Motion Control for a Planetary Exploration Rover with Six Steerable Wheels

Kyrre Glette

July 29, 2004

Abstract

IARES is a highly flexible planetary exploration demonstration rover developed mainly for autonomous navigation and locomotion studies. It has 19 degrees of freedom, including six active, steerable wheels. The rover has software for autonomous navigation, including stereo camera perception, path planning and motion control. It is complemented by a visual simulator that can substitute the real rover for practical purposes.

The goal of this MSc thesis, carried out at CNES in Toulouse, has been to make the most of the locomotion capabilities of this rover.

Given the hardware platform and and the software environment, the partly developed motion control system had to be studied. It was decided to improve parts of the existing motion control system by fixing smaller problems and doing adjustments to the behaviour.

Extensions in form of so-called crab-like movements were also studied and developed. This includes using the steering capabilities of all six wheels of the rover in order to obtain a translational movement. The extensions were implemented into the existing control algorithm.

The studies were conducted on a Solaris platform. Code was developed in C.

The new performance was tested in the simulator and the results were analysed. The six-wheel steering extensions provided more flexibility to the motion control, by allowing position and heading deviations to be corrected independently. The rover managed to correct its position deviations more quickly, and to increase its locomotion speed.

Foreword

This document reports the master thesis work of a student at the Department of Computer Science and Information Technology at the Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

The master thesis counts 30 study points and concludes the 5-year long engineering education which leads to a Master of Science degree.

The work was carried out at the Centre National d'Etudes Spatiales / French space agency (CNES) in Toulouse, France. An agreement between NTNU and CNES made this possible.

I would like to thank the people who assisted me in reaching my goals, especially Laurent Rastel, my supervisor at CNES, and Morten Hartmann, my supervisor at NTNU, for providing useful information, inputs, and feedback.

Toulouse, July 29, 2004

Kyrre Glette

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The report is organized as follows:

- Chapter 1 presents the project and states goals and motivation.
- Chapter 2 gives background information about the subject of planetary exploration.
- Chapter 3 presents the IARES rover and its associated software environment.
- Chapter 4 studies the rover's motion control system.
- Chapter 5 analyzes the current system's performance and proposes improvements.
- Chapter 6 describes general improvements made to the control system.
- Chapter 7 proposes a new locomotion mode and describes its details and implementation.
- Chapter 8 describes performance tests and their results.
- Chapter 9 discusses the results obtained and gives an evaluation of the work. Problems are discussed and future improvements suggested.

1.1 Setting

The Space robotics lab at the Centre National d'Etudes Spatiales (CNES) concentrates its work on planetary exploration rovers. In particular, their field of interest lies in autonomous navigation systems. This includes stereo vision, localization, path planning and locomotion. They currently have two rover prototypes (IARES

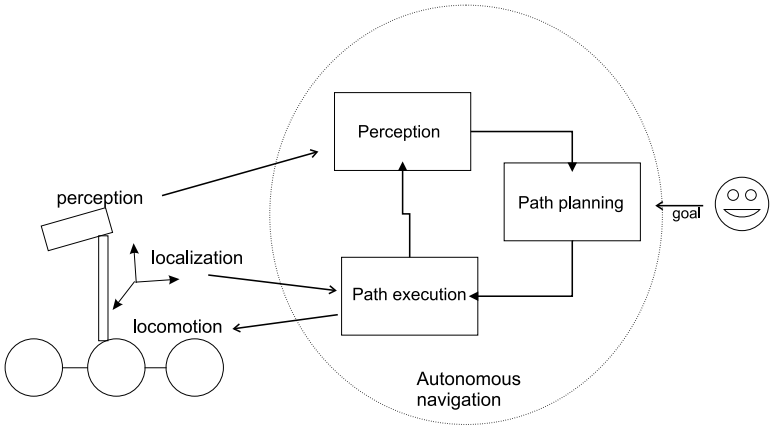

Figure 1.1: System overview. The autonomous navigation system is given a goal from an operator. Communication with the rover chassis provides the system with perception and localization data. Commands are sent from the navigation system to the chassis locomotion system.

and COMARO) for experiments, however, none of them are operational at the moment. The lab has also developed a simulator which is useful for designing, software development, and testing. There is an ongoing cooperation with the European Space Agency (ESA) about the future ExoMars Mars mission.

1.2 Goals and motivation

Given the initial subject title, “Coordinated motion control for the IARES rover”, the project has been somewhat open. The overall goal has been to study the existing motion control part of the CNES autonomous navigation system and find out how to improve it.

An general overview of the autonomous navigation system can be seen in figure 1.1. This system consists of three main subsystems: Perception, path planning, and path execution. The focus for this subject is the path execution subsystem. This subsystem receives a path to execute from the path planning subsystem. It also communicates with the rover chassis to receive localization data. Commands are sent to the chassis locomotion system (motors) in order to move the rover.

It was hoped that the rover, at the time being situated on an indoor test rack, would become testable on the outdoor Mars-like environment test site. This was however dependent on other projects and could not be accomplished by this project alone.

An operational rover would be a good way to demonstrate the performance of the autonomous navigation, including the motion control. However, even without an

operational rover, the rover simulator serves as an effective demonstration platform.

Improving the motion control system can lead to benefits in areas such as flexibility, speed, energy use and reliability, as well as improved localization.

Investigating new ways of locomotion by utilizing some of the IARES rover's mechanical capabilities can provide useful information for the design of possible future planetary missions.

Chapter 2

Mars surface exploration

Mars exploration has been interesting to mankind for a long time. The Americans, Russians and now also the Europeans have all had missions to Mars, more or less successful. (In fact, 22 out of 35 missions to Mars have been lost [7].) There are several risks and problems connected to such missions, and this places high demands to the mission equipment. This chapter presents some aspects to be considered for Mars exploration missions and then introduces some actual Mars rover models.

2.1 Mars surface exploration considerations

Mars' surface is in general dusty with a more or less even distribution of rocks. Rover dimensions have to be designed according to rock size and distribution. They have to have locomotion systems capable of overcoming difficult terrain. As dust storms are common on Mars, it is important that the science equipment is well protected and the mechanical parts are designed accordingly [7, 1].

Power consumption is an important area of concern. Given the cold nights on Mars (down to -135°C), energy is required to heat the electronics. Solar panels can only be used during daytime, and the solar flux received by solar panels on Mars is less than half the amount received on Earth. In addition, there are problems with dust gathering on the solar panels. Therefore, additional power sources are needed, like batteries or radioisotope thermal generators [9, 7].

Communication with Earth can be done by a direct rover-Earth link or via a satellite orbiting the planet. A direct link from the rover could transmit up to a few kilobits per second, but it requires to point the antenna, and would cost up to a few tens of Watts for the transmission. A less energy-consuming way of communication would be via a satellite orbiting the planet or via the the surface descent module.

This would increase the transmission rate too, however, a new point of failure is added (the ExoMars report estimates the chance of mission loss due to orbiter failure is 4 out of 10) [7, 9].

The transmission delays are long, usually around a few tens of minutes, because of the long propagation time of the signal. This makes direct telecommanding of a vehicle very slow and practically impossible for difficult terrain. Some degree of rover autonomy makes the exploration more efficient and increases the maximum distance a rover can travel each day. As mentioned, the power sources are limited and will only last a certain number of days, so efficient exploration is of utmost concern [9].

2.2 Planetary rovers

This section presents some different Mars exploration rover models. The IARES rover will be presented later, in chapter 3. Although somewhat different in style, these rovers are all six-wheeled and have the same basic concepts. Although other locomotion concepts have been studied, only wheeled designs have been chosen for missions so far because of complexity and energy issues. Tracked designs have been considered, but this introduces difficulties with localization and energy consumption. Legged designs have not received much attention, because of increased mechanical and control complexity, as well as high energy consumption compared to a wheel-based design [7, 6].

Six-wheeled rovers offer a good compromise between weight/energy and mobility. Six wheels offer considerably better obstacle clearing capabilities than four wheels, and there is an increased robustness to motor breakdown. However, eight wheels or more becomes costly in terms of weight and energy consumption, compared to what it offers in increased mobility [7, 6].

2.2.1 The EVE rover

The EVE rover was used at CNES before the IARES rover[9]. It was developed for studying different remote control and autonomous navigation modes suitable for Moon and Mars missions. See table 2.1 for characteristics.

The rover featured a pair of stereoscopic cameras for perception. The onboard computer was a VME Rack with a PowerPC 601 CPU board at 100 MHz. It ran a LynxOS operating system. Much of the CNES autonomous navigation and teleoperating software was tested out with this rover, however, the navigation was not calculated on the onboard computer.

Figure 2.1: The EVE rover on CNES test site. Image: CNES

Total Weight	120 kg
Length	120 cm
Width	90 cm
Max. speed	15 cm/s
Max. step size	25 cm
Max. slope	20°

Table 2.1: EVE characteristics. The max. step size indicates the maximum height of an obstacle the rover is able to overcome. Max. slope is the maximum slope inclination the rover is able to climb.

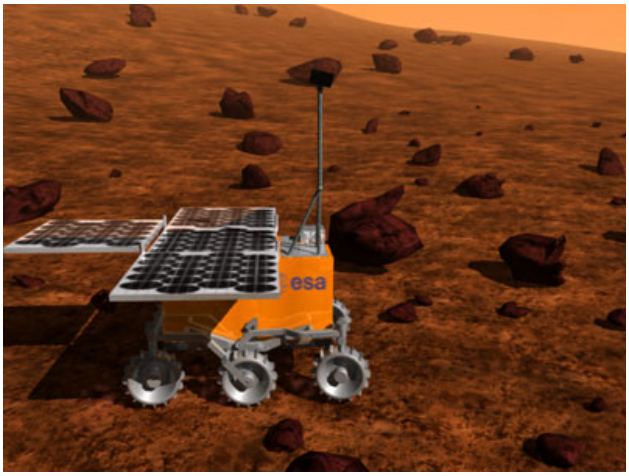

Figure 2.2: Artist's conception of the ExoMars rover. Image: ESA

The chassis was developed by VNII Transmash[1], a Russian company with long experience in rover construction. The chassis is of the Marsokhod family, which has also been used by NASA and LAAS [6].

The EVE chassis was designed with difficult terrain in mind. It has six wheels which have a conical shape for easier obstacle clearing, as it reduces the chance of the rover getting stuck on rocks below its body. All wheels have independently adjustable speeds, but no angular steering. The axles have free rotation around the roll axis. Deformation of the chassis is possible, which allows for a so-called wheel-walking, or peristaltic movement. See section 4.1.4.

A disadvantage to this design, with a possible Mars mission in mind, is the chassis' modular shape, which makes it hard to find place for a payload, as well as solar cell panels. Another problem is the lack of steering on the wheels. All turning movements will cause skidding and will therefore not be optimal in terms of energy use.

2.2.2 The Exomader/ExoMars rover

The ExoMaDeR (ExoMars Demonstration Rover) is a rover developed by ESA for investigating a possible mission to mars in 2009 (ExoMars09). The ExoMars mission is part of the ESA Aurora programme for exploring the Solar system [4].

The Exomader chassis is also developed by VNII Transmash, and is a scaled prototype of the planned ExoMars rover. Exomader will be tested at ESTEC in the Netherlands. See table 2.2 for characteristics. The navigation system for the Exo-

[1] Now called Rover Company Limited (RCL) [14].

Weight	190 kg
Length	130 cm
Width	114 cm
Max. speed	3.3 cm/s
Max. step size	30 cm
Max. slope	25°

Table 2.2: ExoMars rover characteristics

Mars rover will build on CNES' work in this field. CNES has therefore integrated a model of this rover into its rover simulator (see 3.2.2) to be able to take this into consideration when developing. The ExoMars rover is expected to be highly autonomous and therefore able to cover long distances, as much as 500m in 5 hours [7].

In contrast to the EVE and IARES rovers this rover does not have a system of three independent axles. A bogie system has been developed that keeps the main platform very stable and upright, while still having good mobility on uneven terrain. This big central platform makes it very practical for distribution of the payload, and so this design can be said to be more mission-oriented.

Of the six wheels, the front and rear wheels have independent steering, which allows the rover to turn in place and drive in gradual arcs. See section 4.1 for more details. However, as the specifications are not completely fixed yet, there may be changes in the configuration according to mission priorities. For instance it could be decided to implement steering on all 6 wheels in order to have extended locomotion capabilities.

