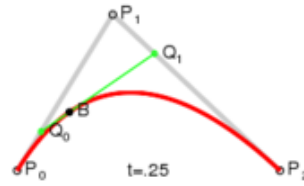


Figure 3.13: Construction of the quadratic Bézier curve. P_0 , P_1 and P_2 are its control points. t is the parameter for constructing the curve, given that the curve is defined by $B(t)$, $t \in [0, 1]$.

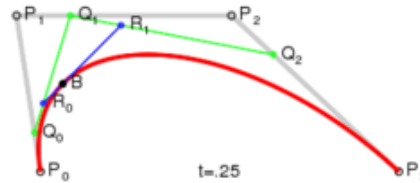


Figure 3.14: Construction of the cubic Bézier curve. P_0 , P_1 , P_2 and P_3 are its control points. t is the parameter for constructing the curve, given that the curve is defined by $B(t)$, $t \in [0, 1]$.

Of course, Bézier curves can be of higher orders and describe much more sophisticated paths. However, their use is discouraged in path planning applications because they are computationally much more expensive. Instead, multiple simple curves can be linked together to form a complete path.

A good path smoothing approach that seeks minimization of the curvature of a path is described in by Choi et al. (2012). It makes use of quadratic Bézier curves to smoothen the corners. In the notation of the paper, quadratic Bézier curve $Q(\lambda)$ is described by three control points q_0 , q_1 and q_2 , such that any point of the curve can be calculated from the Equation 3.5.

$$Q(\lambda) = (1 - \lambda)^2 q_0 + 2\lambda(1 - \lambda)q_1 + \lambda^2 q_2, \quad \lambda \in [0, 1] \quad (3.5)$$

The middle control point q_1 of the quadratic Bézier curve always remains at the corner, whereas the terminal control points \tilde{q}_0 and \tilde{q}_2 are shifted along the path lines, yielding the new control points q_0 and q_2 . If no obstacles are around, the terminal control points also remain at their initial places, that is $q_0 = \tilde{q}_0$ and $q_2 = \tilde{q}_2$, producing a very smooth curve. On the other hand, when such a curve crosses an obstacle, it should be recalculated to fit into the tetragonal concave polygon, defined by four points: three initial control points \tilde{q}_0 , q_1 , \tilde{q}_2 and the obstacle point p , which is the closest obstacle point with respect to the middle control point q_1 . Such a polygon is illustrated in Figure 3.15. To solve

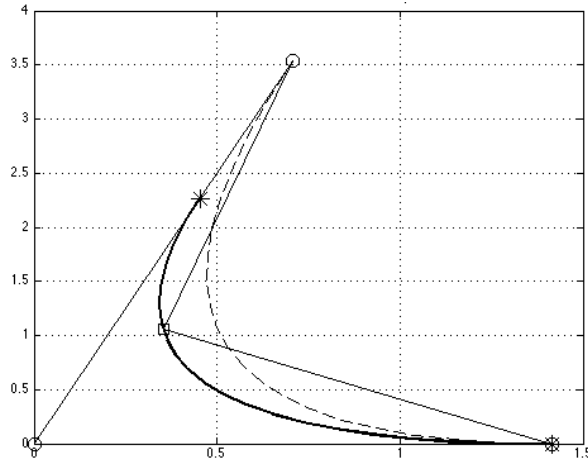


Figure 3.15: A quadratic Bézier curve created by the proposed algorithm. Solid line boundaries denote the tetragonal concave polygon, square is the obstacle point p , circles are the initial control points \tilde{q}_0 , q_1 and \tilde{q}_2 , dashed line is the initial quadratic Bézier curve, bold line is the recalculated quadratic Bézier curve, stars are the recalculated terminal control points q_0 and q_2 .

this problem, Choi et al. introduce two constraints:

- "Given a triangle $\tilde{q}_0q_1\tilde{q}_2$, find the points q_0 and q_2 on the segments $q_1\tilde{q}_0$ and $q_1\tilde{q}_2$, respectively, that minimize the maximum curvature of the quadratic Bézier curve with control points q_0 , q_1 , and q_2 ".
- "The same problem with the additional constraint that the resulting Bézier curve lies anywhere within the quadrilateral $\tilde{q}_0q_1\tilde{q}_2p$, where p is a given point that lies in the triangle $\tilde{q}_0q_1\tilde{q}_2$ ".

In fact, this method alters the distances between the control points rather than the control point positions. These distances are denoted as α and β and can be seen in Figure 3.16. $\tilde{\alpha}$ and $\tilde{\beta}$ are the distances $q_1\tilde{q}_0$ and $q_1\tilde{q}_2$ respectively. Therefore, $0 < \alpha \leq \tilde{\alpha}$ and $0 < \beta \leq \tilde{\beta}$. The positions of the control points q_0 and q_2 can be calculated later from α and β , considering that the middle control point q_1 remains fixed. Omitting all the theorems and their proofs, described by Choi et al., the optimum values α^* and β^* are calculated from the Equations 3.6

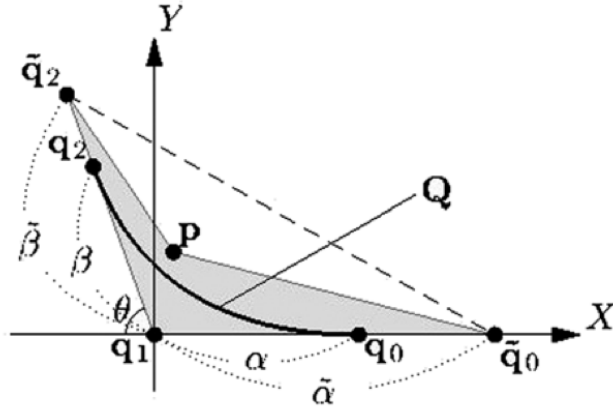


Figure 3.16: Given q_1 , \tilde{q}_0 and \tilde{q}_2 that bounds q_0 and q_2 on $q_1\tilde{q}_0$ and $q_1\tilde{q}_2$, a quadratic Bézier curve Q illustrated by bold solid line is determined by θ , α and β . θ is the heading difference between $\tilde{q}_2 - q_1$ and $q_1 - \tilde{q}_0$. α and β denote $\|q_0 - q_1\|$ and $\|q_2 - q_1\|$, respectively. p defines the tetragonal concave $\tilde{q}_0q_1\tilde{q}_2p$ within which Q must lie. (Source: Choi et al. (2012))

and 3.7 respectively, where separate terms can be found in Equations 3.8-3.14.

$$\alpha^* = \arg \min \frac{\left((\sqrt{\alpha} - \sqrt{K_\alpha})^4 - 2K_\beta \cos \theta (\sqrt{\alpha} - \sqrt{K_\alpha})^2 + K_\beta^2 \right)^{\frac{3}{2}}}{2K_\beta^2 \sin^2 \theta (\sqrt{\alpha} - \sqrt{K_\alpha})^2 \alpha}, \quad (3.6)$$

$$\forall \alpha \in \left[\max(\alpha_c, \alpha_p(\tilde{\beta})), \min(\alpha_m, \tilde{\alpha}) \right]$$

$$\beta^* = \beta_p(\alpha^*) \quad (3.7)$$

$$K_\alpha = p_x + p_y \cot \theta \quad (3.8)$$

$$\alpha_c = \left(\sqrt{K_\alpha} + \sqrt{\frac{K_\beta}{\Theta}} \right)^2 \quad (3.9)$$

$$\alpha_p(\beta) = \frac{K_\alpha}{\left(1 - \sqrt{K_\beta/\beta} \right)^2} \quad (3.10)$$

$$\Theta = \frac{-\cos \theta + \sqrt{\cos^2 \theta + 8}}{2} \quad (3.11)$$

$$K_\beta = \frac{p_y}{\sin \theta} \quad (3.12)$$

$$\alpha_m = \left(\sqrt{K_\alpha} + \sqrt{\frac{K_\beta}{|\cos \theta|}} \right)^2 \quad (3.13)$$

$$\beta_p(\alpha) = \frac{K_\beta}{\left(1 - \sqrt{K_\alpha/\alpha} \right)^2} \quad (3.14)$$

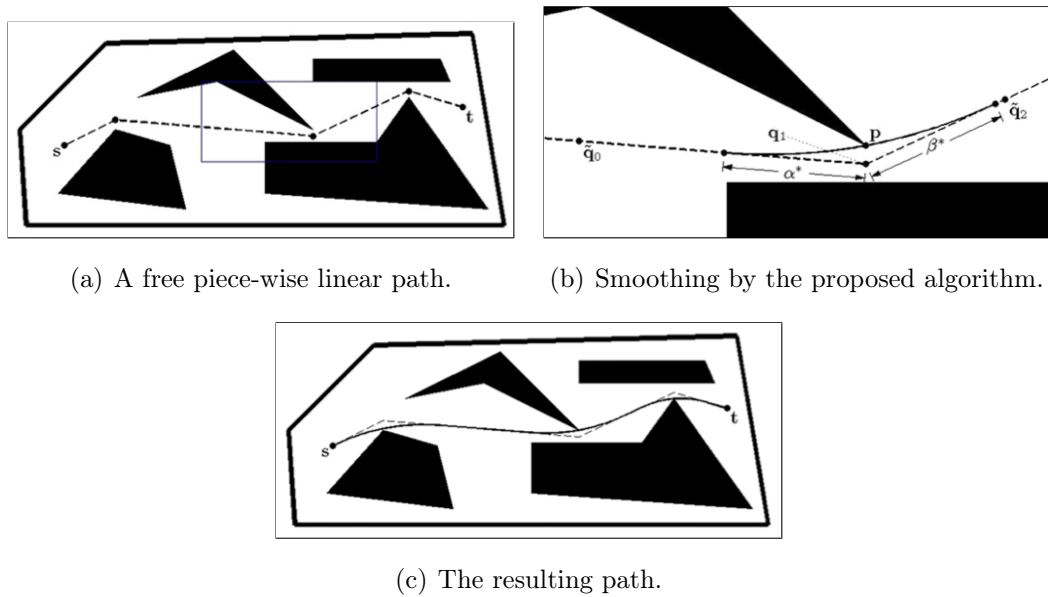


Figure 3.17: Path smoothing analysis conducted by Choi et al..

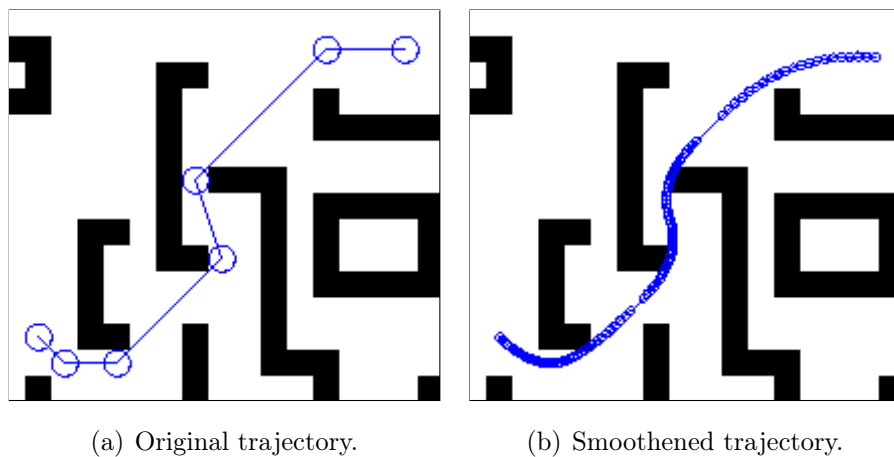


Figure 3.18: The smooth trajectory constructed by the tetragonal concave boundary method. Black cells are obstacles, white cells represent free space, blue circles connected by the straight lines are the waypoints of the trajectory. Circles are so densely distributed in (b) that they look like a thick solid line. Each Bézier curve is cut into 20 segments.

The Bézier curve created by the proposed algorithm is calculated for a single corner of the trajectory. To construct the entire path, Choi et al. suggest iterating over all the corners of the trajectory. The terminal control points \tilde{q}_0 and \tilde{q}_2 are set on the midpoints of the straight paths between the adjacent corners. The middle control point q_1 is set at the corner point. An exception should be made for the first and last straight paths. In this case, the terminal

control point \tilde{q}_0 is set at the beginning of the starting path, whereas the terminal control point \tilde{q}_2 is set at the end of the final straight path. The results of the research, conducted by Choi et al. can be seen in Figure 3.17. The trajectory constructed within our application is shown in Figure 3.18.

3.3 Path Following Algorithms

Path following is a problem that has to be solved in order for a rover to successfully reach its goal by following the predefined trajectory. When designing a path following mechanism, one seeks to minimize the deviation of the actual trajectory from the planned trajectory. In other words, the rover has to stick to the given trajectory as close as possible. It is done by implementing control algorithms described in Section 3.3.1.

3.3.1 Control Overview

In the control theory there are two major types of controls: *open-loop* and *closed-loop* or *feedback control*. In the opposition to the feedback there is a type called *feedforward control*, which is not very common. Finally, there is a term called *fuzzy logic* that can switch between control strategies based on certain parameters. However, fuzzy logic is not considered in this thesis. As explained in the book by Ogata (2010), closed-loop system compares the reference input with the feedback to produce a control signal. The feedback can be the output signal itself or a function of the output signal and its derivatives. In contrast, open-loop control system is not affected by the output signal at all. Open-loop control systems are easier to build and are advisable in the applications, where all the inputs are known in the beginning. Also, open-loop control systems do not have stability issues, which, in fact, are the major problem of the feedback control systems. A system can get unstable when it tends to overcorrect the error, thus, producing oscillations of constant or variable amplitude. Feedforward control is similar to the open-loop control but is not exactly the same. It requires the knowledge of a mathematical model of the plant. Feedforward control system makes a prediction from the disturbance based on the mathematical model and adds it to the input signal. Three of the mentioned control types

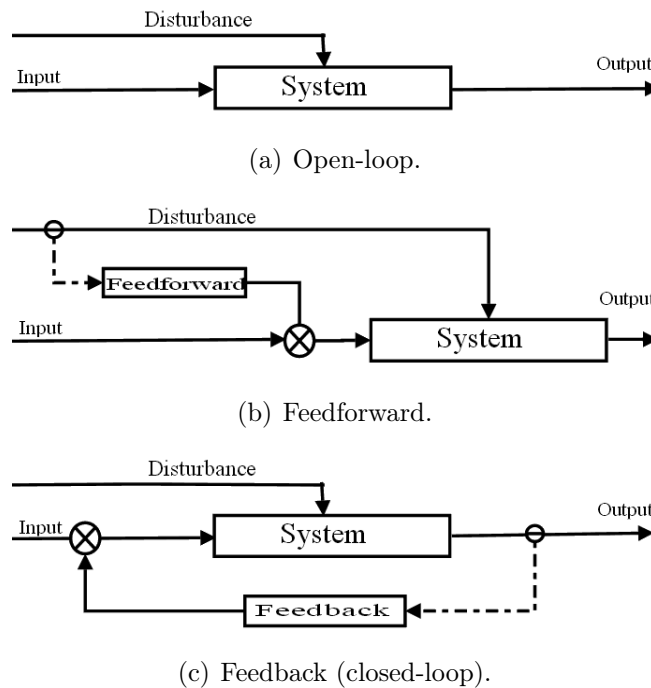


Figure 3.19: The block diagrams of different control systems.

are presented in Figure 3.19. The most popular type of the feedback control system is a Proportional-Integral-Derivative (PID) controller. Its definition can be found in Section 3.3.2.

3.3.2 PID Controller

A PID controller is a common approach for the problems that require minimization of an error. It is a control loop feedback mechanism from the control theory. As stated by Mastascusa (2013), PID controllers are often used because they give a designer more freedom and possibilities to change the dynamics of a system. In particular, one can start by using just the proportional controller and then add the integral or derivative parts later. A PID controller block diagram is shown in Figure 3.20. It takes the system error as an input. The output signal of the controller is a sum of P-term (output of the proportional controller), D-term (output of the derivative controller) and I-term (output of the integral controller).

The proportional controller calculates a term proportional to the error by multiplying it with the proportional gain K_p , as shown in Equation 3.15. Therefore,

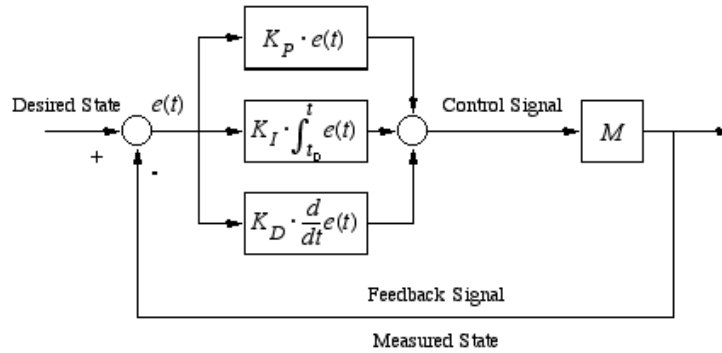


Figure 3.20: The block diagram of a PID controller.

the output of the proportional controller is affected by the present.

$$P_{out} = K_p e(t) \quad (3.15)$$

Very high proportional gain can make the system unstable while very low gain causes the slow response of the system. Also, the proportional controller alone can result in an undesired steady-state error.

The integral controller improves the steady-state error but significantly slows the system down. It outputs the accumulated error from the previous iterations multiplied by the integral gain K_i , as shown in Equation 3.16. Thus, the integral controller is affected by the past, namely the error accumulated in the past.

$$I_{out} = K_i \int_{t_0}^{t_1} e(t) dt \quad (3.16)$$

Usually, it is enough to calculate the accumulated error over a certain period of time rather than the entire lifetime of the system.

The derivative controller predicts the future by measuring the slope of the error. It helps to improve both the response and stability of the system. The output of the derivative controller is calculated by multiplying the derivative gain K_d with the derivative of the error, as shown in Equation 3.17.

$$D_{out} = K_d \frac{d}{dt} e(t) \quad (3.17)$$

As a result, all three terms are preferred because when coupled, they compensate drawbacks of each other. The control signal is then calculated using Equation 3.18. While creating a PID controller, a designer usually alters the

gain values using a trial-and-error method. Setting one of the gains to zero allows to exclude the appropriate controller from the control loop. For example, setting $K_i = 0$ yields a PD controller instead of a PID.

$$\text{Control signal} = K_p e(t) + K_i \int e(t) dt + K_d \frac{de(t)}{dt} \quad (3.18)$$

3.3.3 Ackermann Path Following

The Ackermann path following algorithm combines feedback and feedforward control techniques. The feedback chain is based on a PID controller, which is explained in Section 3.3.2. The feedback error of the controller is a distance between feedback look-ahead point and the closest point of the trajectory. This look-ahead point is put in front of the rover on a distance that is proportional to the longitudinal velocity and has a certain minimal value. This configuration is shown in Figure 3.21. The output of the controller is the turning angle, which is marked as γ in Figure 3.24. The tangential velocity of the geometrical center of the rover is inversely proportional to the angular to this angle.

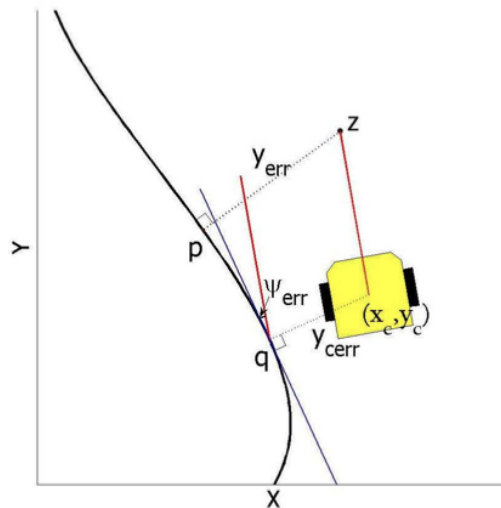


Figure 3.21: Path following using feedback control. The feedback error is measured from the point z to the closest curve point p . (Source: Choi et al. (2008))

On the other hand, feedforward control is based on the system proposed by Hayakawa et al. (2004). It introduces an additional look-ahead point that is placed between the rover and the feedback look-ahead point. The control system tries to predict the curvature of the trajectory ahead of the robot. For this

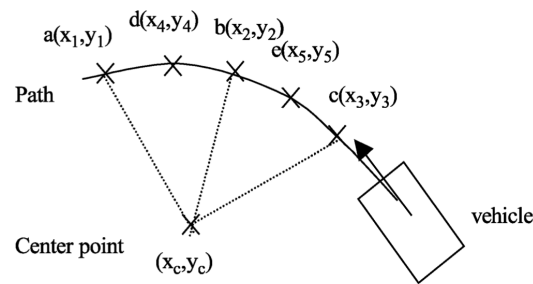


Figure 3.22: Curvature detection for feedforward control. (Source: Hayakawa et al. (2004))

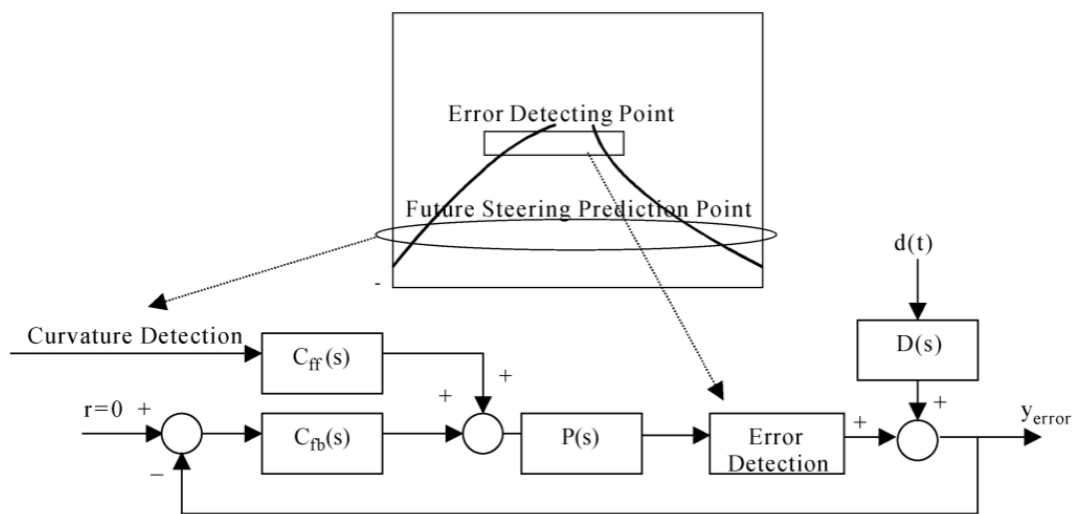


Figure 3.23: The model of the Ackermann steering system. The rectangle on the top depicts the route ahead of the rover with the feedback error detection point, which is rectangular, placed further away and the curvature prediction point, which is elliptical, placed closer to the rover. (Source: Hayakawa et al. (2004))

purpose, five sample points lying on the trajectory are selected with the middle one being equal to the feedforward look-ahead point. Three of them (the terminal points and the middle point) are used to estimate the center of rotation of given trajectory part. Afterwards, radii are calculated for all five points, producing one average radius that is used to calculate estimated curvature. The purpose of the feedforward control is to reduce an early turning on the sharp curves because the feedback look-ahead point can make the rover rotate too early. When used, the curvature prediction feature stops the rover from doing a sharp turn too early. The feedforward curvature detection example is shown in Figure 3.22.

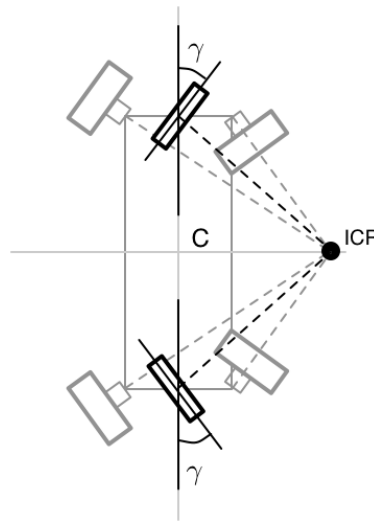


Figure 3.24: Control Bicycle model and the ICR point. γ is the turning angle produced by the PID controller and C is the geometrical center of the rover.

The block diagram of the proposed control systems combination is shown in Figure 3.23. The feedforward prediction result is joined together with the feedback error and given as a reference for a PID controller. Since the bicycle model was used, it had to produce one ICR point for control. It was decided that the ICR point should always be aligned with the Y axis of the rover, that is extending along the lateral direction from the center of the rover. Control signal of the PID controller was used as an angle for the front and rear wheels of the bicycle model. Knowing the offset of the front and rear axes from the center, simple geometry was applied to calculate the position of the ICR point. The example is shown in Figure 3.24.

3.3.4 Point-turn Path Following

Point-turn steering mode features three different states: *Stopped*, *Turning* and *Driving*. The state diagram with possible switching between these states is shown in Figure 3.25. In addition, switching between the states uses another state machine with three subsequent actions:

1. Wait for the wheels to stop rotating.
2. Wait for the wheels to achieve the new desired angles while standing still.

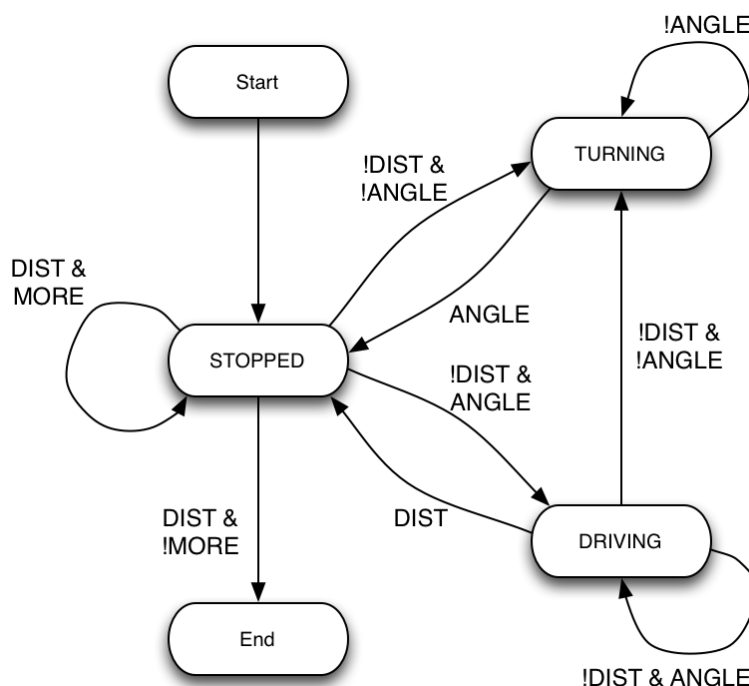


Figure 3.25: The state diagram of the path following routine in Point-turn steering mode. *DIST* variable means that the distance from the center of the rover to the current waypoint is acceptable; *ANGLE* means that the heading error is acceptable; *MORE* means that more waypoints exist and the current waypoint is not the end of the route. When the logic loops in *STOPPED* state (with state variables *DIST & MORE*), the current waypoint is incremented. In the *DRIVING* state the rover drives forward with the constant longitudinal velocity and zero lateral velocity; in the *TURNING* state the translational velocity of the rover is equal to zero and it is rotating around its center point.