2.2.3 The Mars Exploration Rovers

The NASA Mars Exploration Rovers (MER), Spirit and Opportunity, were launched in June and July 2003 and landed on Mars in January 2004. The landings were successful, and they are, at the time of writing, still exploring the Martian surface. Originally planned to explore the surface for only three months, this time has now been almost doubled [8, 12].

The FIDO and Athena SDM rovers are corresponding prototype rovers. Developed at the Jet Propulsion Laboratory (JPL), they have served as platforms for developing navigation and mobility algorithms [3].

The MERs are powered by one Rad 6000 32-bit radiation-hardened PowerPC CPU each, running at 20Mhz. They have 128MB of RAM and additional an 256MB Flash RAM, which is a considerable improvement over the Sojourner rover. Software runs on a VxWorks operating system [3, 12].

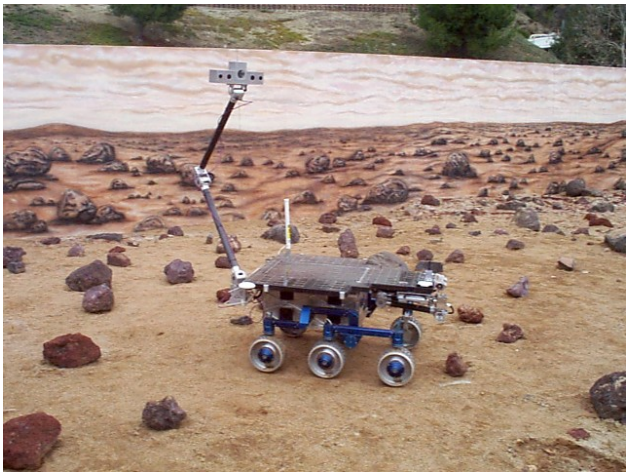

Figure 2.3: The FIDO rover. Image: JPL

Total Weight	174 kg
Length	160 cm
Width	230 cm
Max. speed	5 cm/s
Max. step size	NA
Max. slope	>25°

Table 2.3: Mars Exploration Rover characteristics

In addition to the pair of stereo navigation cameras mounted on the mast, they also have two stereo pairs of hazard-identification cameras mounted at front and back. These cameras are an extra aid to the navigation, spotting possible immediate dangers.

The rovers have 6 wheels, with independent steering of the front and rear wheels. This allows them to turn in place or drive in gradual arcs, like the current configuration for the ExoMars rover. The mechanical wheel suspension system uses the JPL-patented Rocker-Bogie principle, which gives the capability to climb obstacles up to at least one and a half the size of the wheel diameter [2].

The rover design is a result of trade-offs made for a real exploratory mission. Obstacle clearing capability is somewhat limited compared to the IARES or EVE rovers, however this rover has more room for its payload and an area for its solar cell panels (1.3 m^2), necessary for energy supply. Characteristics can be found in table 2.3.

Although the MER landings have been successful, the MERs are quite slow in

exploring the Martian surface. Only 20 meters are covered per day. There is no autonomous navigation (path planning) system on board these vehicles, only monitoring of path execution. This means that a high degree of monitoring and command from Earth is necessary. As discussed in section 2.1, this reduces the efficiency and thus the return on the mission investment [12].

Chapter 3

The IARES development platform

This chapter presents the IARES rover and goes into detail about its mechanical properties, its hardware system, and its software system and environment.

The IARES (Illustrateur Autonome de Robotique d'Exploration Spatiale, or Space Exploration Rover Autonomous Illustrator) vehicle is CNES' current planetary rover prototype. See figure 3.1. It is the successor to the EVE (see section 2.2.1) rover and has some improvements over this, notably in mechanical flexibility, location systems, perception, computing power, and navigation.

As the name indicates this is a demonstrator rover and is not intended for a real mars mission. Emphasis has not been on chassis weight, energy consumption or place for payload (including energy sources), but rather on a very high degree of chassis flexibility. This gives the possibility to experiment with new types of

Figure 3.1: The IARES rover on CNES test site. Image: CNES

locomotion control, including advanced obstacle clearing [9].

3.1 Hardware platform

This section describes the IARES chassis and its mechanical properties. It also describes the computer hardware on board.

The information for this section has been found in [11, 10] and from personal communication with L. Rastel.

3.1.1 Mechanical properties

The IARES chassis was delivered to CNES in 1996 by St. Petersburg-based VNII Transmash. In order to make thorough locomotion experimentation possible, the IARES chassis has a high degree of flexibility. This is provided by its 19 degrees of freedom, of which 17 are active. It results in the following highlights:

- Independent speed and steering on all six wheels. This allows for advanced steering modes, as will be discussed in section 4.1.
- Deformation of chassis. The inter-axle distance can be adjusted, and the wheels on one axle can be lifted from the ground. This makes peristaltic locomotion (see section 4.1.4) and climbing of obstacles possible. See figure 3.2.
- Sideways and backwards/forwards tilting. Tilting can be useful when the rover drives on slopes, as the chassis can tilt in a way such that the mast and payload still lies horizontally. This allows for better stability and improved perception. Tilting can also be used in combination with chassis deformation for obstacle negotiation.

The chassis body is divided in three sections, each containing one axle. They are connected with motorized levers that can change the inter-axle distance. See figure 3.3. The middle axle has a motor that controls the sideways tilting, and the front and back axles have passive joints. This lets the rover adjust itself automatically corresponding to the terrain features.

A pair of “obstacle rollers” are mounted on each axle, see figure 3.4. They are connected to the wheel motors and are rolling at the same linear speed as the wheels. This helps the rover overcome obstacles on which it might otherwise get stuck, like rocks coming between its wheels. The Marsokhod chassis (see section 2.2.1) overcomes such problems by its conical wheels. However, as IARES uses steering on the wheels, the roller solution was proposed.

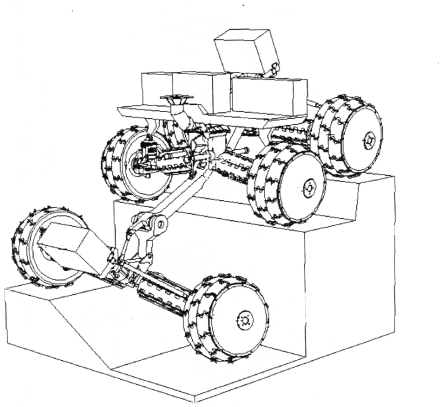

Figure 3.2: Complex obstacle negotiation. The chassis deforms to overcome difficult situations. Figure: VNII Transmash

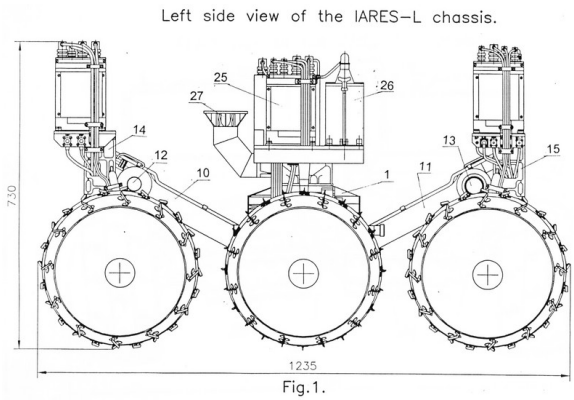

Figure 3.3: Left view of the IARES rover, in its contracted state. The different sections are clearly visible. The levers (10) and (11) can be extended to make the body longer. The perception mast mount can be seen at (27). Figure: VNII Transmash

Weight	170 kg
Length	85-135 cm
Width	120 cm
Max. speed	35 cm/s
Max. step size	50 cm
Max. slope	40°

Table 3.1: IARES characteristics

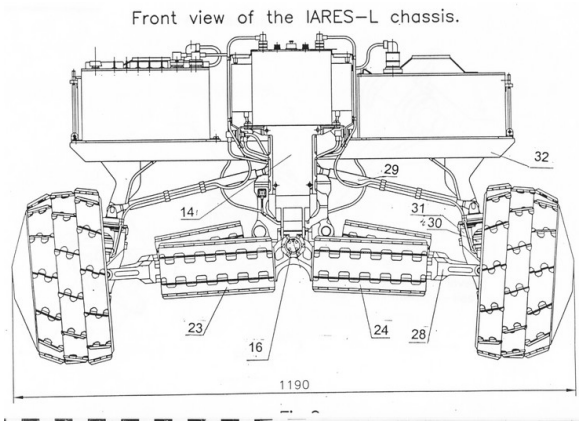

Figure 3.4: Front view of the IARES rover. The obstacle rollers can be seen at (23) and (24). Figure: VNII Transmash

As a result of the high chassis flexibility, the IARES vehicle has very good locomotion capabilities compared to the other rovers studied. See table 3.1. Much of the high step mounting capability can be credited to the chassis deformation ability. The front wheels can be lifted while the other wheels are on the ground, in order to get a grip on the obstacle. The high slope angle can also be explained from the deformation ability. In particular, a method referred to as peristaltic locomotion, or wheel-walking, is used. See section 4.1.4.

3.1.2 Processing unit

In 1996 the IARES was equipped with a Transputer, with software delivered from KFKI, Hungary. The Transputer is a node computer. It has several high-speed links to connect to other nodes. In addition to the Transputer, there was one control module on each section, and each of these modules was supposed to take care of local control. However, this made development complicated and unpractical. After a while it became clear that this system was outdated and it was decided to replace it with one central Motorola PowerPC 755 CPU fitted on a VME board. It is a low power processor running at 400 MHz. This processor is now responsible for running all on-board software. It runs a WindRiver VxWorks real-time operating system.

3.1.3 Perception

The IARES perception system consists of a pair of stereo cameras, mounted on a mast in front of the middle section (see figure 3.3). The cameras have a focal length

of 5 mm, 90 degrees field of view and 512x512 resolution. An effective depth of field of 8 m is obtained. The camera pair, also called stereo bench, is mounted on a platform with motorized azimuth and pitch angle control. This makes it possible to take panoramas.

3.2 IARES Software platform

The IARES software system is a collection of several modules/libraries, and makes in total a quite big system with its more than 300 000 lines of code. Among the most important modules are the rover module, the simulator module and the navigation module. In addition there are several smaller modules. There are for instance smaller programs for testing locomotion abilities and for manually plotting navigation paths.

All code is written in ANSI C. This language was chosen because of the availability of compilers on all platforms involved. Although a higher-level language, like for instance ADA, could have been more appropriate for minimizing error risks, the platform constraints did not allow this.

The information for this section has been found in [9, 7] and from personal communication with L. Rastel, as well as own experience with the software environment and the source code.

3.2.1 Software system principles

The IARES software system builds on the principle of UNIX instruments. The localization part is one instrument, the locomotion part is one instrument, and the perception part is one instrument. The rover control loop can then connect to these instruments. The man-machine interface (MMI) also communicates with the rover in this way.

These instruments can run on different machines in a network. Remote procedure call (RPC) functionality is obtained by using TCP/IP sockets, servers, and the External Data Representation (XDR) protocol. This makes it possible to run simulated parts along with the rest of the system with full transparency. The navigation module, for instance, is unaware about whether it is sending its commands to a real or a simulated rover.

An example configuration of the software system can be seen in figures 3.5 and 3.6. This demonstrates a typical setup where the rover is controlled from ground via the autonomous navigation workshop (see section 3.2.3). Another typical setup would be where the simulator (see section 3.2.2) substitutes the real rover. The use of instruments and RPCs make this change transparent to the ground software.