3. Apply desired rotation to the wheels.

In the *Turning* state, the system compares actual heading with the desired heading. The rotation is controlled by a PID controller, and the action is considered finished when the heading error is small and the rotation rate is close to zero. In the *Driving* state, the rover uses Crab steering, where the Crab angle depends on the lateral deviation from the desired trajectory. This approach helps to compensate the errors in the rotation or lateral slippage. To avoid skidding while turning in place, the rover has to rotate its wheels in a proper way. For this purpose, the ICR-based control approach is used, where ICR point is placed in the geometrical center of the rover, as shown in Figure 3.26.

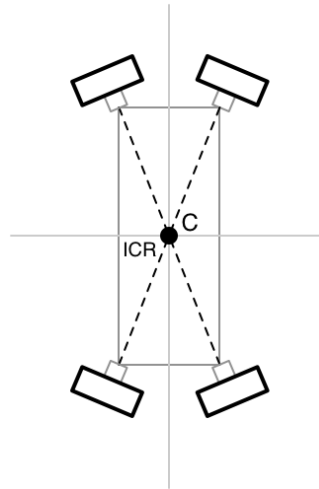


Figure 3.26: Point-turn approach of the all-wheel steered rover. C is the geometrical center of the rover.

3.3.5 Crab Path Following

The wheel formula of the all-wheel steered rover allows it to use Crab steering. Being useful only in particular cases, this steering mode is implemented as a supplement to other steering modes and cannot be set explicitly. Hence, the Crab steering is enabled, when the orientation of the rover remains correct, but the lateral deviation exists. It provides great assistance because without Crab steering, following scenarios could occur:

- In the Ackermann mode, the rover would produce a curve to correct its position, but it would result in unnecessary oscillations because of the nature of a PID controller.
- In the Point-turn mode, the lateral deviation from the desired trajectory would create heading error, which would be increasing as the rover approaches a waypoint. When this error becomes greater than a user-defined threshold, the rover stops and rotates to correct the error, which is time-consuming.

Furthermore, Crab steering becomes extremely useful when driving on the slope covered with a loose soil. If the rover is slipping in a lateral direction down the slope, the trajectory is automatically corrected by the Crab motion. The schematic of the Crab steering is shown in Figure 3.27, where γ is the steering

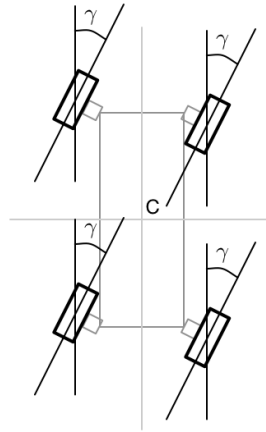


Figure 3.27: Crab control of the all-wheel steered rover. γ is the turning angle and C is the geometrical center of the rover.

angle that is proportional to the lateral deviation from the desired trajectory and cannot be greater than 65° defined by the mechanical constraints covered in Section 3.1.1. In case of the Ackermann steering mode, the translational velocity of the Crab maneuver is kept equal to the value that is recorded before it starts. There is also a lower limit of 0.15 m/s to avoid very slow motion. The Point-turn steering mode assigns the same fixed translational velocity to the Crab maneuver, which is used for driving forward.

3.3.6 Steering Mode Selection

The user can explicitly select Point-turn or Ackermann steering modes. The Crab steering mode cannot be explicitly selected, as explained in Section 3.3.5. It is also possible to let the system decide which steering mode to use on a particular part of the trajectory. The Ackermann mode is always a preference, however, the Point-turn mode is selected when Ackermann steering is not feasible. Currently, only kinematic constraints are used for this evaluation. The algorithm calculates the highest curvature of the curve and compares it with the smallest possible turning radius, defined by the mechanical constraints. When driving in a curve, the ICR is always aligned with the Y axis, as explained in Section 3.3.3. Thus, the turning radius is equal to the offset of the ICR from the geometrical center. The relation between the curvature and the turning radius is presented in Equation 2.1. The maximum curvature calculation uses the algorithm described by Deddi et al. (2000). If the middle control point p_1

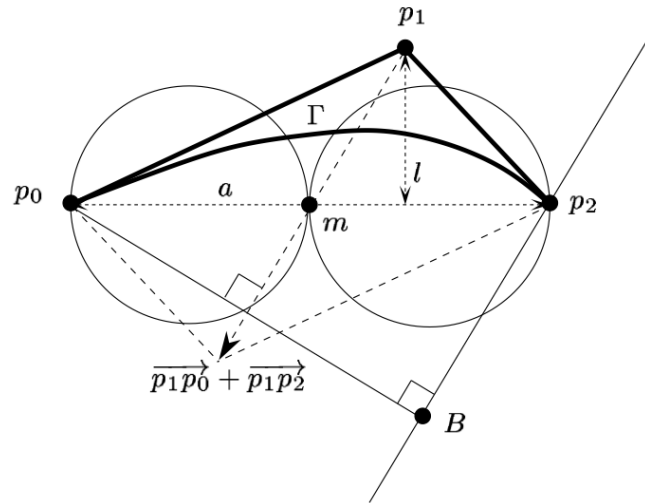


Figure 3.28: Calculation of the maximum curvature. p_0 , p_1 and p_2 are the control points of the Bézier curve Γ . The circles are used to determine the way of calculating maximum curvature. (Source: Deddi et al. (2000))

lies outside the circles shown in Figure 3.28, the maximum curvature is calculated according to the Equation 3.19, where m is the middle point between p_0 and p_2 , and A is the area of the triangle $p_0p_1p_2$. Otherwise the curvature is said to be monotone and is equal to the largest of two values, calculated from Equations 3.20 and 3.21.

$$k(\tau) = \frac{\|p_1m\|^3}{A^2} \quad (3.19)$$

$$k(0) = \frac{A}{\|p_0p_1\|^3} \quad (3.20)$$

$$k(1) = \frac{A}{\|p_1p_2\|^3} \quad (3.21)$$

The final trajectory is a set of paths, where each path has its own steering mode assigned.

3.4 Software Packages

It was decided to use Robot Operating System (ROS) as the foundation for the onboard software. The main advantage of this system is that it is easy to learn and it has a broad community of users, which facilitates the resolution of

common problems. Apart from creating a ROS package for the onboard software, ROS-based simulator plugins were also developed. Further explanation of the simulator plugins can be found in Section 3.5.2. The description of the onboard software package is presented in Section 3.4.1. For the human-robot interaction, a Graphical User Interface (GUI) was developed. It was decided to use cross-platform programming tools for the GUI to make it possible for the future groups to easily take over the development, regardless the types of the machines they will use. Even though, creation of a ROS-based GUI application would require less effort, at the beginning of the development only Linux-based ROS distribution had a stable release. Hence, it would limit the team to using Linux machines only. As a result, QT development environment was selected for the development of the GUI application.

3.4.1 Onboard Software Package

Onboard software features a single package consisting of several components called *nodes*. ROS nodes are separate processes, which communicate via so-called ROS topics (message passing interfaces built upon Transmission Control Protocol (TCP) sockets). Topics implement a publisher-subscriber pattern, where any node can declare itself as a publisher on a particular topic or subscribe to this topic. When a publisher sends a message, all the subscribers of the corresponding topic receive this message. Number of nodes were created to form a complete system, even though not all of them were used to test the motion control subsystem. These nodes are:

- **luna_driver** combines the path planner, path following control and processes commands for teleoperation.
- **luna_mech_gw** forms an intermediate layer between various hardware configurations or simulator interfaces. For example, it can listen to teleoperation commands from the joystick and produce appropriate commands for each wheel of the rover.
- **luna_gui_gw** sends telemetry information to the GUI application.
- **luna_gui_listener** listens to telecommands from the GUI application and distributes them among other nodes.

- **luna_slam** produces a grid-based map. Depending on the configuration, it can read a predefined map from a file or get the dynamic map from Gmapping node. In the future, it will be joined with the code from other team members.
- **luna_aut_gw** intended to process information from visual sensors, such as a laser ranging sensor and stereo camera.
- **luna_fear** watches for the nearby obstacles that can potentially create issues. Once found, it should instantly launch obstacle avoidance behavior.

The communication graph with launched nodes and topics between them is shown in Figure A.1.

Sometimes it is convenient to monitor the workflow of each node separately. Standard ROS tools allow launching several nodes together, but the output messages are not easily accessible. For the sake of development efficiency, shell script was created. It can launch the nodes in separate console windows and can be configured for any possible situation. The detailed description of the shell scripts is referred to in Appendix B.

3.4.2 Graphical User Interface

GUI application provides a tool for the user to interact with the rover. Its current functionality is intended to assist for the development of the motion control. For instance, it allows tweaking certain parameters, required for path generation and the path following routines. It also visualizes the current state of the rover and the world besides providing other useful information. At the beginning of the development, it was aimed that this application will be extended to form a complete user interface for all of the rover's subsystems. Based on the QT foundation, the application takes the advantage of cross-platform development and can be ported with a little effort to the different operating systems, including Linux, Windows and Mac OS. The application has a multiple windows user interface.

Main Window

The main window, which is shown once the application is launched, can be viewed in Figure 3.29. It is split into four sections: telemetry information on the left-hand side; the list of waypoints of the current path on the right-hand side; the world map along with the rover position and current path in the center; a basic control panel on the bottom. The telemetry table displays most of the parameters received from the rover. However, it does not show specific parameters that are shown in *All-Wheel Control* panel or *Trajectory Analysis* panel.

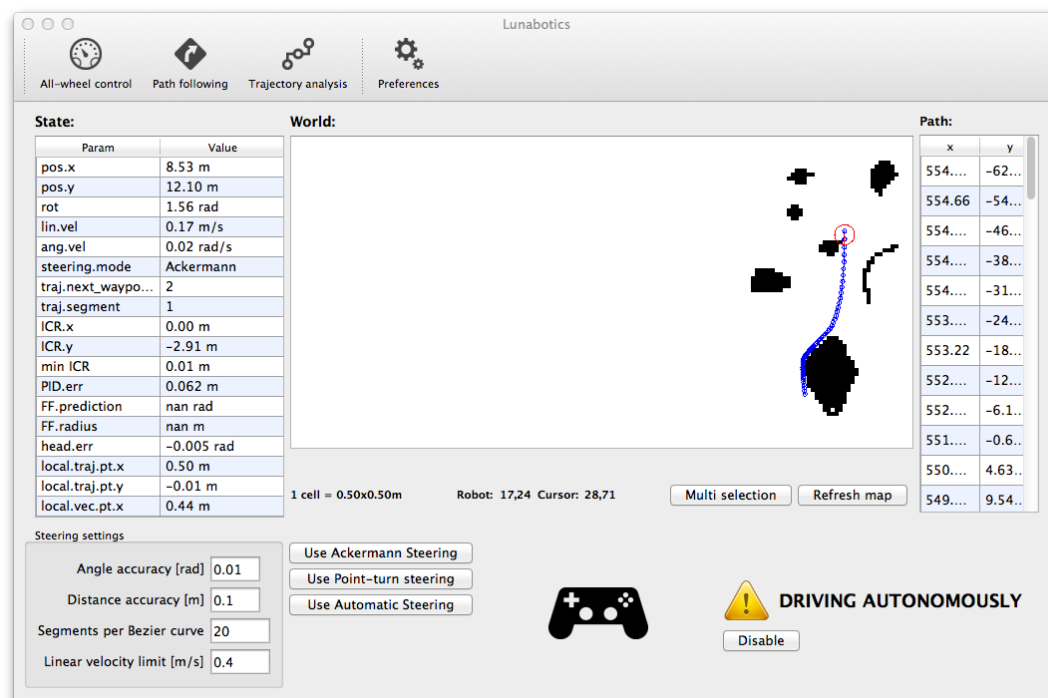


Figure 3.29: The main application window of the robot interaction software. In this example, the robot is using Ackermann steering.

The world map shows the occupancy grid with gray-scale tones, depending on the occupancy of the certain cell. The robot position is displayed by a red circle with the pointer defining its heading. The path is represented by a set of blue circles (waypoints), connected by blue lines. When the Point-turn steering mode is selected, waypoint circles are scaled to fill the entire cell because the waypoints are always at the center of a cell. In contrast, in other modes, waypoints are not bound to cells, thereby, blue circles are not scaled

but are of fixed size. When the trajectory curve following mode is active, the feedback error can also be seen on the map. It is represented by a purple line, connecting the feedback look-ahead point and the closest point of the trajectory.

Apart from the map, the world section shows the information about the map resolution, robot grid coordinates and the mouse cursor grid coordinates, which may be useful when manually setting a goal. Defining a goal for the motion control occurs by simply clicking the corresponding cell. The command is then sent to the rover that calculates the route, transfers the path to the GUI and starts driving in accordance with the selected steering mode. It is also possible to define intermediate waypoints. To switch to the multiple waypoints mode, user has to push the *Multi selection* button. He can then click multiple cells and push the *Send* button. The route will be calculated taking the selected waypoints into consideration, and the rover will start driving. The way of selecting multiple waypoints and the generated trajectory received from the onboard software is shown in Figure 3.30. The path section, displaying coordinates of the waypoints, is useful together with the Point-turn steering mode.

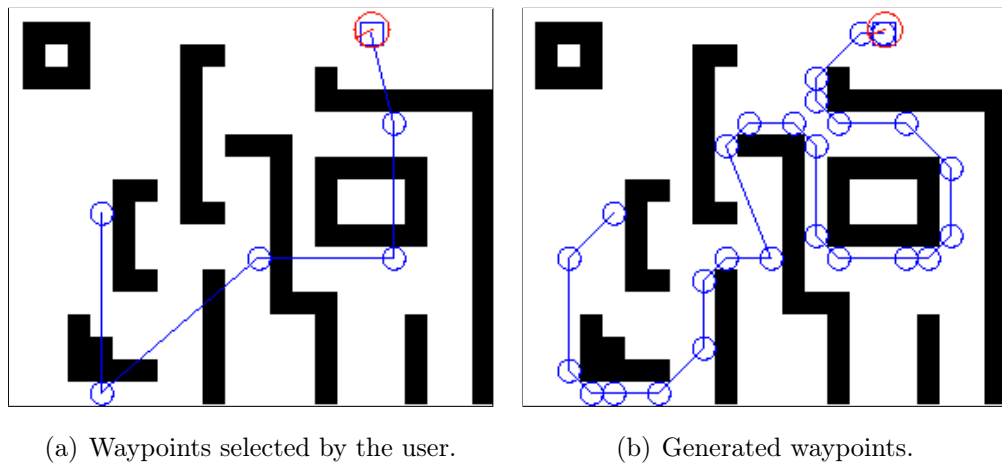


Figure 3.30: The example of selecting multiple waypoints. In this case, the trajectory was generated for the Point-turn steering mode.

The control section is used for basic control of the rover. Selection of steering modes is complemented by few parameters. Teleoperation is done by pressing F5, F6, F7, F8 keys for the forward, left, back and right commands, respectively. Also, it is possible to connect a USB game controller for teleoperation. The sketch of the game controller in the main window shows which buttons

are being pressed. Even though it is sufficient to teleoperate the rover, the advanced motion commands can be found in *All-Wheel Control* panel. When teleoperated using basic commands, the rover uses predefined *Drive forward*, *Drive backward*, *Turn left*, *Turn right* commands when only one appropriate button is pressed. When longitudinal motion should be joined with rotation, ICR-based control is used to drive the rover.

All-Wheel Control Window

The all-wheel control panel is used to control the chassis of a four-wheel rover in various ways. User can explicitly define the angles for each of the steering motors and the angular velocities for each of the driving motors. Besides that, the user is able to control motion by ICR, determine the heading angle and the linear velocity for the Crab steering or select one of the available ready-made commands. When defining the ICR point, it is relative to the center of the rover and uses the frame, where X axis extends forward in the longitudinal direction of the rover and the Y axis extends to the left from the center. The

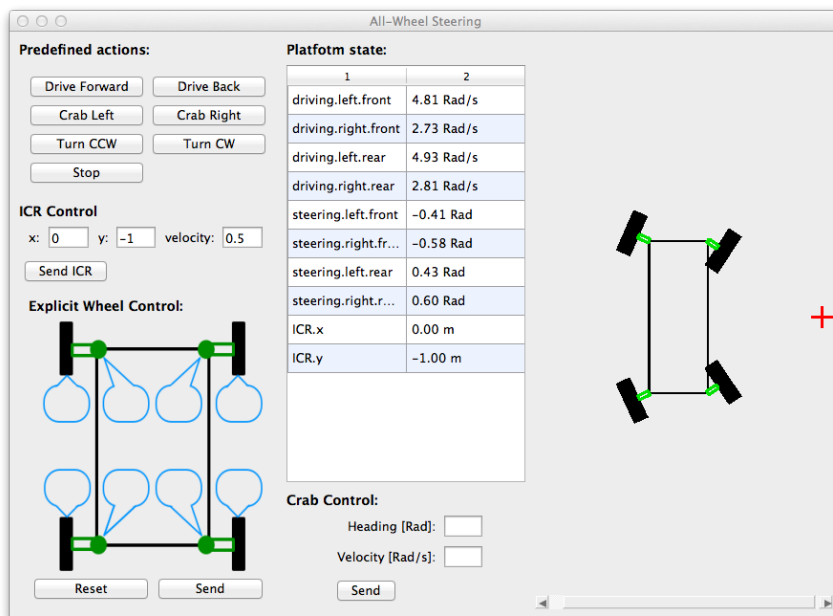


Figure 3.31: All-Wheel Control window. In this example, the rover is controlled to drive around the ICR with the offset of -1 meter along the Y axis and the linear velocity of the rover's center of 0.5 m/s. ICR point is represented by a red cross in the sketch.

window is shown in Figure 3.31. The table in the middle of the window provides information about the states of all the motors. Finally, the sketch on the right-hand side visualizes the current configuration of the chassis. The proportions of the sketch reflect those from the real rover, including positions of the steering motors, offsets of the wheels, thickness and radii of the wheels.

Path Following Window

The path following window is used to monitor the rover while it follows a curvy path. The window can be seen in Figure 3.32. Its upper part consists of a sketch, which shows the parameters of the feedback and feedforward control seen from the center of the rover. The lower part allows adjusting look-ahead point constraints for the feedback and feedforward control along with the gains of the PID controller.

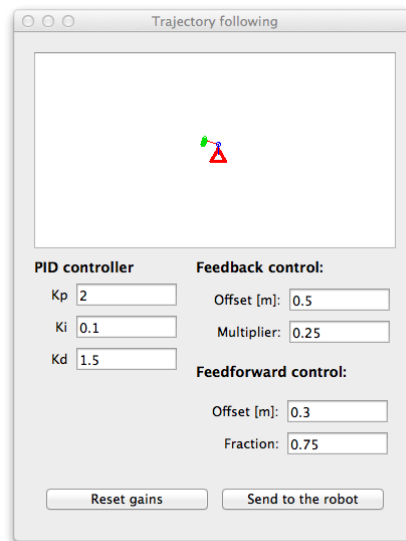
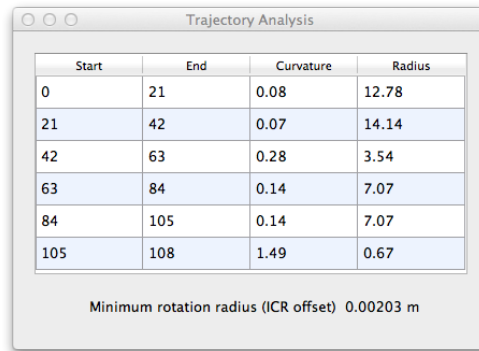


Figure 3.32: Path following window. On the sketch, the red triangle represents the rover; the blue circle extending upward from the triangle is the look-ahead point of the feedback control; the red line is the feedback error, connecting the feedback look-ahead point with the closest point of the trajectory; the five green points (on the left) are the points used in the feedforward control for the prediction of the curvature of the trajectory.



Start	End	Curvature	Radius
0	21	0.08	12.78
21	42	0.07	14.14
42	63	0.28	3.54
63	84	0.14	7.07
84	105	0.14	7.07
105	108	1.49	0.67

Minimum rotation radius (ICR offset) 0.00203 m

Figure 3.33: Trajectory Analysis window. Each row in the table represents one Bézier curve. The smallest rotation radius is the one recorded onboard the rover since it has started following the current trajectory.

Trajectory Analysis Window

Trajectory Analysis is a simple window showing separate curves, composing the final trajectory. Since the curves are eventually transformed into point sets, each of the rows in the table contains the starting and ending index of the curve in the point set of the whole trajectory. Each curve also has its maximum curvature calculated and the radius, corresponding to that curvature. Hence, this window helps to analyze the trajectory and the possibility for the rover to follow certain curves. The window is shown in Figure 3.33.

3.4.3 Communication

The communication between the GUI and the rover is based on TCP sockets. TCP is one of two low-level transport protocols widely used in network programming. User Datagram Protocol (UDP) is another protocol used in the network programming. The main difference between these protocols is that TCP always checks whether the data has been transmitted correctly. In contrast, UDP does not check it, what makes it more rapid. Hence, UDP is often used in applications like Voice over Internet Protocol (VoIP) clients, such as Skype, and TCP is used whenever data integrity should be preserved during the transmission. As stated above, two TCP sockets have been used via two ports: for the transmission from the GUI to the rover and vice-versa. The socket in the GUI is using `QTCP Socket`, which is a base class for the TCP sockets inside the QT library.

On the other hand, onboard software relies on Boost Asio, a C++ network programming library found in (Kohlhoff, 2013), to set up the communication.

The communication interface is based on protocol buffers library from Google, called *protobuf*. This library allows seamless serialization and deserialization of structured data. It ensures the minimal size of the bit stream by excluding unspecified data during the serialization process. It also ensures data integrity during the whole communication process. As explained in (Google Inc., 2012), any data structure can be defined in the specific format and saved in a file with the *.proto extension. Definition files are then compiled using command-line utility that generates C++ class files for each of the protobuf data structures. Resulting C++ classes contain getter and setter methods for each of the attributes as well as methods for serialization and parsing of the corresponding data structures.

Created data structures for communication between the GUI and onboard software includes the following items:

- **Point** contains data of a 2D point, namely its x and y coordinates.
- **Twist** contains data for linear and angular velocities.
- **Wheels** includes values of an arbitrary parameter for each wheel of a four-wheel rover.
- **AllWheelState** contains angles of steering motors and angular velocities of driving motors for each wheel of a four-wheeled rover by utilizing *Wheels* data structure.
- **AllWheelControl** is used for controlling the motion of a rover. The data structure has a flag to define the control type, which should be used: predefined actions, Crab steering, ICR-based control or explicit control of each motor. Also, the data structure includes control type-specific data, for instance, the ICR point relative to the robot's center or the type of the predefined action.
- **Teleoperation** is the simple structure, defining the joystick keys that are being pressed, namely: forward, back, left and right keys.

- **SteeringMode** defines a steering mode, such as Ackermann, Point-turn or Automatic. It also provides the steering settings, namely: heading accuracy, position accuracy, linear velocity limit and the number of segments per Bézier curve.
- **DefineRoute** contains one or more waypoints depending on the mode that has been used for pointing: single-selection or multiple-selection.
- **AdjustPID** contains information found in the *Path following* panel, namely: gains of the PID controller, feedback and feedforward control look-ahead point parameters.
- **Telecommand** is the container used to send actual commands from the GUI to onboard software. It defines a type of the command, such as teleoperation, steering mode change, disabling autonomy, defining a route, refreshing the map, adjusting path following parameters or sending a motion control command for a four-wheeled rover. Along with the type, this container provides a type-specific data for the command.
- **State** is a general telemetry of the rover. It can include the pose of the rover, its velocities and path following parameters.
- **World** is used to represent the world map. This data structure contains information about dimensions and resolution of the occupancy grid along with the occupancy information of each cell.
- **Path** is used to draw a path on top of the world map. This data structure includes waypoints of the path and the information about the different curves of the trajectory.
- **Geometry** when the dimensions of the rover are acquired either from the simulator or from the real hardware, they are transmitted to the GUI to make a correct representation of the chassis in the All-wheel control panel.
- **Telemetry** is the container used to send all the required data from the onboard software to the GUI. It can optionally include *State*, *World*, *Path* and *Geometry* data structures.

3.5 Simulation

Simulation is a crucial part of the project, since most of the algorithms are tested in simulators before being applied to the real system. It helps to detect mistakes that may cause the damage of hardware. Moreover, as in our case, simulation is the only way to test the system before the actual hardware has been manufactured. There are few simulators available which are compatible with ROS operating system: Stage, Gazebo, Webots and DARPA Robotics Challenge Simulator (DRCSim) by the Defense Advanced Research Projects Agency (DARPA). Webots simulator seems to be advanced, however, it was not investigated because it requires buying a license. The rest of the listed simulators are free to use. Nonetheless, DRCSim is built on top of Gazebo and adds just some convenience functions specific for the Robotics Challenge. Hence, there is no benefit to use it in our application, as we can use the Gazebo simulator instead. As a result, the Stage and Gazebo simulators were selected for the project.