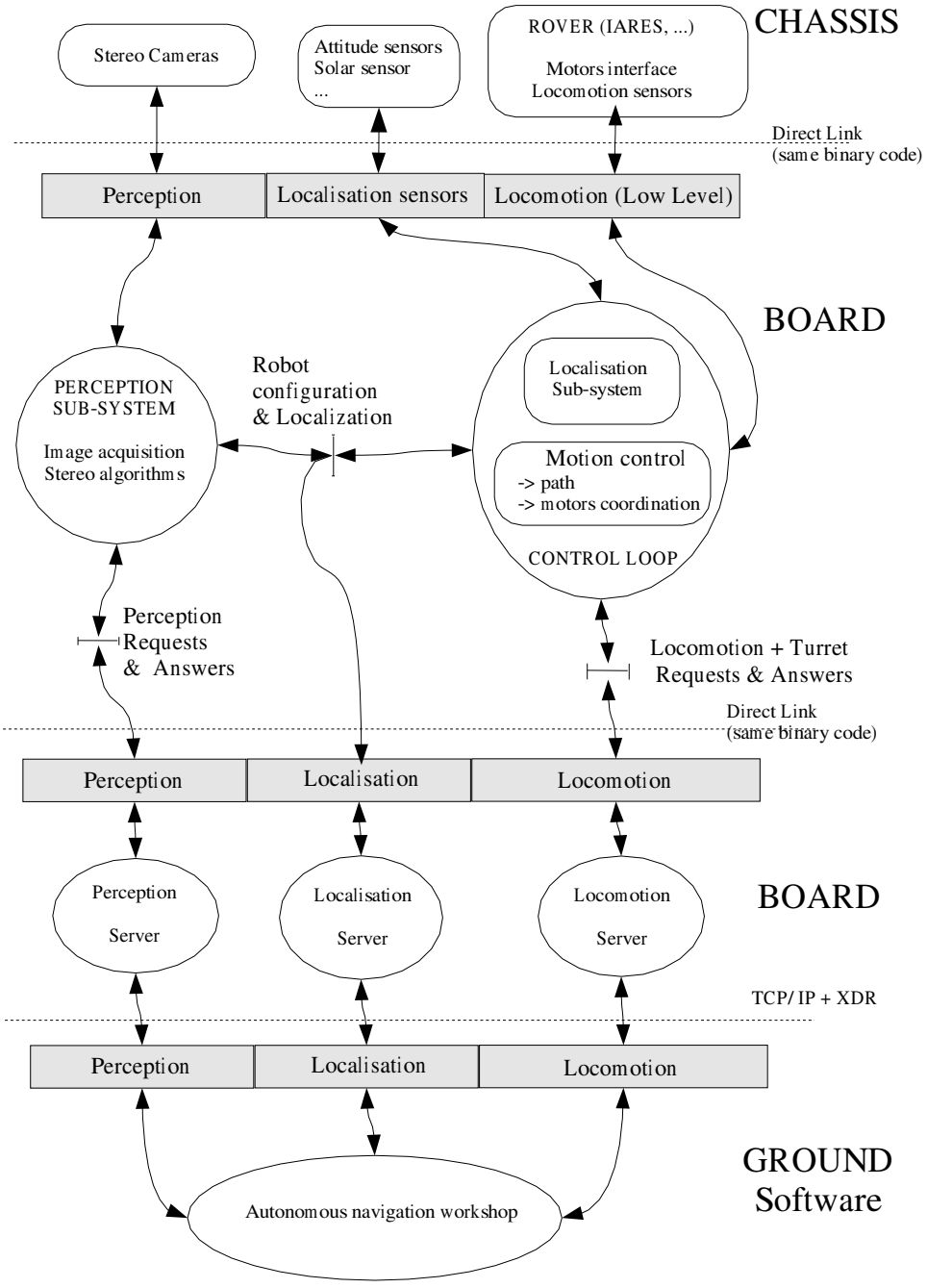

Figure 3.5: Example software system overview. The meanings of the symbols can be found in figure 3.6. Notice how the communication between the ground software and the rover on-board systems is made through the instrument interfaces. The control loop process is the active part of the rover system. It takes care of executing paths commanded by the ground software. Chassis sensors are interfaced and motors are commanded. Figure: CNES

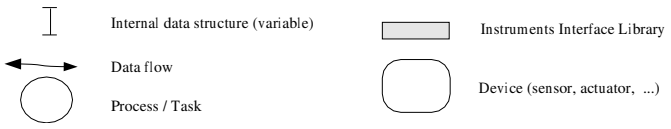

Figure 3.6: Symbols for software system overview (figure 3.5). Figure: CNES

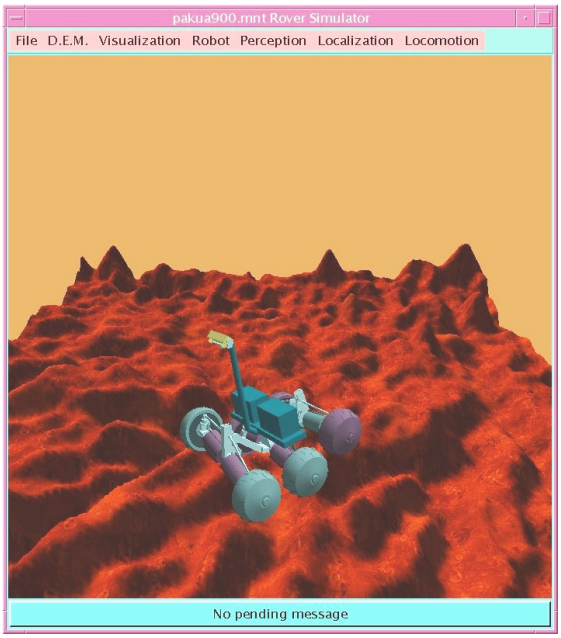

Figure 3.7: Rover simulator screenshot. This is the main simulator visualization display. The rover's movements and placement on the ground can be observed. Grey wheels or rollers indicate ground contact, while purple means absence of contact.

3.2.2 Rover simulator

The rover simulator (see figure 3.7) has been developed from scratch at CNES. Such a simulator has been found completely necessary for efficient software development. The simulator currently runs on Sun workstations with Solaris 9 OS. It also runs on a Linux platform. OpenGL is used for visualization. The simulator can use different digital terrain models (DTM) and also different rovers. Currently there is support for the IARES and Exomader rovers.

The simulator features a kinematic simulation model. The rover's placement on the ground is determined from a set of defined contact points on the wheels and the DTM. From the possible interactions between these, and by simulating the effect of gravity, a valid geometric configuration for the rover (height, tilt on each axle,

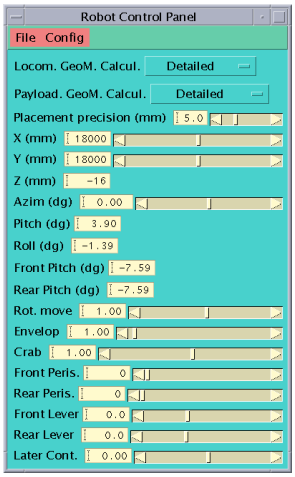

Figure 3.8: Rover control panel. This is one of several control panels available in the rover simulator. The rover position, as well as the heading (azimuth) can be controlled. There are also options for controlling the simulation accuracy, and for manipulating the rover parts.

and other parameters) is found.

From the graphical user interface (see figure 3.8), several parameters can be controlled. The rover can be placed at any position and with any heading. The viewer parameters can be adjusted, and also different technical parameters like refresh rate and simulation resolution. The graphical interface gives easily interpretable feedback when using navigation or locomotion test programs. There are also possibilities for some basic keyboard control of the rover in the simulator.

3.2.3 Autonomous navigation

The autonomous navigation subsystem is given a main objective, ie. a goal position or a direction, from the operator. Information about the surrounding terrain is obtained by stereo perception. From these informations, a path towards the goal is planned. Of this path, a short-term path is sent to the motion control subsystem. When this has been executed, the path planning is restarted.

Autonomous navigation process

An iteration of the autonomous navigation process can be summarized in the following way [9, 7]:

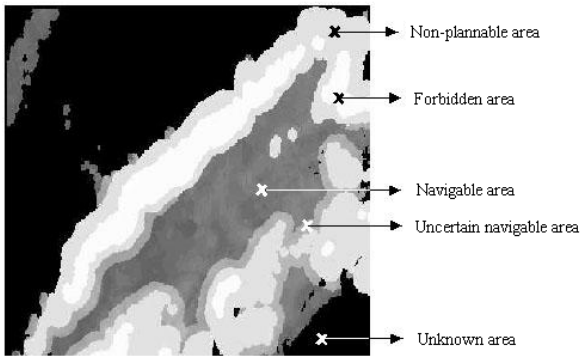

Figure 3.9: Example of a navigation map constructed from a DTM. According to the elevations in the terrain the different zones are classified for navigation planning. The navigable region is safe for path planning, while the other regions are more or less unsafe because of slope or discontinuity parameters. Figure: CNES

1. Construction of a local DTM from stereo perception
2. Translation of the DTM into a simplified local navigation map
3. Merging of the local map with previous data
4. Planning of a two-dimensional path over the navigation map
5. Planning of perception in order to make navigation continuous

Construction of a local DTM from stereo perception A correlation algorithm is run on the stereo pair of images taken by the cameras. Disparities between points in the images makes it possible to map points on to a grid in order to obtain a DTM. The cameras rotate and take several images in order to get a map of the rover's surroundings.

Translation of the DTM into a simplified local navigation map This step consists of analyzing the DTM in terms of navigability. Two factors are considered: slope and discontinuity. Discontinuity describes shorter obstacles like an isolated rock, while slope describes slopes in the terrain. This information is then coded into cells in a local navigation map. See figure 3.9.

Merging of the local map with previous data The local navigation map calculated in the previous step is now merged into the global navigation map obtained from previous perception.

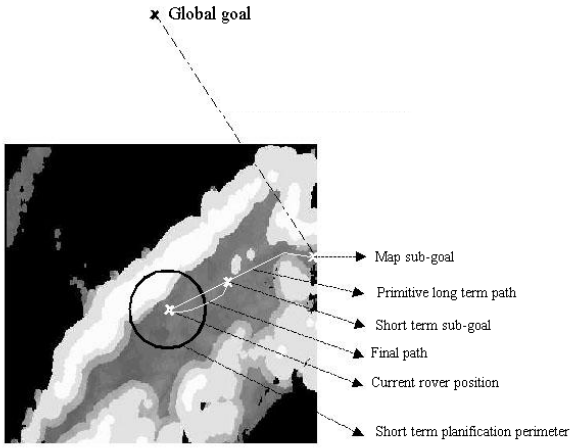

Figure 3.10: Navigation planning. The local sub-goal has been found on the edge of the local navigation map. The primitive long-term path can be seen, as well as the more refined short-term path. Figure: CNES

Planning of a two-dimensional path over the navigation map The planning method used is a so-called “continuous planning” method. The method does not guarantee an optimal solution, but it is often the case. Good results have been obtained in tests.

First, a local sub-goal is found, and a simple path solution from the rover to this point is calculated. See figure 3.10. The sub-goal is supposed to be a point on the edge of the local navigation map that approaches the rover to the global goal. The path to the sub-goal is not necessarily optimal.

Secondly, a short-term optimal path from the rover in the direction of the first path is planned, and this is the path that will be sent to the locomotion system. When the rover arrives at the end of this short-term path, the whole process is repeated, with perception of a new local map and planning of new paths.

Planning of perception in order to make navigation continuous This is a step that tries to perform the perception while the rover is moving. This avoids that the rover has to stop at the end of each short-term path for a panorama perception.

Resulting path for locomotion system

The short-term path that is sent to the locomotion system is in the form of a set of waypoints, which then connects to form line segments. See figure 3.11. Each segment has properties associated to it, like speed and security margin. This margin

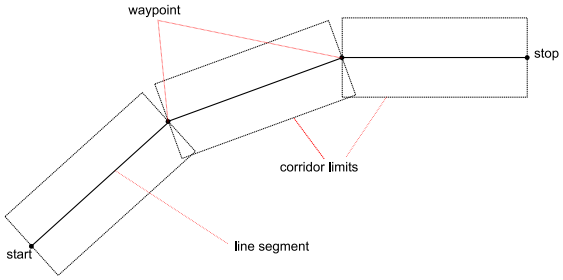

Figure 3.11: Example planned path with waypoints, line segments and associated margins (corridors). This path, however, has fewer waypoints than the paths planned by the path planning algorithm.

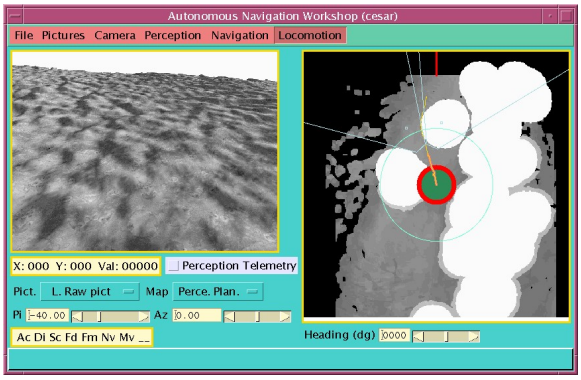

Figure 3.12: Screenshot from Autonomous Navigation Workshop

can be interpreted as a kind of corridor that the rover has to keep inside of.

Autonomous Navigation Workshop

The Autonomous Navigation Workshop is a program with a graphical user interface that lets the user control the autonomous navigation process. See figure 3.12. One has access to camera views, various image processing stage views and the resulting navigation map. It is possible to adjust various navigation parameters (like slope thresholds), to set goals, and to execute path planning. Different stages of the navigation process can be tested and executed manually, like for instance the building of a DTM model out of a stereo snapshot.

Chapter 4

The motion control system

This chapter studies the already existing motion control system. The goal for this is to know the system in order to be able to propose improvements.

The motion control system takes as input a path from the navigation system. Its task is to make the rover follow this path, as closely as possible. Inputs are also taken continuously from the rover localization subsystem. Outputs from the motion control system are wheel speed and steering angle commands, and possibly also commands to other chassis control parameters. The low-level motor control system reads these parameters and takes care of controlling the motors.

A view of the architecture of the system can be seen in figure 4.1. It is divided into three layers. The bottommost level takes care of low-level control of the motors on the rover chassis, and is beyond the scope of this subject. The middle layer coordinates the speeds and the steering of the six wheels (and possibly other elements), in order to make one coordinated movement, for example a turn in place movement. The topmost layer takes care of controlling the coordinated movements, in such a way that a path can be executed.

The current locomotion control system was originally developed for the EVE vehicle (see section 2.2.1), ie. a vehicle with no individual steering capability for the wheels. Later the system has been extended to also use steering of wheels on front and rear axles, in order to benefit from some of IARES' extended locomotion capabilities. This also makes the system convenient for controlling the Exomader rover (see section 2.2.2).

The information for this chapter has been found in [11] and [5], as well as some personal communication with L. Rastel, but also very much from reading and interpreting the source code.

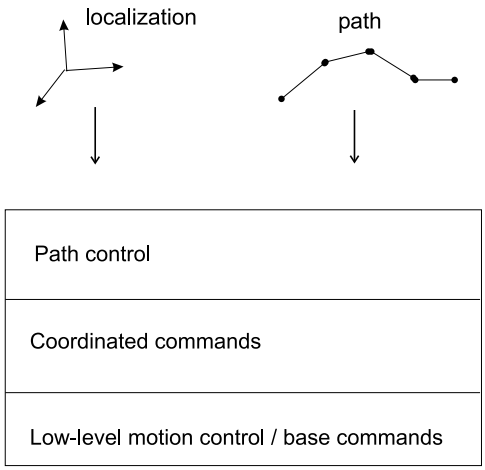

Figure 4.1: The motion control layered architecture. A path and localization data are available to the path control layer.

4.1 Locomotion configurations

This section presents the currently known locomotion configurations. Not all of the configurations are used in the motion control algorithms, but they are at least testable in teleoperated mode.

4.1.1 Turning

Every type of turning, at a given instance, can be seen as a movement of rotation about a point. This point is called the instantaneous center of rotation (C), and the distance from this point to the rover is the turning radius (r). An alternative way of expressing the turning amount is by curvature ($c = 1/r$). Both types of turning presented here have C on a line extending the center axle of the rover.

Apart from going straight forward or backward, turning is the most usual movement for the rover. And straight motion can actually be seen as a special case of turning, where the turning radius is infinitely large.

Turning by difference of wheel speeds

The simplest form for turning is turning by difference of wheel speeds, the same way as a tank turns. This is feasible for vehicles having no steering on wheels, and tracked vehicles. The method works by turning the wheels on one side of the rover (furthest away from the center of rotation) faster than the wheels on the other side.

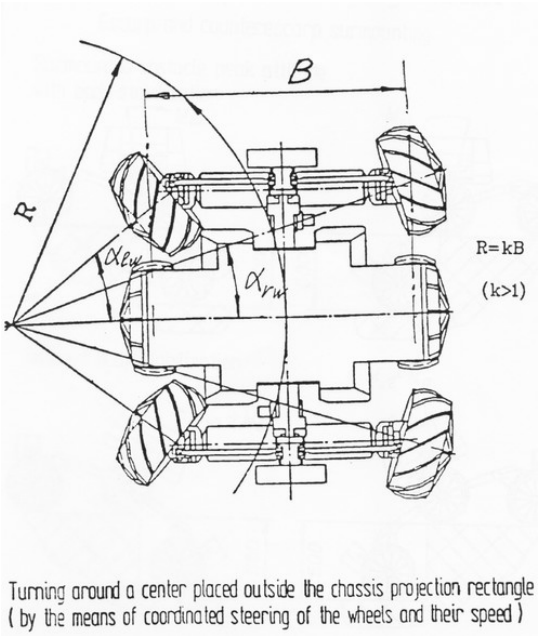

Figure 4.2: Turning around a point by wheel steering. All wheel axes meet in the instantaneous center of rotation. Figure: VNII Transmash

This mode works for vehicles with any number of wheels, however, the more wheels there are, the more skidding there will be. Actually, a theoretical rover with only two wheels would be able to perform this type of turning without skidding. Skidding arises since not all wheels are oriented around the same center of rotation. The actual movement of each wheel is not the same as it would have been if the wheel was not connected to the chassis.

This method is implemented in the motion control system.

Turning by wheel steering

By steering the wheels on the rover in such a way that the axes of all wheels meet in the instantaneous center of rotation, skidding is avoided. The wheel speeds will also have to be adjusted according to their distance from the center. See figure 4.2. The minimum turning radius (ie. the maximum turning curvature), R_m, is limited by the maximum steering angle on the innermost wheels.

This way of turning makes sense on four-wheeled and six-wheeled rovers, but can also be applied to rovers with more wheels. However, four-wheeled rovers can accomplish this turning by having steering only on two wheels, and six-wheeled rovers need four steerable wheels. An eight-wheeled rover would need six steerable

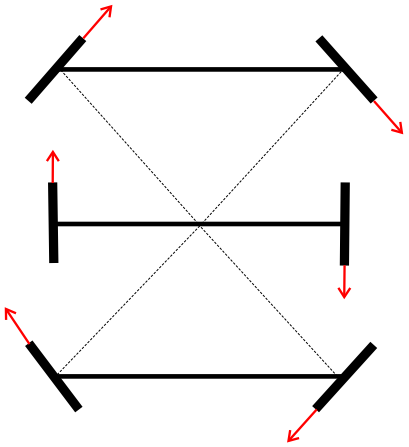

Figure 4.3: Rotation in place by envelope configuration. The wheel speeds are indicated in with red vectors. Notice that the axes of all wheels meet in the center of the rover.

wheels. This is based on the principle that the rotation center C lies on an extension of one of the vehicle axles. Otherwise, all wheels would have to be steerable. When this principle applied to four-wheeled vehicles, it is called Ackerman steering [13], and is the same principle as used on normal cars. Here the center of rotation lies on a line extending the car's rear axle.

This method is implemented in the motion control system. However, it does not work correctly. This will be discussed in chapter 5.

4.1.2 Rotation in place

Rotation in place is a practical movement for reorienting the rover. It can be seen as a special case of turning, where the center of rotation is at the center of the rover.

Rotation by difference of wheel speeds

Rotation in place can be accomplished by using opposite wheel speeds on opposite sides of the rover. However, as with turning using difference of wheel speeds, this movement causes skidding, as long as the vehicle has more than two wheels.

This method is implemented in the motion control system.

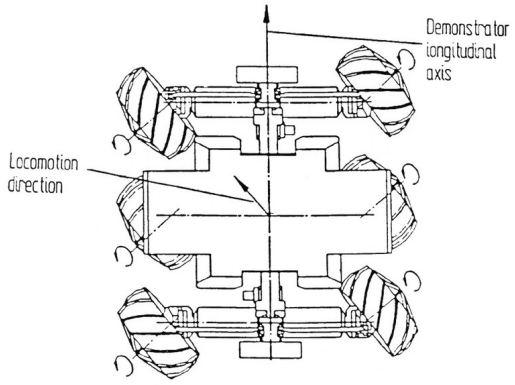

Figure 4.4: The crab configuration. All wheels have the same steering and speed, and the locomotion direction makes an angle with the rover's longitudinal axis. Figure: VNII Transmash

Rotation by envelope configuration

By going into a special wheel steering configuration with front and back wheels, the rover can turn in place very efficiently. See figure 4.3. The wheels are all oriented so that their axes are meeting in the center of the rover. The left and right wheels are rolling in opposite directions, and the middle wheels have different speed than front and back wheels. There is no skidding when turning this way.

This method is implemented in the motion control system.

4.1.3 Crab configuration

The rover goes into the so-called crab configuration by simultaneously commanding the same steering to all 6 wheels. This permits the rover to perform a sideways diagonal translational movement without changing its heading. See figure 4.4.

4.1.4 Wheel-walking mode

The IARES rover has the ability to perform so-called wheel-walking, or peristaltic, movement. See figure 4.5. By only moving one wheel at a time, the rover can advance in difficult terrain with little friction, such as very sandy areas. This locomotion mode also helps to avoid sliding down steep slopes, and very much explains IARES' exceptional slope climbing capabilities.

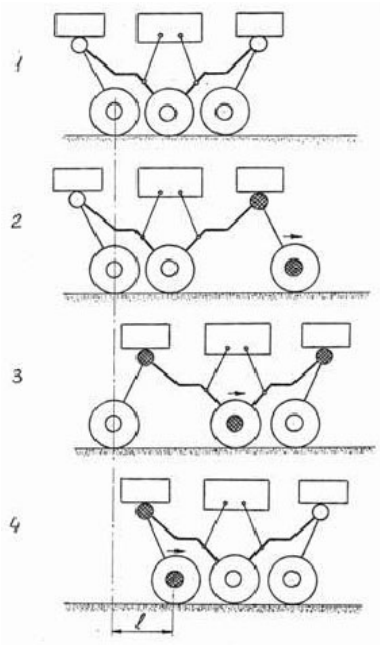

Figure 4.5: Wheel walking. The dark points indicate moving joints. The rover begins in a contracted state in step 1. Then, in step 2, the front section advances together with the front wheel, rolling. After this, the front and rear wheels stay still while the middle section advances in step 3. Finally, in step 4, the rear section and wheel advances like the first did in step 2. The process then restarts. Figure: VNII Transmash

4.2 Motion control algorithm

This section describes the motion control algorithm currently implemented in the software system. This algorithm can be divided into several different behaviours, and these will be explained in detail.

When the control algorithm was designed, it had the following requirements [5]:

- To keep the rover as close as possible to the planned path, while at the same time trying to avoid stopping for recenter operations.
- To keep the vehicle sufficiently stable during movement on slopes.

The first requirement is satisfied by the following three main behaviours:

- General line following control system
- Recenter inside corridor if an exit is inevitable (preventive mode)
- Return to corridor if the rover has exited (corrective mode)

The second requirement is satisfied by slope compensation actions, interwoven in the general line following system.

4.2.1 General line following control system

The goal of this control process is to keep the rover as close to the line segment as possible while advancing to the next waypoint.

The steering can be accomplished in two ways, either by using difference of wheel speeds, or by steering the angles of the back and front pairs of wheels (see section 4.1.1). The advantage is that with both of these methods, the steering can be seen as a turning movement with an instantaneous center of rotation, and the amount of steering can thus be described with a radius (r) or a curvature (c).

The amount of steering to be commanded to the rover is found from the following deviation parameters:

- The distance of the rover from the line segment (d)
- The angular deviation between the rover's heading and the direction to the waypoint (θ)
- The angular deviation between the rover's heading and the direction vector of the line segment (u)

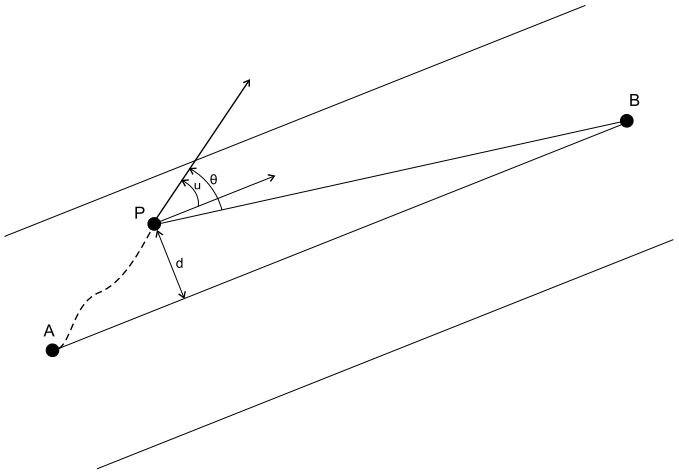

Figure 4.6: Rover following line segment. The rover, going from A to B, is situated at point P. The different deviations considered for the line following algorithm are marked.