3.5.1 Stage Simulator

The Stage simulator is a very basic application and was initially created to simulate robot swarms. It visualizes the world in two dimensions but can create a feeling of 3D environment by producing the isometric picture. The simulated environment is generated from bitmap files. The simulator accepts so-called twist messages (a combination of linear and angular velocity vectors) and can publish odometry and laser scan data. Being very basic, it can not simulate any noise. Hence, the published odometry data does not drift over time, as it happens in real life. An example of a simulation in Stage application from the top view and isometric view is shown in Figure 3.34.

This simulator was used to test the system in the early stage of the development when only the linear and angular velocities of the entire robot were calculated. Later, the system was tested on the Pioneer 3-DX robot that has a differential drive configuration. The published and received data formats of both Stage simulator and Pioneer 3-DX robot are identical to each other. The only difference between the simulated and real robot is the lack of physical prop-

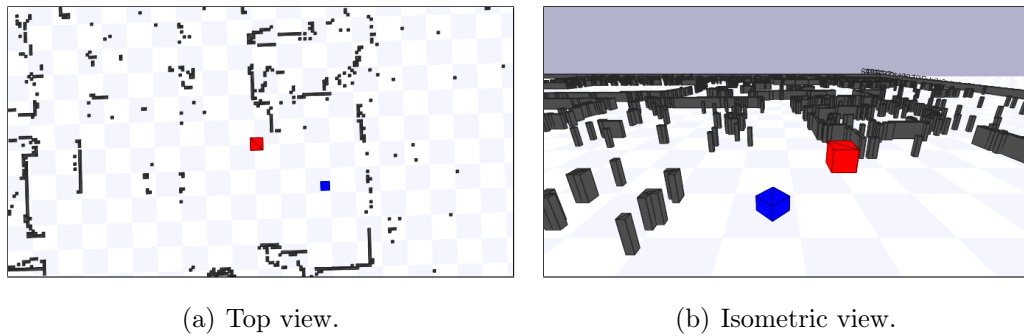


Figure 3.34: The environment in the Stage simulator. The blue square (smaller) is the operating robot, the red square (larger) is just a box, the black objects are obstacles.

erties in the simulator, for example, mass or friction. During this phase, the path generation method explained in Section 3.2.5 was tested together with a basic path following algorithm. Unfortunately, Stage simulator is not sufficient to conduct tests on all-wheel steering robots, as each individual wheel should be controlled separately. Thus, the Gazebo simulator, explained in Section 3.5.2, was used for the next step.

3.5.2 Gazebo Simulator

The Gazebo simulator is more advanced than Stage and provides 3D environment by utilizing OpenGL capabilities via Object-oriented Graphics Rendering Engine (OGRE) library. As stated in (Open Source Robotics Foundation, 2012), it is built upon Open Dynamics Engine (ODE), a library for simulating rigid body dynamics, and OPCODE, a collision detection library. The Gazebo simulator uses the following notations:

- **World:** It contains information about simulated environment, such as models present in this environment and physical properties of the world, for example, g-force.
- **Model:** It is any physical object contained by the environment. A model can be a robot, an obstacle, a ground plane or a light source. Just as in classical kinematics, a model is composed of a set of rigid bodies. These bodies are called links. Each link contains information about its visual

representation (how it looks in the simulator), collision model and physical properties, such as its mass, inertia and friction. The collision model can be different from the visual representation in order to reduce the computation load of the physics engine. For example, the collision model of a bookshelf can be described by a box, even though visually there are small handles and cabinets in the bookshelf. It is much easier to calculate collision with a box rather than a sophisticated object with many vertices. Moreover, in the given example approximation to the box is affordable. Models also contain information about their joints. The joints are the connections between links. Each joint specifies the links it is attached to and the joint type. As explained by Craig (2005), joints can be of six types, which are shown in Figure 3.35. However, only two types are commonly used: prismatic and revolute.

- **Sensor:** It is a perception object used by simulated robots. Commonly used sensors are lidars, sonars and cameras. The sensor object provides information, specific to every type of the sensor. For example, ray sensors contain information about their angular range and resolution, the number of samples and a distance range.

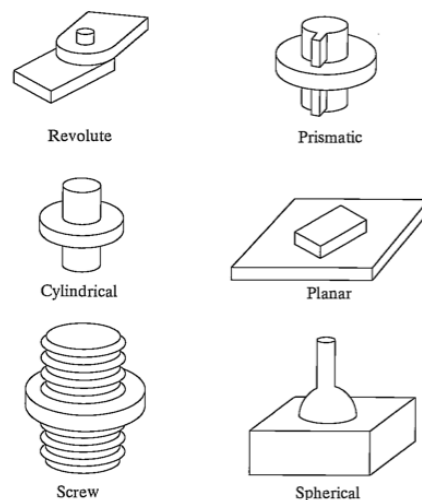


Figure 3.35: The six possible lower-pair joints. (Source: Craig (2005))

Each of the entities listed above are described in Simulation Description Format (SDF) that is based on Extensible Markup Language (XML). The model of an all-wheel steered robot was created using this format and can be seen in Figure 3.36. It is composed of the number of links, including chassis, wheels,

steering motors, driving motors, gearboxes, rear axle and a lidar sensor. For the lidar, the ready-made Hokuyo model was used. As this lidar has the angular range of 270° , it has been reduced to 180° to mimic the characteristics of the SICK lidar sensor that will be used in the real rover. In addition, the shape of the wheels was taken directly from the CAD model files. Regarding the dynamics, most of the links have approximate shapes and approximated dynamics models. Thus, the chassis and the steering modules are approximated to the box shape, while motors and gearboxes are represented as simple cylinders. Even though, they have real mass values, the inertia tensors matrices have been approximated to the corresponding shapes. These approximations are referred to in Table 3.1. The inertia tensor of a cylinder with mass m , radius r and height h , calculated from Equation 3.22, where Z axis is aligned with the height h . The inertia tensor of a tube is calculated from Equation 3.23, where m is the mass, r_1 is the inner radius, r_2 is the outer radius and h is the height. Finally, the box inertia tensors are represented by the matrix in Equation 3.24, where m is the mass, h is the height, w is the width and d is the depth. As only the specification of the chassis of the rover is complete, none of the secondary elements was assigned dynamic properties. For example, the bucket for storing regolith is purely visual and is just an approximate model of what will be placed in the future. The lidar dynamics is also omitted. Other sensors and parts, such as conveyors for collecting the regolith are not even present in the current model.

$$I = \begin{bmatrix} \frac{1}{12}m(3r^2 + h^2) & 0 & 0 \\ 0 & \frac{1}{12}m(3r^2 + h^2) & 0 \\ 0 & 0 & \frac{1}{2}mr^2 \end{bmatrix} \quad (3.22)$$

$$I = \begin{bmatrix} \frac{1}{12}m(3(r_1^2 + r_2^2) + h^2) & 0 & 0 \\ 0 & \frac{1}{12}m(3(r_1^2 + r_2^2) + h^2) & 0 \\ 0 & 0 & \frac{1}{2}m(r_1^2 + r_2^2) \end{bmatrix} \quad (3.23)$$

$$I = \begin{bmatrix} \frac{1}{12}m(3h^2 + d^2) & 0 & 0 \\ 0 & \frac{1}{12}m(3w^2 + d^2) & 0 \\ 0 & 0 & \frac{1}{12}m(3w^2 + h^2) \end{bmatrix} \quad (3.24)$$

All these links are glued together via revolute joints, where almost all the joints are fixed to create rigid connections. Only nine joints are not fixed, namely: four pairs of steering and driving pivot axes and the revolute joint connecting the rear axle with the rest of the chassis. The properties of these joints

can be seen in Table 3.2. The rotation ranges are defined by the mechanical constraints, while torque and frequency values are calculated from the nominal values of the motors and reduction rates of the corresponding transmission elements. For example, both steering and driving motors have rotation rates of 3000 RPM . Knowing that the steering worm drive has a transmission ratio of $1/75$, it is possible to calculate the frequency of the steering joint as $3000 \text{ RPM}/75 = 40 \text{ RPM}$. Similarly, the continuous torque of the steering joint has to be $0.22 \text{ Nm} \times 75 = 16.5 \text{ Nm}$, given that the continuous torque of the steering motor is 16.5 Nm .

The wheels contain specific surface properties, which define the slippage of the rover. Since in Coulomb friction model the friction coefficient depends on the properties of both contacting surfaces, it is not possible to get the exact coefficient. The wheels are known to have a polyurethane rubber layer that has friction coefficients provided in Table 3.3. Unfortunately, the classic law does not apply to the loose soil where the grip is a combination of the properties of individual particles. Moreover, ODE does not follow the exact Coulomb law, but uses approximation instead, as explained by Smith (2006). The approximated rule is defined by Equation 3.25, where f_T is the tangential force vector, f_N is the normal force vector. This equation creates a "friction cone" such that when the total friction force vector is within the cone, the contact is sticky. In contrast, it is sliding when the force vector comes out of the cone. Hence, μ represents the friction threshold to start sliding rather than conventional friction coefficient. Due to all above reasons, the μ value was not fixed for the rover, but it was varied for the different test cases until the rover behavior looked realistic in the simulator.

$$|f_T| \leq \mu \times |f_N| \quad (3.25)$$

Apart from having the friction property, the wheels were assigned a certain level of sponginess by introducing a soft Constraint Force Mixing (CFM) parameter, which is referred to by Smith. This parameter allows the contacting bodies to violate the constraints to a certain extent resulting a "soft" contact.

The Gazebo simulator allows a high level of control of nearly every aspect of the scene via plugins. It supports plugins of four different types: the system plugin, the world plugin, the model plugin and the sensor plugin. The system plugin is loaded on startup, whereas other types are loaded on demand and can

Property	Mass	Approximation	Dimensions	Inertia Tensor
Steering module	4 kg	Box	$w = 14 \text{ cm}$ $d = 16.5 \text{ cm}$ $h = 6 \text{ cm}$	$I_{xx} = 0.0103$ $I_{yy} = 0.0156$ $I_{zz} = 0.0077$
Wheel	6.64 kg	Tube	$r_1 = 11.7 \text{ cm}$ $r_2 = 14.95 \text{ cm}$ $h = 10 \text{ cm}$	$I_{xx} = 0.0599$ $I_{yy} = 0.0599$ $I_{zz} = 0.1197$
Driving motor	1.3 kg	Cylinder	$r = 2.75 \text{ cm}$ $h = 10.75 \text{ cm}$	$I_{xx} = 0.00149$ $I_{yy} = 0.00149$ $I_{zz} = 0.00008$
Gearbox	2.14 kg	Cylinder	$r = 3.5 \text{ cm}$ $h = 8.05 \text{ cm}$	$I_{xx} = 0.00181$ $I_{yy} = 0.00181$ $I_{zz} = 0.00131$
Chassis	20 kg	Box	$w = 16 \text{ cm}$ $d = 106 \text{ cm}$ $h = 16 \text{ cm}$	$I_{xx} = 0.0853$ $I_{yy} = 1.9153$ $I_{zz} = 1.9153$

Table 3.1: The properties of the individual links of the Gazebo rover model. Inertia tensors have only diagonal values because the rest values are zeros, as can be seen from Equations 3.22-3.24.

Property	Rotation range	Torque	Frequency
Steering joint	$[-90^\circ; 65^\circ]$	16.5 Nm	40 RPM
Driving joints	Unlimited	22 Nm	60 RPM
Rear axle pivot joint	$[-30^\circ; 30^\circ]$	-	-

Table 3.2: The properties of the individual joints of the Gazebo rover model. The rear axle pivot joint is a passive joint and does not have any torque or frequency limits.

Dry concrete	Wet concrete	Dry asphalt	Wet asphalt	Dry steel	Wet steel	Lubricated
0.6	0.4	0.4	0.2-0.3	0.2-0.4	0.1-0.2	< 0.1

Table 3.3: The friction coefficients of the polyurethane wheel layer on various surfaces.

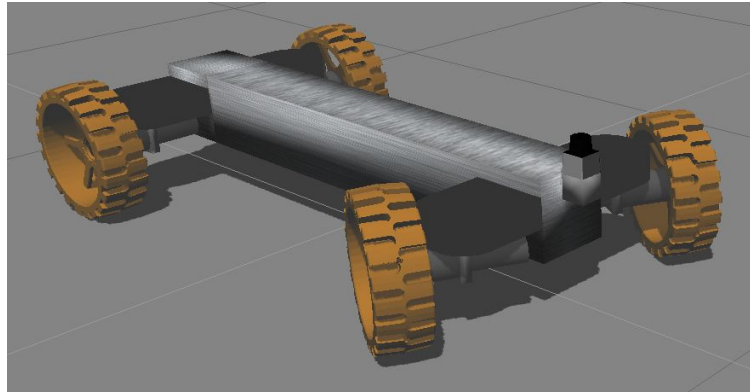


Figure 3.36: The model of the project robot on the simulated lunar surface. The components which do not contribute to the dynamics of the rover are hidden here.

access and modify the properties of the relevant entities. A number of plugins were created for the project. They act as ROS nodes; consequently, they can transfer information between the simulator and our system. These plugins are listed below.