These parameters can be seen in figure 4.6. The steering amount commanded from the line control process is expressed in curvature (c). Mathematically, this can be expressed in the following way:

$$c = f(d, u, \theta) \tag{4.1}$$

f is a regulation algorithm. Basically, the curvature to be used for steering is found by adding the deviation components, multiplied by gains (coefficients). The gain for u is static, while the other gains depend on the rover's speed, and the position gain also takes the remaining distance to the goal into account. This can be expressed as

$$f(d, u, \theta) = k_{p,d} \times d + k_{p,u} \times u + k_{p,\theta} \times \theta \tag{4.2}$$

Where the k_p's are the gains.

This algorithm makes the rover arrive at its target, but it does not guarantee that the rover stays inside the corridor [5]. Therefore, the line control is complemented by a check to see if the robot has exited or is about to exit the corridor, and a possible correction for getting back on track.

4.2.2 Corridor exit test

At all times there is a test to see if the rover has exited or is about to exit the corridor. The exit test functions as a test for both the preventive and the corrective mode. The preventive test detects whether the rover is about to exit in the near

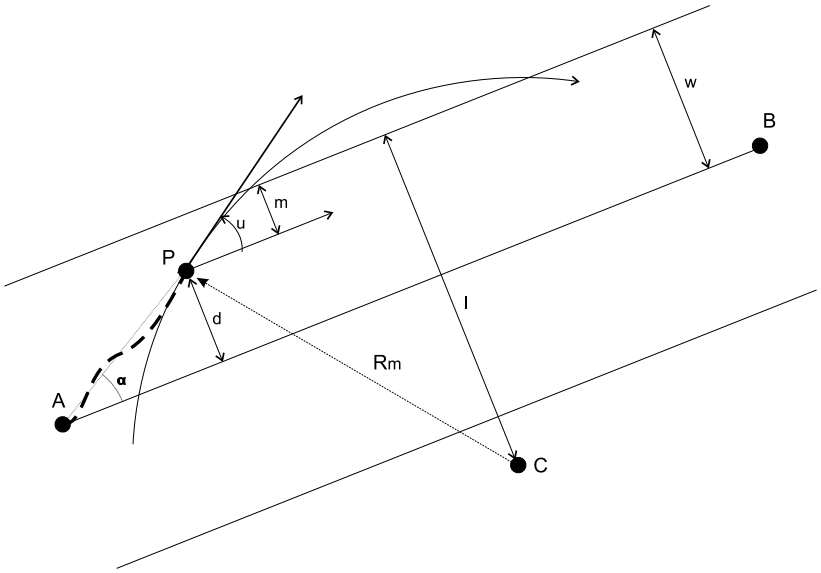

Figure 4.7: Corridor exit test. The rover, going from A to B, is situated at point P. Its smallest possible turning radius is R_m, which will, in this case, not be sufficient to stay inside the corridor.

future, in order to allow for an avoidance of this by appropriate correcting actions. However, should the rover despite of this wander outside the limits of the corridor (sudden terrain change, obstacle), this is also detected.

For the corrective test, the robot's distance d to the line segment is calculated.

$$\frac{d}{\|\mathbf{AP}\|} = \sin\alpha \tag{4.3}$$

$$\frac{d}{\|\mathbf{AP}\|} = \frac{\|\mathbf{AP}\times\mathbf{AB}\|}{\|\mathbf{AP}\|\|\mathbf{AB}\|} \tag{4.4}$$

$$d = \frac{\|\mathbf{AP}\times\mathbf{AB}\|}{\|\mathbf{AB}\|} \tag{4.5}$$

See figure 4.7.

If $|d| > w$, the robot is outside the corridor.

For the preventive test, the following assumption is made: The rover travels at a constant speed, where the best possibility for steering away from the corridor edge is by a circular movement. See figure 4.7. The maximal turning amount (or the minimal circle radius) of this movement is limited by mechanical properties of the rover. For instance will turning by wheel speed difference and turning by wheel steering have different minimum radii.

The heading and the position of the robot, supplied by the localization system,

allows finding the rotation center C_r (at distance R_m) for the sharpest turn possible.

If the distance l from the rotation center to the edge of the corridor is shorter than R_m, an exit is imminent. That is, the rover will not be able to keep inside the corridor by using the steering commanded by the general control algorithm. The exit condition can be expressed as follows:

$$R_m > l \tag{4.6}$$
$$R_m > R_m \cos(u) + m \tag{4.7}$$
$$R_m(1 - \cos(u)) > m \tag{4.8}$$

where m is the difference between w and d, and u is difference between the rover heading and the line segment heading. This relation avoids passing by the calculation of the position of C_r.

4.2.3 Recenter inside corridor and return to corridor

When it is clear that the robot is outside or is about to exit the corridor, it is stopped before performing a rotation in place. This is done to ensure an as quick return to the line segment as possible. The goal angle for this rotation is determined by the position of the "return point" on the line segment. The return point is decided by first projecting the rover's position down on the line segment and then adding a certain return margin.

After having turned in place, the robot moves in a straight line back to the return point. A new rotation in place, aligning the rover with the line segment direction, has to be made before the rover can continue with the normal control algorithm.

4.2.4 Transition to a new segment

When the rover arrives at the end of a line segment, the robot brakes and stops completely. There it can perform a stationary rotation for aligning to the next segment, depending on the waypoint properties. Then the new line segment parameters are taken into account, and the rover continues with the general control algorithm.

4.2.5 Slope compensation

When travelling on sideways slopes, it is possible that the rover will loose its grip and slide sidewards to some extent. This is solved by slope compensation in the turning movements. The localization subsystem gives the control algorithm the rover's current inclination value. This is then used for adjusting the curvature commanded by the line following control system. When turning down/with the slope,

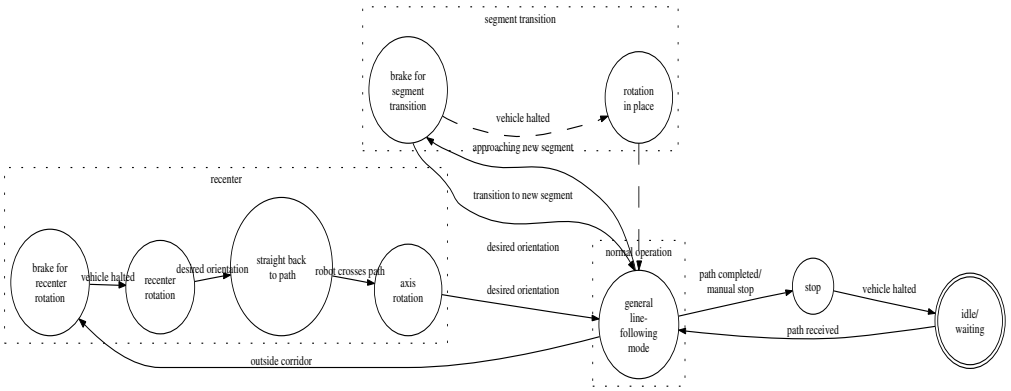

Figure 4.8: Rover control states. The dashed lines represent an alternative state transition. The dotted boxes represent general phases of the motion control behaviour.

the curvature amount is slightly reduced, according to the inclination amount of the rover. When turning up/against the slope, the curvature amount is increased. In addition to this, speed is decreased when it is detected that the rover moves on a steep slope.

4.2.6 Implementation

Architecture-wise, The motion control algorithm and the general line following system control belong in the path control layer (see figure 4.1). The calculation of wheel angles and speeds, according to different locomotion configurations, belongs to the coordinated commands layer.

The control algorithm can be seen as a state machine, illustrated in figure 4.8. Starting in the idle state, the robot waits for a planned path consisting of several path segments. When a path is received, the robot goes into the normal control mode where the general line following system is applied. When the exit corridor check succeeds, the robot enters a recenter phase. This starts with a brake state, then recenter rotation, then a back to path state. The final part of the recenter phase is the realignment rotation phase which is triggered once the robot reaches the line segment.

There is also a segment transition phase at the end of each line segment, which consists of a braking state and then a possible rotation state for aligning with the next segment. When the rover has stopped completely, parameters for the new segment are taken into account. Then, depending on the parameters of the waypoint, the rover either goes into a rotation state, or it passes directly on with the general line control algorithm.

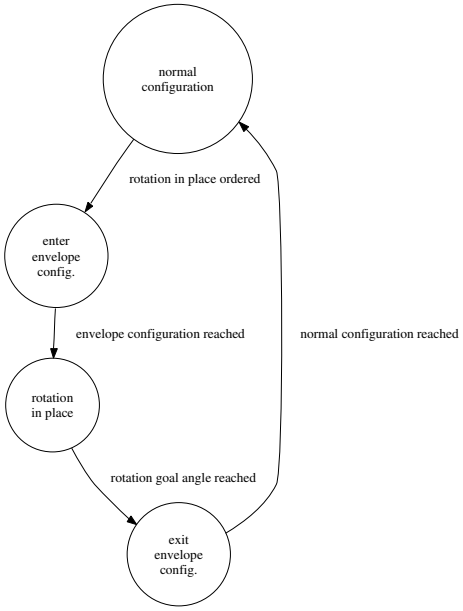

Figure 4.9: Rotation in place states using envelope mode.

The rotation in place can be accomplished in two different ways. Either by using the rotation by wheel speed difference method, or by using the envelope method (see section 4.1.2). If the envelope mode is used, the rotation state is divided into three states: enter envelope configuration (positioning of wheels), envelope rotation (the actual turning) and exit envelope configuration (positioning of wheels back to normal state). See figure 4.9.

Chapter 5

Analysis and design choices

This chapter analyses the functionality and performance of the autonomous motion control system studied in the previous chapter. Based on this analysis, propositions were made about changes to the system, including extended functionality and performance improvements.

The analyses performed in this chapter are mostly of a qualitative nature. Quantitative tests will be performed in chapter 8, when the current state of the system can be compared with the new developments.

5.1 IARES platform development and testing conditions

At the moment, the IARES rover is still mounted on an indoor test rack. There are plans to get it out on the outdoor test site, but it is still uncertain how long it will take before this can be accomplished. The actuator level software will have to be developed, and a localization module, as well as a general control loop, will have to be integrated. This is being worked on by other people in parallel with this locomotion development.

In the meantime, the simulator will have to be used. This offers a kinematic simulation mode, so physics features like friction and skidding are not available. Such factors will therefore be impossible to test before the rover is ready for the test site. However, the simulator is very practical for development as it provides a clear environment with good feedback and overview. Real-life noise on for instance localization input can be simulated, but it is practical to remove this at least for early development stages.

All analysis of the current locomotion system performed in this chapter is based on rover behaviour in the simulator. It cannot be excluded that real life performance would be significantly different, however it is unlikely to believe that something

which does not work in the simulator would work in real life. Making things work in the simulator first will therefore be necessary in any case.

5.2 Functional analysis of locomotion capabilities

There IARES rover is a very flexible vehicle, giving several possibilities for carrying out the locomotion task. The features can be tried out from the teleoperated mode. However, at the moment, for the autonomous motion control mode only parts of them are utilized.

5.2.1 Turning by difference of wheel speeds

The most basic mode of turning, the so-called difference of speeds method, is fully functional. This mode was already known from the previous rover model, EVE.

5.2.2 Turning by four-wheel steering

Turning by four-wheel steering was discovered to have been implemented in the motion control system as well, contrary to what was known. The implementation, however, was in an untested state, and it had errors. While turning to the right, the wheels are steered to wrong angles. Inner and outer angles were swapped.

Turning by wheel steering is in general preferred over the difference of speeds method for at least two reasons. Firstly, since skidding is avoided, one has better control over how the rover has travelled. The method of measuring the rover's position by checking how much the wheels have turned, is called odometry. Skidding introduces errors to this method and it is therefore more difficult to estimate the rover's position. Secondly, by eliminating skidding, the movement is also more energy efficient. Energy saving is, as mentioned in chapter 2, a primary concern for Mars rovers.

This turn mode uses the front and back wheels for steering, and will thus also be compatible with other six-wheeled rovers having steering capabilities only on front and back wheels.

5.2.3 Six-wheel steering

The crab mode, or in general, IARES' capacity of steering all six wheels, is not utilized. At the moment, crab steering is allowed only from telecommanded mode.

Crab steering, although not used by any mission rovers yet, could be useful for

motion control. From the control algorithm point of view, it would permit to correct independently the position and heading deviations. The diagonal movement would allow for a correction in the rover's position without needing a turn. Turning could then be used only for correcting heading deviations. The problem with controlling the rover using only a turning mode (as with the four-wheel steering turn around a point mode) is that an attempt to correct a position deviation necessarily also leads to a heading deviation, and vice versa.

The crab configuration could also be used for closing up on objects sideways, or maneuvers in very tight conditions. Ideally, wheels with omnidirectional steering (steering up to 90°) would allow for a pure sideways movement, just like for a real crab's movement.

The downside about including six-wheel steering on a mission rover, is the extra cost and complexity connected to this. Two more motors for steering will have to be added, this adds extra weight and commplexity to the system.

5.2.4 Advanced locomotion

The peristaltic movement ability (see section 4.1.4), or more in general, deformation of the rover chassis, is not used. This is related to the fact that it is a complicated maneuver and difficult to plan automatically.

An application of the peristaltic mode could be "walking" in steep slopes and areas with low friction, such as loose sand. However, the simulator does not provide functionality for testing such factors yet. Testing and parts of development would have to be carried out with the rover on the outdoor test field.

Another application for the chassis deformation possibilities would be complex obstacle negotiation. This is a very complex task and would at least require some extended sensoring possibilities.

In addition, the IARES has inclination abilities which are not used at the moment. A typical application for this would be for stabilizing the rover while it is going up or down a slope, and also for sideways stabilization. A horizontal payload increases the wheel grip and facilitates perception, since the mast also moves accordingly to the payload part.

5.3 Performance analysis

This part discusses the performance of the already existing locomotion systems. A few smaller problems have been discovered, and will be discussed. The line following control system will also be evaluated.

5.3.1 General line following control

The general line following control part seems to do its work, as the rover manages to keep on track. It is however possible that this work could have been done in a better way. Position deviation seems to be slowly corrected.

Wheel steering angle update is not instantaneous. The motors steering those angles are limiting the update speed. The line following control algorithm does not take this into account. In particular, at transitions between segments there are discontinuities in the commands. Such abrupt changes in the commands will not be possible to follow by the motors.

The slope compensation part has not been analyzed. However, the source code indicates that it could need some investigation. There is slope compensation in the general control algorithm function and then an extra slope compensation in another place, because the first compensation is believed to be insufficient. This seems illogical and unstructured.

5.3.2 Segment transition problem

When making a transition from one segment to another, the rover seems to rotate in place two times. First aligning to the new segment direction, then another time, not changing azimuth. This became apparent when trying out the turn around a point steering mode, as there is a phase of preparing the wheels before and after rotating in place. This clearly seems very inefficient.

By analyzing the rover's behaviour more closely, it becomes clear that this happens only when the two segments are at an angle to each other. From debug output it can be read that the rover thinks it is about to exit the corridor after having passed on to the new point. This is true - however, the rover then passes on to the recenter phase (See figure 4.8). An action which seems inappropriate, since the rover, after passing on to the next segment, already is in the center of the corridor. The recenter phase requires two rotations in place, while it would be sufficient with only one rotation for aligning with the new segment.

Rotation in place can be forced though, by setting the control waypoint as a "stop point" instead of a "waypoint". However, there is no automation in controlling which points should be what.

The system also has a bug in the rotation in place state transition. When switching to rotation in place in the segment transition phase, the internal state goes directly from normal mode to rotation in place (see figure 4.8). This causes the rover to jump over the enter envelope configuration phase when in wheel steering mode. The result is that the wheels start to rotate while they are being steered towards the envelope configuration. This causes skidding and is clearly not wanted.

5.3.3 Braking between segments

The current locomotion algorithm always brakes the rover to a halt when reaching a new line segment, both for waypoints and stop points. This makes the movement somewhat choppy and slow, especially if the path is composed of many short line segments. This is often the case with the path planned from the navigation algorithm. In addition, these segments often have a small angle against each other. The navigation algorithm tries to fulfill the criterion of making as continuous paths as possible, avoiding turning in place if possible. This indicates that it could be possible to avoid braking between each waypoint.

Stopping between each line segment is in contradiction with the navigation algorithm's wish of keeping continuity in the rover movement. There is no need with continuous perception (see section 3.2.3) if the rover stops many times between each planning step anyway. Mission-wise it is an overall goal to minimize energy consumption, and braking and acceleration does not help this. Frequent stopping also reduces the rover's daily traverse, and since the rover has a limited lifetime, this should be avoided if possible.

5.4 Discussion

At the moment, the rover seems to perform OK with the current control systems. Although the general line control system is not optimal, it keeps the rover on the line. The segment transition and braking problems are, although minor, problems that make the path execution inefficient.

Depending on the turning modes, the rover has more or less difficulties with keeping inside the corridor. Turning by wheel steering performs better than the difference of speeds mode, even if it is not correctly implemented.

There are several possibilities for improving the current motion control system. Some of them are listed below:

- Fine-tune and improve current general line following control algorithm in order to make it perform better, with for instance better position correction, or with motor speed compensation.
- Develop a new type of line-following control algorithm that works on other principles than the regulation approach. This could for instance be a geometrical/mathematical or a fuzzy logic approach.
- Extend the rover's functionality by adding six-wheel (crab) steering. This could be integrated in the current control system.

- Develop approaches for improving the general performance of the rover, notably by fixing the problems with the line segment transitions. Fix bugs like the rotation in place bug and implement a correct wheel steering mode.
- Redesign the motion control system by taking new approaches to the different requirements.

5.5 Choices

In the beginning of the project it was not known that the wheel steering mode was already partially implemented, and there was little knowledge about how the motion control algorithms worked in general. This is partly because of staff reductions at the lab and unfinished and undocumented projects. While the expectations for this project may have been oriented towards the control algorithm and implementation of the wheel steering mode, this should now be reconsidered.

Since the current motion control system seems to already work to a certain degree, it is more interesting to try to to extend the current functionality. I have chosen to investigate a way to include crab steering, or six-wheel steering in general, into the motion control algorithm. As there are already a certain number of corridor exits during the path following process, it is of interest to come up with a more powerful and flexible way of correcting the rover's motion.

Little literature has been found on six-wheel steering, and motion control in general. Most rover-related publications concentrate on path-planning, localization, or mechanical properties. This could very well be because these subjects are greater challenges and thus more suitable for research and publication. In addition, few rovers with six steerable wheels exist. JPL has six steerable wheel prototypes, but no publications of interest have been found.

Since such a locomotion mode has not been explored very much, this is a interesting area of investigation. It will be of interest to see if this extension can improve the current motion control system. The results of this could then be used as a base for considering whether it is worthwhile with steering capabilities for the middle axle wheels.

In addition to this, since I have little knowledge of regulation techniques, an attempt to try to improve the current line-following control algorithm seems rather pointless. A new approach for the control algorithm could have been attempted, but this would require knowledge about and analysis of both methods. As for a new design of the whole system, this could be a challenging task, but probably a bit too much. There is also at the moment little need for this, as the current system seems to perform in an acceptable way except for the minor problems mentioned.

I have also decided to find solutions to some of the small problems discovered. This

is anyway essential in order to appreciate the results of the functional extensions, and can thus be seen as a preliminary stage to the crab extension part. For instance will it be difficult to compare turning by wheel steering and crab-extended mode if the wheel steering does not work correctly. The possible gains by these smaller improvements seem to justify the amount of effort that needs to be put into the task.

Chapter 6

General improvements

This chapter considers general improvements made to the motion control system. These are smaller improvements that were useful to get in place before starting with the functional capability extensions.

6.1 Turn around a point by 4-wheel steering

This configuration has already been implemented in the motion control system. However, it did not work as expected, and had probably not been tested. The functional behaviour was not satisfying, and the code was unclear, making it difficult to get an overview and point out errors. It was decided to rewrite this part.

6.1.1 Inverse kinematic model

A direct kinematic model goes from wheel speeds and angles and gives global values like speed and rotation of the vehicle. The inverse kinematic model goes from global values and gives local values. To calculate the rover's wheel angles and speeds from the curvature command (c) and rover global speed, an inverse kinematic model is needed.

Wheel angles

The wheel angles that have to be calculated for the turn around a point configuration are $\alpha_1, \alpha_2, \alpha_5$ and α_6. See figure 6.1. The middle axle wheel angles (α_3, α_4) are zero. For this calculation, the following variables are involved:

- The turning radius r, derived from the curvature commanded ($r = \frac{1}{c}$). The

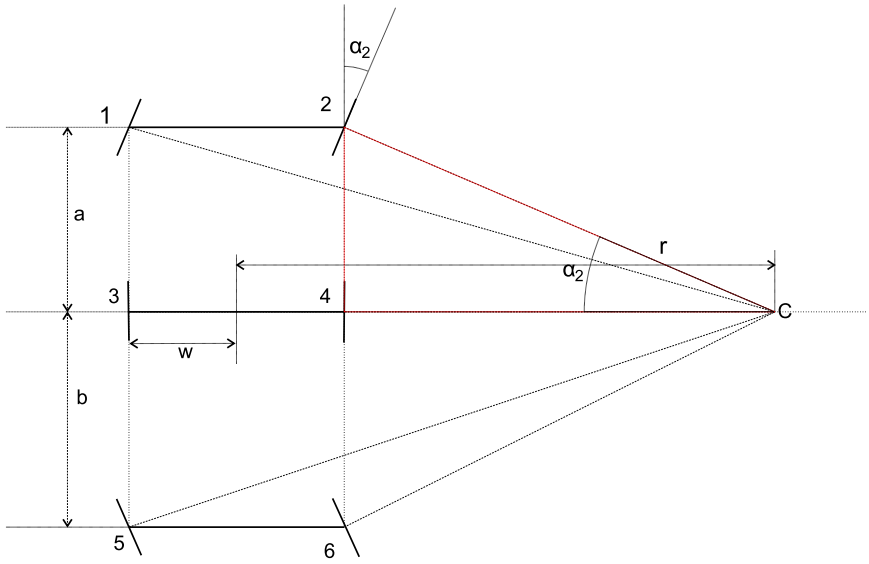

Figure 6.1: Angle calculation for turning around a point. The triangle used for calculation of α_2 is marked in red. On the figure, r and α_2 are negative, while the other sizes are positive.

radius is counted from the center of the vehicle.

- The distance from the center to the middle left/right wheel of the vehicle, w.
- The distance from the middle axle to the front axle, a.
- The distance from the middle axle to the rear axle, b.

The relations are as follows:

$$\alpha_1 = \arctan \frac{a}{r - w} \qquad \alpha_2 = \arctan \frac{a}{r + w} \tag{6.1}$$

$$\alpha_5 = \arctan \frac{-b}{r - w} \qquad \alpha_6 = \arctan \frac{-b}{r + w} \tag{6.2}$$

Wheel speeds

The wheel speeds for the different wheels are calculated from the commanded vehicle speed v from the line following algorithm. This is a linear translation speed (not angular wheel rotation speed) an can be seen as a speed vector from the center of the vehicle. See figure 6.2.

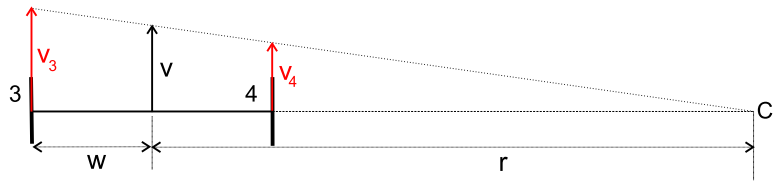

Figure 6.2: Calculation of wheel speeds for middle axle wheels. v_3 and v_4 are calculated from v by means of proportionality.