- **Clock Server World Plugin:** It broadcasts the time information and helps to synchronize the internal clock of the simulator with the internal clock of the ROS system. The reason behind it is that the internal clock of the simulator can differ from the CPU real-time clock. Therefore, this synchronization should ensure more accurate operation of the simulated robot.
- **Transform Server Model Plugin:** ROS uses the mechanism of broadcasting messages with information about the transformations between various reference frames, for instance, sensor-to-robot or robot-to-world. It helps to keep certain subsystems generic and independent of a specific configuration. For example, RViz visualization tool requires transformation messages in order to correctly display the information that the robot is perceiving. In addition, Gmapping component, which offers ready-made SLAM service, requires the transformation information between the laser-centered frame and the robot-centered frame.
- **Odometry Model Plugin:** This plugin reads and publishes the odometry information from the simulator, including the robot position, heading, relative linear and angular velocities.

- **Lidar Model Plugin:** This plugin reads and publishes the data from the Hokuyo laser sensor installed on the simulated model.
- **Differential Drive Model Plugin:** This plugin was created for the model of Pioneer 3-DX robot used in the early stage of the development. It accepts the same control commands as Stage simulator and the real Pioneer 3-DX robot, that is, linear and angular velocity vectors.
- **All-wheel Steering Model Plugin:** This plugin allows to control all individual motors installed on the simulated chassis, including four steering motors and four driving motors. The steering motors are driven by the individual PID controllers. It allows to assign the azimuth angle for each steering motor to make it rotate the wheel in the given direction. In addition, this plugin also provides feedback when all the steering motors finish rotating to the desired angles. It is useful when performing a sequence of actions. For example, when using Crab steering, the robot first rotates its wheels, while standing still. After it gets the feedback that all the wheels finished rotating, it gives a command to the driving motors to start rotating. Velocities for the individual wheels are calculated as explained in Section 3.1.1.

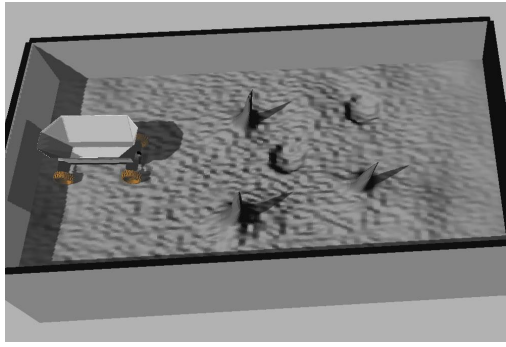
Chapter 4

Simulation Results

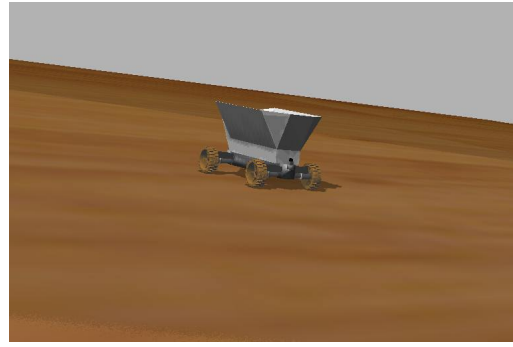
This chapter overviews the tests conducted in the Gazebo simulator and evaluates the performance of the developed system. Testing has been conducted purely by simulations because the rover hardware was not ready at the time of writing this Thesis. As the rover has been developed with the focus on Lunabotics Mining Competition, the priority was given to the tests inside LunArena simulated environment. The description of this environment and related tests can be found in Section 4.1. Sections 4.2 and 4.3 describe additional environments that were used for testing. All these environments are presented in Figure 4.1.

4.1 LunArena

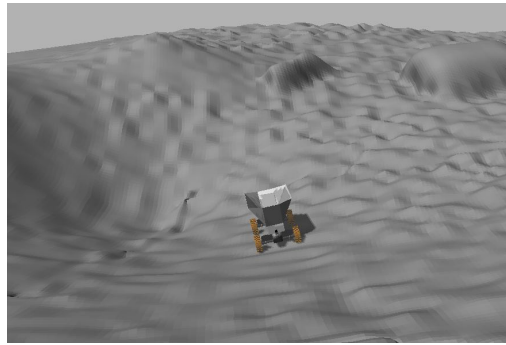
This LunArena environment replicates the area of the competition, including its dimensions, quantity of obstacles and their kind. Initially, it was aimed to integrate the motion control with SLAM subsystem that would provide a dynamically built map. For this purpose, the rock obstacles were transformed into high cliffs, such that the lidar sensor, positioned parallel to the surface, could observe them. LunArena walls were also raised for the same reason. However, as the SLAM system was not available at the time of testing, the dynamic map was not used. Instead, the static map was created by hand from the top-view image of the area. The rocks were marked as obstacles and inflated appropriately to suit the used inflated obstacle model. Craters were not marked as obstacles, as



(a) LunArena.



(b) Slippery slope.



(c) Moon-like surface.

Figure 4.1: Test environments created for the Gazebo simulator.

the rover was able to overcome them without significant problems. The resulting world map had 48×92 occupancy grid with each cell being $8 \times 8 \text{ cm}$. Path planning on this map took under five seconds to create the route between the furthest points of the LunArena, which is a very good result. The surface has a certain level of slippage to mimic the motion on the loose soil, such as lunar regolith.

At first, Ackermann steering with different velocity limits was tested, as presented in Figure 4.2. The trajectory was accurate when driving not faster than 0.5 m/s . Otherwise, high slippage occurred on the curves and the control algorithm was creating an oscillating trajectory of increasing amplitude, when trying to correct the error. For example, the test with the velocity limit of 0.6 m/s resulted in an unacceptable trajectory. Tests with 0.7 m/s (see Figure 4.2(d)) and 0.8 m/s that is not shown in the figure failed because the rover could not reach its final destination but crashed into obstacles. On the other hand, with the limit set to 0.2 m/s , Ackermann steering was able to not only follow the path accurately but also compensate the unevenness of the terrain,

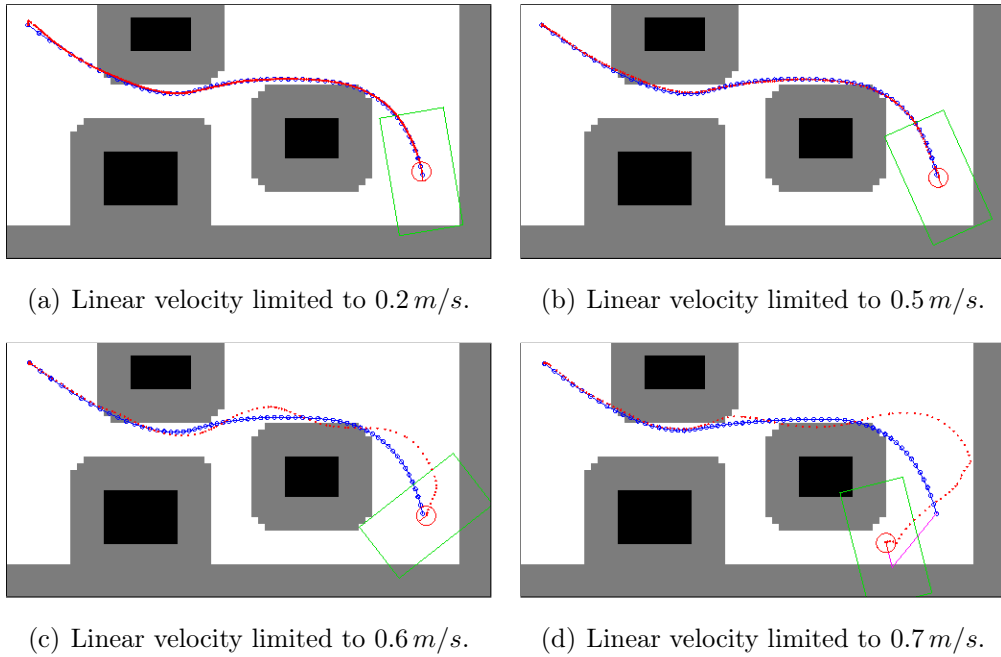


Figure 4.2: Ackermann steering in the LunArena environment with the cell size of $8 \times 8\text{ cm}$. The blue thick curve is the planned trajectory, whereas the red overlay line is the actual trajectory. The higher speed limit causes larger trajectory deviation after the turns. The linear velocity of 0.6 m/s and higher results in an unacceptable trajectory.

such as craters. Nonetheless, 0.5 m/s is quite a high speed for a planetary rover and is accurate enough. The times between getting a command from GUI and finishing the route were 74.883 s , 36.035 s and 27.839 s for velocity limits of 0.2 m/s , 0.5 m/s and 0.6 m/s respectively.

Once the 0.2 m/s was considered a "safe" option, it was used in the tests of the three types of steering modes presented in Figure 4.3. In these tests, all the settings together with start and end points of the trajectory were equal. Ackermann steering showed the best performance with the time of 60.489 s . Besides being a time-consuming mode (120.551 s), Point-turn was dramatically affected by the uneven terrain, even though it was very precise on the flat surface. In particular, the central crater seen in Figure 4.3(d) is located in the middle of the planned trajectory. The rotation in that area caused errors. Although the errors were corrected by Crab maneuver, it became clear that craters should be treated as obstacles to get the best performance from the Point-turn mode. The automatic steering selection was expected to benefit from the Point-turn

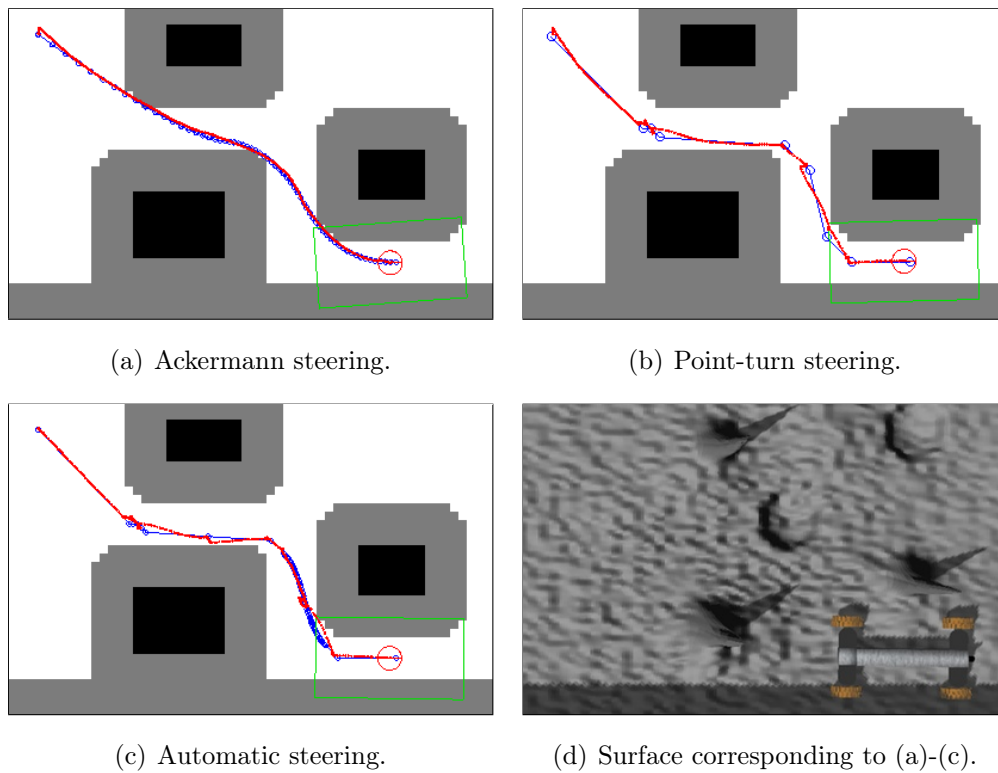


Figure 4.3: Driving test in the LunArena environment. The blue circles are the planned trajectory, whereas the red overlay line is the actual trajectory. The linear velocity limit was set to 0.2 m/s .

mode in areas where the high precision is required, for example, narrow passes. However, the performance of automatic steering selection was reduced by the Point-turn parts and eventually was worse than Ackermann steering. It also became clear that the selection algorithm needs to be improved because some short curves could have a high curvature value, although their influence on Ackermann trajectory is insignificant due to their size. Hence, Ackermann steering should be selected in those areas rather than Point-turn. The time spent on Automatic steering was 132.815 s .

4.2 Slope

The purpose of this environment was to test the ability of the rover to keep its attitude while drifting down the slope. Thereby, the time was not measured in these tests. As seen in Figure 4.1(b), this environment contains a plane inclined

by 11.5° with surface properties, such that the rover drifts down several centimeters per second when placed on it. The conducted test results are presented in Figure 4.4. The Automatic steering was not tested in this environment, as it is useless for wide open area, since Ackermann steering would be always chosen. This slope was represented by an obstacle-free map of size 100×100 where each cell was $8 \times 8 \text{ cm}$. The linear velocity limit was set to 0.2 m/s .