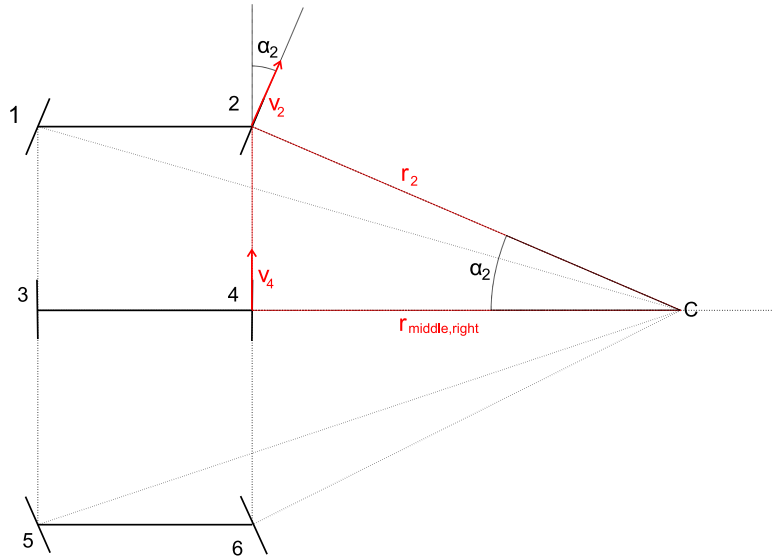

Figure 6.3: Calculation of wheel speeds for front and rear wheels. The triangle used for calculating v_2 is shown in red.

Starting with v, in the center of the vehicle, one can find the speeds for the middle axle wheels v_3 and v_4. These are proportional to v, according to the turning radius r. Thus, one has the following relations, where w is the distance from the center of the rover to the side:

$$v_3 = v \times \frac{r - w}{r} \qquad v_4 = v \times \frac{r + w}{r} \tag{6.3}$$

The speeds for the front and rear wheels can then be derived. Another proportionality relation can be used, this time between the radiuses of the actual wheel, r_i, and the middle wheel, $r_{middle,side}$. These two radii can be seen as one side and the hypotenuse of a right triangle, respectively. See figure 6.3. We therefore have:

$$v_i \quad = \quad \frac{r_i}{r_{middle}} \times v_{middle} \tag{6.4}$$

$$\frac{r_{middle}}{r_i} = \cos\alpha_i \tag{6.5}$$

This gives the following relations:

$$v_1 = \frac{v_3}{\cos\alpha_1} \qquad v_2 = \frac{v_4}{\cos\alpha_2} \tag{6.6}$$

$$v_5 = \frac{v_3}{\cos\alpha_5} \qquad v_6 = \frac{v_4}{\cos\alpha_6} \tag{6.7}$$

6.2 Segment transition rotation problem

Since the double rotation in place at segment transitions seems unnecessary, it was decided to optimize this behaviour. The option about stop point types, as explained in section 5.3.2, has been left aside. Since the point type is controlled by the navigation algorithm, and apparently not much used, it has not been changed. Manipulating the navigation algorithm could be risky because of lack of knowledge about all factors involved.

Instead, a change was introduced at the motion control level. Simply, when transitioning to a new segment, a check is performed. This checks the size of the angle between the two line segments. If this angle exceeds a certain threshold, the rover does one simple rotation in place for reorientation. Otherwise it just continues with the general line following control algorithm.

This threshold angle can be adjusted according to which steering mode is employed. A too small threshold angle will make the rover do the rotation too often, even if it could just have continued, and managed to follow the path with the general line following algorithm. If the threshold angle is too big, the rover will not reorientate when it is necessary. This means that its heading deviates too much from the direction of the new line segment, and this causes the rover to sooner or later detect a possible corridor exit. The whole “get back on track” recenter operation will then have to be performed, which includes two rotations in place.

In addition, the rotation in place state transition bug was fixed. When going into a recenter rotation phase at the beginning of a new line segment, the braking phase is now the first phase entered. After this phase a transition to the enter envelope configuration is done if the 4-wheel steering mode is activated.

6.3 Implementation

All implementation has been done by modifying existing functions. The modifications for the 4-wheel steering involved a remake of the existing code, reducing the

number of different cases. The inverse kinematic model is implemented in the coordinated commands layer. The segment transition improvements implementations were straightforward, and implemented in the path control layer.

All code has been written in ANSI C. Makefiles were used as a compilation aid. Emacs was used for code editing.

Chapter 7

Six-wheel steering

Previously the motion control algorithm did not use steering of wheels on the middle axle. As the IARES has the capability to steer all wheels, this feature can be exploited. Intuitively the addition of steering of the middle axle wheels can be seen as adding crab steering to the already existing four-wheel turn around a point steering.

This could offer an easier and more effective way of motion control in the line-following algorithm: As the crab steering allows for translation without rotation, this steering can be used to correct the positional deviation from the path. The heading deviation of the rover can be corrected by the turn around a point steering mode. The current system uses the turning to correct for both types of deviation.

In this chapter, two different methods of doing six-wheel steering are proposed. Development of the first method gave ideas to a second method that used a different approach. Both have been included, since both have their advantages and disadvantages.

7.1 First method proposed

7.1.1 Principle

This method consists of superposing steering values from the turn around a point configuration (see section 4.1.1) with crab (see section 4.1.3) steering values. See figure 7.1. Intuitively this allows for a turning movement at the same time as a diagonal translational movement. The turning part of the movement takes care of correcting the azimuth (heading) deviation (u,θ) of the rover, while the crab part corrects the position deviation (d). If the turn amount is expressed as t, and the

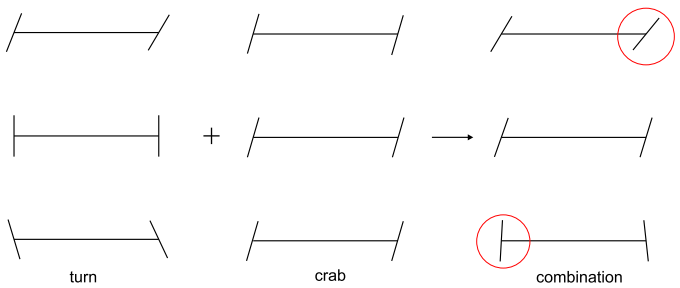

Figure 7.1: Turn and crab combination. The turning and crab angles are added to make a combined movement. Notice how the front right wheel angle gets a big amplitude while the rear wheel angles are almost zeroed out.

crab amount as c, we have, in the line following algorithm:

$$t = f(u, \theta) \tag{7.1}$$
$$c = g(d) \tag{7.2}$$

7.1.2 Inverse kinematic model for wheel angles

The inverse kinematic model for this mode is basically a superposition of the models for the turn around a point mode and the crab mode. The turn around a point model can be found in section 6.1. The crab mode is straightforward, as it is sufficient to set all wheels to the same angle and speed. The crab angle is controlled directly from the line-following algorithm.

The individual wheel steering angles ($\alpha_{i,t}$) are found from the curvature commanded by the motion control algorithm, like in the normal turn around a point mode. These angles are then added to the crab angle (α_c) commanded. This gives, for the inverse kinematic model:

$$\alpha_i = \alpha_{i,t} + \alpha_c \tag{7.3}$$

where

$$\alpha_{i,t} = h_i(t) \tag{7.4}$$
$$\alpha_c = c \tag{7.5}$$

Here, $h_i(t)$ corresponds to the functions for calculating the steering angles for the turning mode (see section 6.1.1).

The inverse kinematic model for the wheel speeds will be explained in section 7.1.5.

7.1.3 Direct kinematic model

For clipping purposes, it is of interest to be able to calculate a new center of rotation based on a modified wheel steering angle. From the formulas established in section 6.1.1 the direct kinematic model is deduced:

$$r = \frac{a}{\tan \alpha_1} + w \qquad r = \frac{a}{\tan \alpha_2} - w \tag{7.6}$$

$$r = \frac{b}{\tan \alpha_5} + w \qquad r = \frac{b}{\tan \alpha_6} - w \tag{7.7}$$

These relations only take into account the turn around a point configuration. See section 7.1.4.

7.1.4 Clipping

There is a risk of saturating the steering for the wheels when adding the values from the turn and the crab mode. See figure 7.1. This will have to be controlled by some sort of clipping of the wheel steering angles.

Principle

The clipping should take into account the wheel where the total steering amplitude will be largest, clip here, and make this clip affect the steering on the other wheels. This means that after clipping, the movement can still be decomposed into a crab configuration and a turn configuration.

This can be understood by observing the instance in figure 7.1. The front right wheel, when turning right in combination with crab to the right, will saturate more quickly than the rear right wheel (here the two modes will tend to zero out the movement). If only the front right wheel values would be clipped, the total steering result for all wheels would no longer be a sum of a crab and a turn configuration.

Instead of allocating a maximum of half of the wheel steering limit angle α_M to each of the steering components, a dynamical method should be used.

Method

The following method is proposed:

1. Localize the critical angle wheel

Turn, t	Crab, c	Wheel
< 0	< 0	2
< 0	> 0	6
> 0	< 0	5
> 0	> 0	1

Table 7.1: Critical clipping angle

2. Check if the sum of the turn and crab steering angles for this wheel exceed the maximum steering angle
3. If necessary, clip/calculate new crab and steering angles α'_c,α'_t for this wheel
4. Calculate a new turning radius r' from the new steering angle
5. Use r' for calculating wheel steering angles for other wheels, together with new crab angle.

The critical angle wheel is the wheel that risks saturation for a given configuration of turn and crab parameters. The possible configurations are summarized in 7.1.

For this critical angle wheel, the sum of the steering angles calculated from the inverse kinematic model is checked against the maximum possible steering angle, α_M. In other words, clipping will have to be performed if the following is true:

$$\alpha_c + \alpha_t \quad > \quad \alpha_M \tag{7.8}$$

If clipping is necessary, the following operations will be used to find the new steering angles:

$$\alpha'_c \quad = \frac{\alpha_c}{\alpha_c+\alpha_t} \times \alpha_M \tag{7.9}$$

$$\alpha'_t \quad = \frac{\alpha_t}{\alpha_c+\alpha_t} \times \alpha_M \tag{7.10}$$

Here, α'_c is the new clipped crab angle and α'_t the new clipped turn angle. These have the property $\alpha'_c + \alpha'_t = \alpha_M$.

A new turning radius, r', is then found from one of the relations established in section 7.1.3. From this the new steering angles $\alpha'_{t,i}$ for the other wheels can be calculated. The clipped crab component α'_c can be used directly for all wheels. The final clipped steering angles for each wheel can then be expressed as:

$$\alpha'_i \quad = \quad \alpha'_{i,t} + \alpha'_c \tag{7.11}$$

7.1.5 Wheel speeds

The wheel angle configuration of this combined mode does not make all the wheel axes meet in a point, and some minor skidding is inevitable. This also means that there is no ideal wheel speed configuration. The wheel speeds from both modes cannot be added directly to each other, this would result in about double the wanted speed. One could use either the wheel speeds set for the turn around a point mode, or one could use the speed for the crab movement, ie. the same speed on all wheels. It makes more sense, however, after the clipping model has been introduced, to use a weighting system like the one in 7.1.4.

The following is proposed, for each wheel:

$$v_i = a \times v_c + b \times v_{t,i} \tag{7.12}$$

Here, a and b are the clipping factors calculated at the critical angle wheel like explained in section 7.1.4.

$$a = \frac{\alpha_c}{\alpha_c + \alpha_t} \tag{7.13}$$

$$b = \frac{\alpha_t}{\alpha_c + \alpha_t} \tag{7.14}$$

Intuitively, this means that if the crab part uses a majority of the steering capability, it would also dominate the wheel speeds, and vice versa.

7.1.6 Implementation

In the *s_rs_carvel* data structure, which contains the commands for motion control like curvature and speed, a new field has been added. This controls the crab amount, in radians. This value is taken into account in the inverse kinematic calculation. There the wheel angles are taken directly from this angle and applied to all wheels, in addition to the wheel angles calculated by the turn around a point mode.

Clipping is done during the inverse kinematic calculation. First the turn around a point angles and crab angles are calculated normally and added. Then the critical clipping wheel is localized according to table 7.1. Clipping is performed using formulas 7.9 and 7.10, and new steering angles are calculated for every wheel. Wheel speeds are then calculated out from these new clipped steering angles.

In the line-following control algorithm, the curvature command is now calculated from azimuth deviation (u,θ), using the same coefficients as before. The crab command is calculated from position deviation. The crab amount commanded is directly proportional to the position deviation of the rover. The constant to go from deviation in millimeters to crab value in radians, is set to $9\frac{rad}{mm}$.