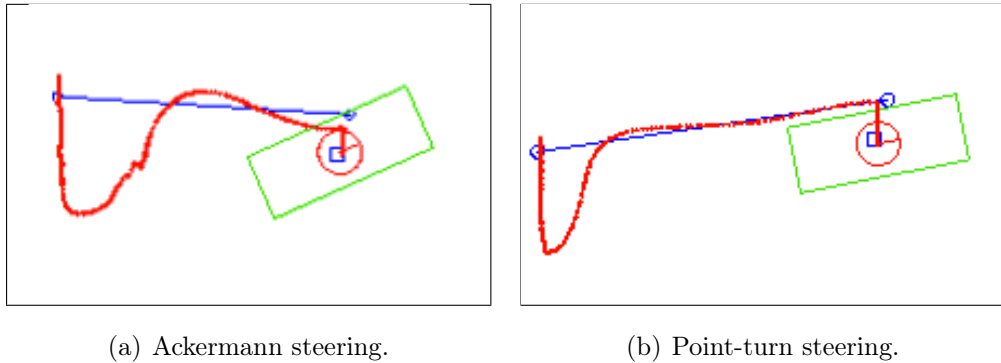


Figure 4.4: Path correction test on a slope. The blue line is the planned trajectory, whereas the red overlay line is the actual trajectory. The linear velocity limit was set to 0.2 m/s . Both tests have a huge spike at the beginning of the trajectory caused by the delay of the path planning.

The delay caused by path planning was of order of several tens of seconds. Consequently, in Figure 4.4, it is clearly seen that the rover was drifting down while standing still even after receiving a command from the user. However, this error was later corrected by the control algorithms. In the Point-turn mode, the path was corrected by the Crab maneuver, whereas Ackermann mode was using an ICR-based driving. Ackermann path following has higher oscillation because the orientation was changing, whereas the Crab maneuver would keep the orientation fixed. The ICR-based driving was selected over Crab maneuver because the initial orientation of the rover was different from the desired.

4.3 Lunar Surface

For the Moon-like environment, terrain with small hills and high slippage possibility was built, as shown in Figure 4.1(c). In the beginning, the gravity vector

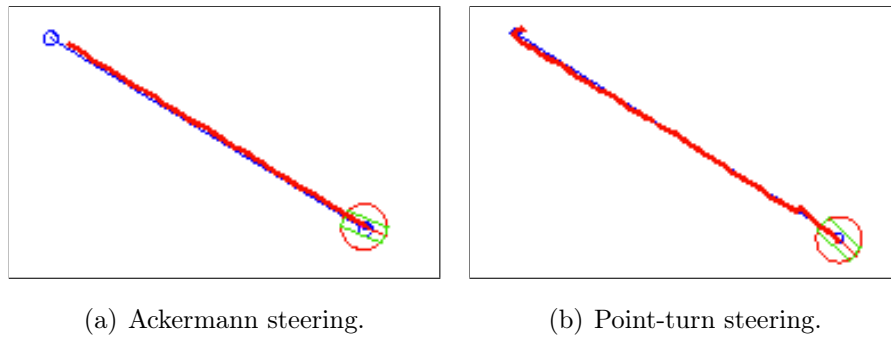


Figure 4.5: Driving on the Moon-like surface test. The blue line is the planned trajectory, whereas the red overlay line is the actual trajectory. The linear velocity limit was set to 0.2 m/s . Both tests are conducted on the same area of the surface.

was set to 1.622 m/s^2 , which is the gravity magnitude of the Moon. Unfortunately, the rover was tending to roll over even on relatively small hills, which did not look realistic. It was especially noticeable when turning in place, as the wheel base gets quite short in the lateral direction. Later, the gravity was set to the nominal magnitude for the Earth, that is 9.81 m/s^2 . It allowed to test the steering modes without rolling over. The Automatic steering was not tested here for the same reason as in Section 4.2. This environment was assigned the same occupancy grid as the slope in Section 4.2. Test results that are referred to in Figure 4.5 show excellent performance, despite a hilly surface. In both cases, the rover was driving at the speeds not higher than 0.2 m/s on the same part of the surface.

Chapter 5

Conclusions and Further Work

The main objective of this Thesis was to develop a motion control subsystem for the rover that is being built for participation in NASA Lunabotics Mining competition. The onboard software package has been developed and extensively tested using the Gazebo simulator. In addition, a cross-platform GUI application has been created. It gives user a flexible control over the motion control subsystem and allows adjusting certain parameters on-the-fly.

The review of existing planetary rover configurations and their performance helped to understand the importance of key design features, such as wheel configuration or suspension type. However, the only mobility system that has been extensively tested in practice is the one adopted by Sojourner, MER and MSL rovers. Rocker-bogie suspension on a six-wheel rover has been proven to be efficient. Nevertheless, researchers keep studying robot-soil interaction to be able to improve the autonomy of future rovers by reducing driving errors. In this project, the four-wheeled chassis was chosen due to the constrained budget and the ease of manufacturing. The fact that all the wheels are steered and actuated made it possible to use three different steering modes, namely: Ackermann, Point-turn and Crab steering. Ackermann and Point-turn steering modes were built independent, whereas Crab steering was designed as an assistive mode and is enabled automatically when needed. The onboard software, developed in C++ for ROS, was equipped with algorithms for path planning, path smoothing, path following and steering mode selection:

- Path planning relies on the modified A* algorithm with some features of the potential field and configuration sampling methods.
- The task of the path smoothing algorithm is to transform a set of straight path pieces into a set of Bézier curves.
- The Path following routine consists of several algorithms specific for Ackermann and Point-turn steering modes. Ackermann steering mode uses a combination of feedback and feedforward control techniques. In the Point-turn mode, the rover uses a PID controller to point towards the next waypoint, while straight driving relies on the Crab motion, where the angle is proportional to the lateral deviation from the trajectory.
- The Automatic steering mode bases its decisions on the maximum curvature of the particular piece of the trajectory. If the calculated curvature is possible to achieve via Ackermann steering, it is used by default; otherwise, the Point-turn mode is selected.

Apart from the onboard software package, a GUI application has been developed using Qt framework and, thereby, is compatible with various operating systems, including Windows, Linux and Mac OS. It communicates with the onboard software over the Internet using TCP sockets. The application has many features, including steering mode selection, the tuning of the control parameters, display of the state of the rover, its position on the world map, teleoperation and emergency stop. For the purpose of testing, a set of plugins for the Gazebo simulator was created. They provide full control over the rover inside the simulated environment. Once finished, the complete system was tested in different environments, including LunArena, slippery slope and Moon-like environment.

Tests confirmed the designed system to be sufficient for the competition. Crab steering was extremely useful in most of the tests. Furthermore, the rover was able to drive accurately enough with the speed up to 0.5 m/s , which is quite fast for a planetary rover. Nevertheless, the system comes with some drawbacks as well. First of all, the performance of the A* with the post-processing algorithms decreases exponentially with the growth of the world map. Therefore, a quick replanning is impossible with a really large map. Nonetheless, it did not cause any problems as the planning delay during LunArena test was only few seconds. Also, there is no rush during a real planetary mission, thereby, it is

believed to be a minor problem. Secondly, Point-turn mode was accurate only on the flat surfaces, while causing large errors on uneven terrain when the rover was rotating. Undoubtedly, despite being a time-consuming mode, Point-turn steering has to be used when the precision is a preference. Consequently, automatic steering selection was performing worse than Ackermann steering in the LunArena tests because of the Point-turn mode rotation errors. Apart from the list of drawbacks of the motion control, it was also impossible to simulate lunar regolith. Instead, a rigid slippery surface was used for testing. As no precise slippage values could be calculated, the slippage was adjusted in the simulator by hand until it appeared to be realistic.

Clearly, field testing on the working prototype is needed in the future, as no hardware was available at the time of writing. Moreover, to remedy the problems listed above, the further work implies the improvements of the relevant parts of the system. For example, path planning can adopt Field D* algorithm that is more suitable for navigation systems of the planetary rovers than conventional A*. Besides, the 3D map analysis should be used instead of 2D. It will benefit during the automatic steering selection because it will avoid switching to the Point-turn mode in the uneven spots. Also, the Automatic steering can be improved to assess not only the kinematic but also dynamic capabilities of Ackermann steering. For example, the comparison of the curvature change rate to the ability of the rover to adjust its wheels quickly, would allow avoiding Ackermann steering where it is not feasible for the rover. In addition, the Automatic steering could ignore small curves of high curvature which result in switching to Point-turn mode but, in fact, are not causing significant errors for Ackermann steering. Finally, the Crab steering can be improved for the use in emergency cases. For example, when one of the steering motors fails, the rover could rotate and align its wheels in such a way that it could drive by crabbing. The rover would remain operational with greatly reduced mobility.

To summarize, the developed system is sufficient for the competition, but it also has room for improvements by implementing ideas described above.

References

ABOUT.COM (2012). *Bezier Curve / Pierre Bézier*.

URL: <http://graphicssoft.about.com/od/glossary/l/blbezier.htm>

ANDRADE, G., AMAR, F., BIDAUD, P., AND CHATILA, R. (1998). *Modeling robot-soil interaction for planetary rover motion control*. In *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, volume 1, pages 576 –581 vol.1. doi:10.1109/IROS.1998.724680.

ARKIN, R.C. (1998). *Behavior-Based Robotics*. MIT Press, Cambridge.

BAJRACHARYA, M., MAIMONE, M., AND HELMICK, D. (2008). *Autonomy for Mars Rovers: Past, Present, and Future*. *Computer*, 41(12):44 –50. ISSN 0018-9162. doi:10.1109/MC.2008.479.

BARES, J. AND WHITTAKER, W. (1990). *Walking robot with a circulating gait*. In *Intelligent Robots and Systems '90. 'Towards a New Frontier of Applications', Proceedings. IROS '90. IEEE International Workshop on*, pages 809 –816 vol.2. doi:10.1109/IROS.1990.262500.

BAUER, R., LEUNG, W., AND BARFOOT, T. (2005). *Experimental and simulation results of wheel-soil interaction for planetary rovers*. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 586 – 591. doi:10.1109/IROS.2005.1545179.

BIESIADECKI, J. AND MAIMONE, M. (2006). *The Mars Exploration Rover surface mobility flight software driving ambition*. In *Aerospace Conference, 2006 IEEE*, page 15 pp. doi:10.1109/AERO.2006.1655723.

CARNEGIE MELLON UNIVERSITY (2011). *Uranus*.

URL: <http://www.cs.cmu.edu/afs/cs/user/gwp/www/robots/Uranus.html>

CARSTEN, J., RANKIN, A., FERGUSON, D., AND STENTZ, A. (2007). *Global Path Planning on Board the Mars Exploration Rovers*. In *Aerospace Conference, 2007 IEEE*, pages 1–11. ISSN 1095-323X. doi:10.1109/AERO.2007.352683.

CHOI, J.W., CURRY, R., AND ELKAIM, G. (2008). *Path Planning Based on Bezier Curve for Autonomous Ground Vehicles*. In *World Congress on Engineering and Computer Science 2008, WCECS '08. Advances in Electrical and Electronics Engineering - IAENG Special Edition of the*, pages 158–166. doi:10.1109/WCECS.2008.27.

CHOI, J.W., CURRY, R.E., AND ELKAIM, G.H. (2012). *Minimizing the maximum curvature of quadratic Bezier curves with a tetragonal concave polygonal boundary constraint*. *Comput. Aided Des.*, 44(4):311–319. ISSN 0010-4485. doi:10.1016/j.cad.2011.10.008.

URL: <http://dx.doi.org/10.1016/j.cad.2011.10.008>

CORDES, F., DETTMANN, A., AND KIRCHNER, F. (2011). *Locomotion modes for a hybrid wheeled-leg planetary rover*. In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pages 2586–2592. doi:10.1109/ROBIO.2011.6181694.

CRAIG, J.J. (2005). *Introduction to robotics : mechanics and control*. Pearson Education, Upper Saddle River, N.J. 07458, third edition edition. ISBN 0-13-123629-6.

URL: <http://opac.inria.fr/record=b1101583>

DEDDI, H., EVERETT, H., AND LAZARD, S. (2000). *Interpolation problem with curvature constraints*. In C.R..L.L.S. A. Cohen, editor, *Curve & Surface Fitting*. Vanderbilt University press. This book contain a selection of papers presented at the 4th International Conference on Curves & Surfaces, Saint-Malo, 1999. Contribution à un ouvrage. A00-R-390 || deddi00a A00-R-390 || deddi00a.

URL: <http://hal.inria.fr/inria-00099245>

DENG, Z., FAN, X., GAO, H., AND DING, L. (2011). *Influence analysis of terramechanics on conceptual design of manned lunar rover's locomotion system*. In *Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on*, volume 2, pages 645–648. doi:10.1109/EMEIT.2011.6023183.

DU, J., REN, M., ZHU, J., AND LIU, D. (2010). *Study on the dynamics and motion capability of the planetary rover with asymmetric mobility system*. In *Information and Automation (ICIA), 2010 IEEE International Conference on*, pages 682 –687. doi:10.1109/ICINFA.2010.5512479.

ENCYCLOPEDIA ASTRONAUTICA (2013). *Project Horizon*.

URL: <http://www.astronautix.com/articles/prohorizon.htm>

EXPLOREMARS.ORG (2012). *MSL Picture of the Day: T-4 Days: Mission Essentials*.

URL: <http://www.exploremars.org/msl-picture-of-the-day-t-4-days-mission-essentials>

FUKE, Y., APOSTOLOPOULOS, D., ROLLINS, E., SILBERMAN, J., AND WHITTAKER, W. (1995). *A prototype locomotion concept for a lunar robotic explorer*. In *Intelligent Vehicles '95 Symposium., Proceedings of the*, pages 382 –387. doi:10.1109/IVS.1995.528312.

GOOGLE INC. (2012). *Developer Guide*.

URL: <https://developers.google.com/protocol-buffers/docs/overview>

GRECENKO, A. (1992). *The slip and drift model of a wheel with tyre compared to some other attempts in this field*. *Journal of Terramechanics*, 29(6):599 – 604. ISSN 0022-4898. doi:10.1016/0022-4898(92)90039-M.

URL: <http://www.sciencedirect.com/science/article/pii/002248989290039M>

HAKENBERG, J.P. (2008). *Mobility and autonomous reconfiguration of Marsokhod*. Master's thesis, Helsinki University of Technology.

HARVEY, B. (2007). *Soviet and Russian lunar exploration*. Springer ; Published in association with Praxis Pub., Berlin; New York; Chichester, UK. ISBN 9780387218960 0387218963.

HAYAKAWA, Y., WHITE, R., KIMURA, T., AND NAITO, G. (2004). *Driver-compatible steering system for wide speed-range path following*. *Mechatronics, IEEE/ASME Transactions on*, 9(3):544–552. ISSN 1083-4435. doi:10.1109/TMECH.2004.834653.

HEADQUARTERS, DEPARTMENT OF THE ARMY (1993). *Technical Manual for Truck, Forklift; 6,000 lb. Variable Reach, Rough Terrain*. Integrated Publishing.

URL: http://constructionforklifts.tpub.com/TM10393066010/css/TM10-393066010_59.htm

HOEPFLINGER, M., KREBS, A., PRADALIER, C., LEE, C., OBSTEI, R., AND SIEGWART, R. (2008). *Description of the Locomotion Control Architecture on the ExoMars Rover Breadboard*. In *Proc. of The 10th ESA Workshop on Advanced Space Technologies for Robotics and Automation (ASTRA)*.

HWANG, J.H., ARKIN, R., AND KWON, D.S. (2003). *Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control*. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, pages 1444 – 1449 vol.2. doi:10.1109/IROS.2003.1248847.

INTEGRATED PUBLISHING (2003). *Construction Mechanic Basic. Integrated Publishing, Volume 01*.

URL: <http://www.tpub.com/eqopbas/27.htm>

ISHIGAMI, G., NAGATANI, K., AND YOSHIDA, K. (2006). *Path Following Control with Slip Compensation on Loose Soil for Exploration Rover*. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5552 –5557. doi:10.1109/IROS.2006.282271.

ISHIGAMI, G., NAGATANI, K., AND YOSHIDA, K. (2008). *Trafficability analysis for lunar/planetary exploration rover using Thrust-Cornering Characteristic Diagram*. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 2228 –2233. doi: 10.1109/IROS.2008.4651186.

ISHIGAMI, G. AND YOSHIDA, K. (2005). *Steering characteristics of an exploration rover on loose soil based on all-wheel dynamics model*. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3099 – 3104. doi:10.1109/IROS.2005.1545277.

JEAN-MARC MACLOU (2011). *The Wheel Reinvented*.

URL: <http://www.unusuallocomotion.com/pages/locomotion/the-wheel-reinvented.html>

KHOUKHI, A., BARON, L., AND BALAZINSKI, M. (2007). *Fuzzy Parking Manoeuvres of Wheeled Mobile Robots*. In *Fuzzy Information Processing*

Society, 2007. NAFIPS '07. Annual Meeting of the North American, pages 60–65. doi:10.1109/NAFIPS.2007.383811.

KOENIG, S. AND LIKHACHEV, M. (2002). *D*lite*. In *Eighteenth national conference on Artificial intelligence*, pages 476–483. American Association for Artificial Intelligence, Menlo Park, CA, USA. ISBN 0-262-51129-0.

URL: <http://dl.acm.org/citation.cfm?id=777092.777167>

KOHLHOFF, C. (2013). *Boost.Asio*.

URL: http://www.boost.org/doc/libs/1_53_0/doc/html/boost_asio.html

KULKARNI, N., IPPOLITO, C., KRISHNAKUMAR, K., AND AL-ALI, K. (2006). *Adaptive inner-loop rover control*. In *Space Mission Challenges for Information Technology, 2006. SMC-IT 2006. Second IEEE International Conference on*, pages 8 pp. –504. doi:10.1109/SMC-IT.2006.12.

KYRKI, V. (2012). *Locomotion, Kinematics and Motion Control; Lecture notes distributed in Field and Service Robotics, AS-84.3144*.

LIANG, Z., ZHENG, G., AND LI, J. (2012). *Automatic parking path optimization based on Bezier curve fitting*. In *Automation and Logistics (ICAL), 2012 IEEE International Conference on*, pages 583 –587. ISSN 2161-8151. doi:10.1109/ICAL.2012.6308145.

LINDEMANN, R. AND VOORHEES, C. (2005). *Mars Exploration Rover mobility assembly design, test and performance*. In *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, volume 1, pages 450 – 455 Vol. 1. doi:10.1109/ICSMC.2005.1571187.

MASTASCUSA, E.J. (2013). *PID Controllers - General PIDs*.

URL: <http://www.facstaff.bucknell.edu/mastascu/econtrolhtml/PID/PID3.html>

MONTES, N., MORA, M., AND TORNERO, J. (2007). *Trajectory Generation based on Rational Bezier Curves as Clothoids*. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 505 –510. ISSN 1931-0587. doi: 10.1109/IVS.2007.4290165.

MORALES, M., PEARCE, R., AND AMATO, N. (2007). *Analysis of the Evolution of C-Space Models built through Incremental Exploration*. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1029–1034. ISSN 1050-4729. doi:10.1109/ROBOT.2007.363120.

MORELAND, S., SKONIECZNY, K., WETTERGREEN, D., ASNANI, V., CREAGER, C., AND ORAVEC, H. (2011). *Inching locomotion for planetary rover mobility*. In *Aerospace Conference, 2011 IEEE*, pages 1 –6. ISSN 1095-323X. doi:10.1109/AERO.2011.5747265.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION (2012a). *Lunabotics Mining Competition*.

URL: <http://www.nasa.gov/lunabotics>

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION (2012b). *Mars Science Laboratory*.

URL: <http://mars.jpl.nasa.gov/msl/mission/>

NISHIDA, S., OKABAYASHI, Y., AND WAKABAYASHI, S. (2011). *Analyses and testing of new mobility system for lunar rover*. In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pages 2574 –2579. doi:10.1109/ROBIO.2011.6181692.

NISHIDA, S. AND WAKABAYASHI, S. (2010). *Analyses for mobility and control system of lunar rover*. In *SICE Annual Conference 2010, Proceedings of*, pages 799 –803.

OGATA, K. (2010). *Modern control engineering*. Instrumentation and controls series. Prentice Hall, 5th edition. ISBN 9780136156734.

URL: <http://books.google.fi/books?id=Wu5GpNAelzkC>

ONIGA, F., NEDEVSKI, S., DANESCU, R., AND MEINECKE, M. (2009). *Global map building based on occupancy grids detected from dense stereo in urban environments*. In *Intelligent Computer Communication and Processing, 2009. ICCP 2009. IEEE 5th International Conference on*, pages 111–117. doi:10.1109/ICCP.2009.5284776.

OPEN SOURCE ROBOTICS FOUNDATION (2012). *Gazebo Architecture*.

URL: <http://gazebosim.org/wiki/Architecture>

PEYNOT, T. AND LACROIX, S. (2003). *Enhanced locomotion control for a planetary rover*. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 311 – 316 vol.1. doi:10.1109/IROS.2003.1250646.

- PLANETARY RESOURCES (2013). *Mission*.
URL: <http://www.planetaryresources.com/mission/>
- RUSSELL, S. AND NORVIG, P. (2003a). *Artificial Intelligence: A Modern Approach*, chapter Solving Problems by Searching, pages 59–94. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition.
- RUSSELL, S. AND NORVIG, P. (2003b). *Artificial Intelligence: A Modern Approach*, chapter Informed Search and Exploration, pages 94–137. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition.
- SAFADI, H. (2007). *Local Path Planning Using Virtual Potential Field*. McGill University School of Computer Science.
URL: <http://www.cs.mcgill.ca/hsafad/robotics/>
- SEEGMILLER, N. AND WETTERGREEN, D. (2011). *Control of a passively steered rover using 3-D kinematics*. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 607 –612. ISSN 2153-0858. doi:10.1109/IROS.2011.6094930.
- SEENI, A., SCHAFFER, B., REBELE, B., AND TOLYARENKO, N. (2008). *Robot Mobility Concepts for Extraterrestrial Surface Exploration*. In *Aerospace Conference, 2008 IEEE*, pages 1 –14. ISSN 1095-323X. doi:10.1109/AERO.2008.4526237.
- SHAMAH, B. (1999). *Experimental Comparison of Skid Steering Vs. Explicit Steering for a Wheeled Mobile Robot*. Master’s thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.
- SMITH, R. (2006). *Open Dynamics Engine v0.5 User Guide*.
URL: <http://www.ode.org/ode-latest-userguide.html>
- SOCIETY OF ROBOTS (2013). *OMNI-WHEEL ROBOT - FUZZY*.
URL: http://www.societyofrobots.com/robot_omni_wheel.shtml
- SOHEL, F., KARMAKAR, G., AND DOOLEY, L. (2005). *A generic shape descriptor using Bezier curves*. In *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, volume 2, pages 95 – 100 Vol. 2. doi:10.1109/ITCC.2005.11.

SPUDIS, P. AND LAVOIE, T. (2011). *Using the resources of the Moon to create a permanent, cislunar space faring system*. In *Lunar Science Forum*. Lunar and Planetary Institute, NASA Marshall Space Flight Center, Huntsville AL USA.

SUTOH, M., NAGAOKA, K., NAGATANI, K., AND YOSHIDA, K. (2012). *Evaluation of influence of surface shape of locomotion mechanism on traveling performance of planetary rovers*. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3419 –3424. ISSN 1050-4729. doi:10.1109/ICRA.2012.6225024.

TANN, N. (2012). *July 25, 2012: Three of six 20 inch aluminum wheels*.
URL: <http://darkroom.baltimoresun.com/2012/08/nasa-lands-curiosity-rover-on-mars/three-of-six-20-inch-aluminum-wheels-are-seen-on-an-engineering-model-of-nasas-curiosity-mars-rover-as-it-navigates-a-sandy-mars-like-environment-named-the-mars-yard-at-nasas-jet-propulsion-labora/>

TAO, J., DENG, Z., HU, M., LIU, J., AND BI, Z. (2006). *A small wheeled robotic rover for planetary exploration*. In *Systems and Control in Aerospace and Astronautics, 2006. ISSCAA 2006. 1st International Symposium on*, pages 6 pp. –418. doi:10.1109/ISSCAA.2006.1627655.

U.S. AIR FORCE SYSTEMS COMMAND, SPACE SYSTEMS DIVISION (1961). *Lunar Expedition Plan Lunex*.
URL: <http://www.astronautix.com/data/lunex.pdf>

WOO-RAM LEE (2013). *MoVille, Tear Drop Shaped Futuristic Car Concept with Big Grin*.
URL: <http://www.tuvie.com/moville-tear-drop-shaped-futuristic-car-concept-with-big-grin/>

WÜRGLER, S. AND SUKKARIEH, S. (2010). *Path Planning for a Planetary Rover*. In *10th Australian Space Science Conference*. National Space Society of Australia, Australian Centre for Field Robotics, University of Sydney, NSW, Australia.

YLONEN, S. AND HALME, A. (2002). *WorkPartner - centaur like service robot*. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International*

Conference on, volume 1, pages 727–732 vol.1. doi:10.1109/IRDS.2002.1041477.

YU, X., DENG, Z., FANG, H., AND TAO, J. (2006). *Research on Locomotion Control of Lunar Rover with Six Cylinder-conical Wheels*. In *Robotics and Biomimetics, 2006. ROBIO '06. IEEE International Conference on*, pages 919 –923. doi:10.1109/ROBIO.2006.340343.

ZHANG, P., DENG, Z., HU, M., AND GAO, H. (2008). *Mobility performance analysis of lunar rover based on terramechanics*. In *Advanced Intelligent Mechatronics, 2008. AIM 2008. IEEE/ASME International Conference on*, pages 120 –125. doi:10.1109/AIM.2008.4601645.

Appendix A

Internal Communication of the ROS Nodes

Appendix B

Shell Scripts of the Onboard Software

To make the shell commands accessible from anywhere in the system, setup file was created. It can be found in the root directory of the package and launched by the command below.

```
> export 'roscd lunabotics '/setup.bash
```

After the setup, it is possible to launch the onboard software package using the command below.

```
> elias
```

This command is very flexible and can accept various parameters. For example, user can specify the location, where to search for the predefined map file, set the configuration to differential drive or four-wheel rover. The full description of available options can be reached by the command below.

```
> elias -h
```

The Stage simulator with the selected erratic world is launched using command

```
> lunastage
```

and the Gazebo simulator with a four-wheeled rover model can be launched with the command below.

```
> lunagazebo
```

The Gazebo simulator with a four-wheeled rover model inside LunArena environment can be launched with the command below.

```
> lunarena
```