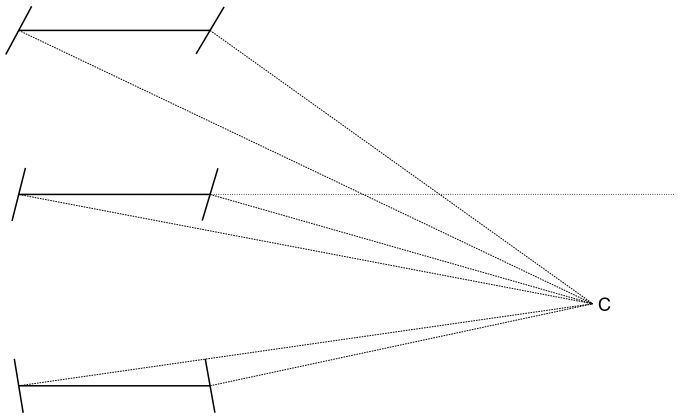

Figure 7.2: Turn and crab combination, second method. Notice that the instantaneous center of rotation, C, is no longer on the line extending from the middle wheel axle. All wheels steer in such a way that their axes meet in C.

No changes are done to the program architecture. Code is just added in existing functions, with extensions of C *case* statements.

7.2 Second method proposed

The second method does not add together two separately calculated angles for each wheel. Instead it takes a more general approach by calculating the steering angles from both the turning and crab commands at the same time.

7.2.1 Principle

The principle is to turn around a point, but this time the point does not have to be on the line extended from the rover's middle axle. See figure 7.2. By moving this point up and down from this center line, intuitively one gets something that resembles a crab movement in addition to the turning. The main difference from the previous method is that the wheels are now still turning around the same point.

The "crab" part of this movement can be demonstrated by setting the turning radius to a very large value. The wheel steering will be minimal. Then imagine the turn center being translated vertically from the middle axle axis. As the turning radius is large, all wheels will tend to steer in the same direction, like in the crab mode.

However, the translation of the rotation point from the middle axle, needed to perform a crab-like movement, will vary depending on the turning radius. This is solved by determining the translation amount in function of the wanted crab an-

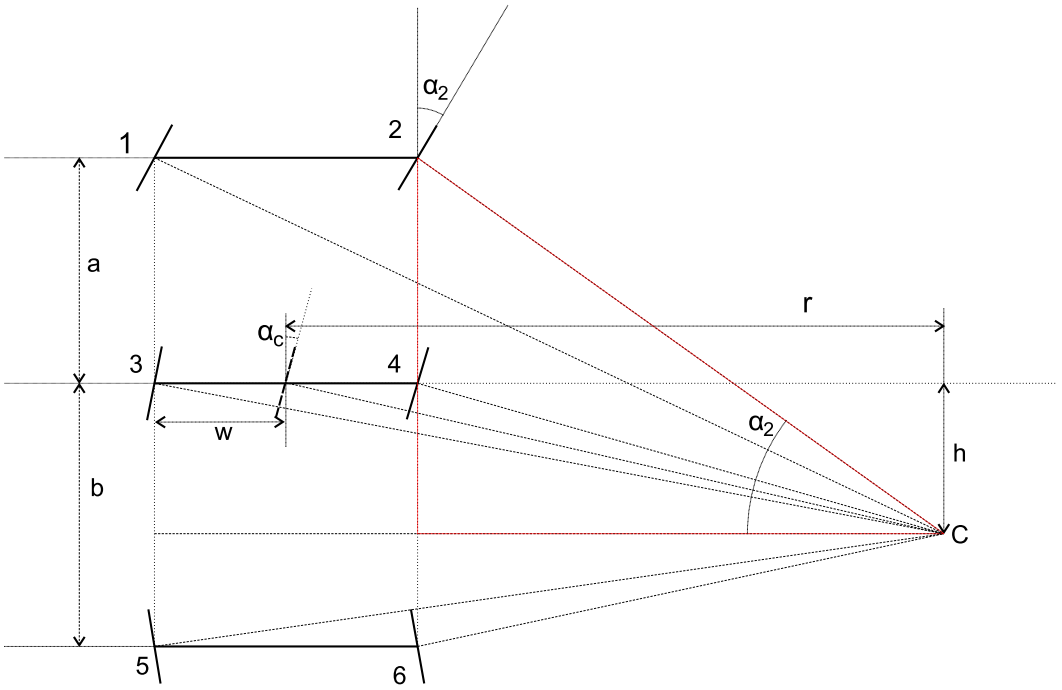

Figure 7.3: Wheel angle determination for turn and crab combination, second method. In this figure, α_2 is taken as an example. The right triangle used for the determination of α_2 is marked in red. h is determined from α_c, which is set to the crab amount. On the figure, r, α_2, α_c, and h are negative values.

gle. This crab angle is set to an imaginary wheel at the center of the middle axle. By using this value and the turning radius, the translation value can be calculated. After this, the real wheel angles can be determined.

As all the wheels turn around the same point, this method produces no skidding. Thus, this method is theoretically more "ideal" than the first method proposed. However, it may seem less intuitive and poses some more problems with the implementation, notably in the clipping part.

7.2.2 Inverse kinematic model

The inverse kinematic model for the second method is quite different from the first. Although we still have $t = f(u, \theta)$ and $c = g(d)$, as in the previous method, we now have $\alpha_i = h_i(t, c)$ instead of $\alpha_i = \alpha_{i,t} + \alpha_c$. No steering values are summed.

Wheel angles

The variables included in the calculations are the same as in 7.1.2, with one addition:

- The distance of the turning center from the line extending the rover's middle axle, h.

In this method all wheel steering angles have to be calculated, except the imaginary crab angle, α_c, in the center of the middle axle. This angle is set to the commanded crab angle, c. See figure 7.3. α_c then controls the "crab" height h by the following trigonometric relation:

$$h = -r \arctan \alpha_c \tag{7.15}$$

When h is found, one can calculate the wheel angles α_i. See figure 7.3. Trigonometric relations are used. By drawing a line between the rotation center and the center of the wheel to be calculated, it becomes the hypotenuse in a right triangle. The other sides lie in the the chassis side and in the line that is parallel to the wheel axles and goes through the rotation center. An equality between angles allows finding α_i with a $\tan$ function. The wheel angle relations then become as follows:

$$\alpha_1 = \arctan \frac{a-h}{r-w} \qquad \alpha_2 = \arctan \frac{a-h}{r+w} \tag{7.16}$$

$$\alpha_3 = \arctan \frac{-h}{r-w} \qquad \alpha_4 = \arctan \frac{-h}{r+w} \tag{7.17}$$

$$\alpha_5 = -\arctan \frac{b+h}{r-w} \qquad \alpha_6 = -\arctan \frac{b+h}{r+w} \tag{7.18}$$

Wheel speeds

Like in the determination of wheel speeds for the four-wheel steered turn around a point mode (see section 6.1.1), one begins with the calculation of the speeds in the middle, here called v_L and v_R. See figure 7.4. The difference here is that these speeds cannot be applied directly to the middle wheel axle. The commanded speed v is attributed to a point that lies on the line going perpendicularly from the rover to C. v_L and v_R are then speeds for imaginary wheels on an axle situated at this point.

$$v_L = v \times \frac{r-w}{r} \qquad v_R = v \times \frac{r+w}{r} \tag{7.19}$$

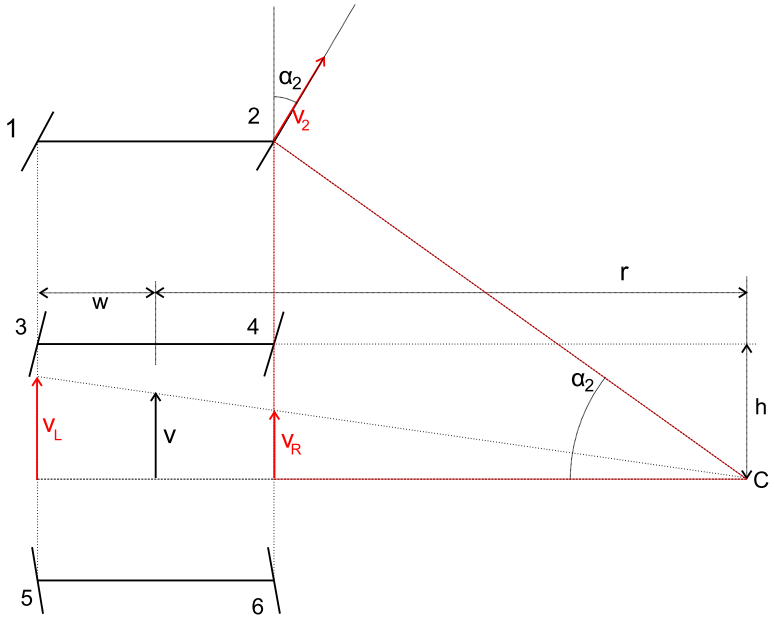

Figure 7.4: Determination of wheel speeds for method 2. Notice that the v_L and v_R speeds are not on the middle axle. The triangle used for the calculation of v_2 is shown in red.

After this, the calculation of the rest of the wheel angles can follow the same proportion rules as explained in section 6.1.1. This gives the following relations:

$$v_1 = \frac{v_L}{\cos\alpha_1} \qquad v_2 = \frac{v_R}{\cos\alpha_2} \tag{7.20}$$

$$v_3 = \frac{v_L}{\cos\alpha_3} \qquad v_4 = \frac{v_R}{\cos\alpha_4} \tag{7.21}$$

$$v_5 = \frac{v_L}{\cos\alpha_5} \qquad v_6 = \frac{v_R}{\cos\alpha_6} \tag{7.22}$$

7.2.3 Clipping

As the inverse kinematic model now is $\alpha_i = h_i(t, c)$, the clipping is not as straightforward as in the first method. A direct kinematic model of this will have to be solved for either t or c. Since both of these variables are supposed to be updated with a new clipped value, the clipping cannot be done at the α level.

An alternative is to do the clipping at the c, t level. However, h_i is not linear This implies that there cannot be any defined max to clip against as there is on the α

level. Clipping can therefore not be accurate. An approximate is found by using

$$a\mathbf{c} + b\mathbf{t} \quad \leq \quad Max \tag{7.23}$$

where a and b are constants adjusting for the different units.

7.2.4 Implementation

The implementation of the second method is done much like the first method (see section 7.1.6). What differs in the inverse kinematic calculation is first the calculation of h. Secondly, there is no clipping in this section, except a final individual saturation clipping for each wheel. This is done just in case the clipping in the line control algorithm should be inaccurate, and should normally not be activated. This makes the inverse kinematic calculation for the second method less complex than for the first method.

In the line-following control algorithm, the commands are calculated just in the same way as in the first method. After this, the command values are clipped, like described in section 7.2.3. The clipped values are then sent to the inverse kinematic calculation. The a and b constants are set to a default value of $1/0.7$ and 1, respectively. This gives values between 0 and 1 for the commands. This gives an adequate, although not completely optimized, clipping in most cases.

Chapter 8

Tests and results

This chapter concerns the testing of the improvements made to the motion control system. Tests are proposed and results are presented. The tests are performed in order to prove the functionality of the implementation, and to analyze its performance.

The tests do not perform exhaustive verification of the software. The software system should be seen as a research product and not a mission-ready system. However, it is not unlikely that the CNES software once will be used more or less heavily as inspiration for mission software. When this day comes, thorough verification will have to be performed in order to ensure safe operation. In any case, it cannot be guaranteed that the whole system will work in every situation, since it is dealing with real-world inputs. These cannot be precisely predicted and reproduced in a testing situation.

8.1 Test conditions

The rover's performance can only be tested in the simulator, as the rover is not operational on the outdoor test site. This makes it difficult to measure factors such as energy consumption and performance on different terrain types. Skidding cannot be measured either, as the simulator has no model for this.

However, the simulator still offers possibility for some evaluation. An overview of the performance of the different locomotion methods can be had by measuring the speed and deviation amounts of each. By observing the rover's behaviour during path execution a general impression can be made.

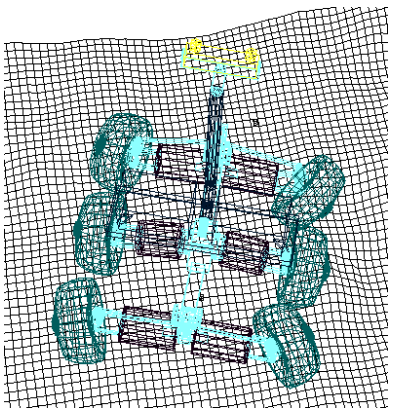

Figure 8.1: Snapshot of four-wheel steering. Notice how the middle axle wheels are not steered.

8.2 Functional observations

The implemented locomotion modes were observed in the rover simulator. They all seem to perform correctly for different situations (turning more or less in different directions, with more or less crab command). Snapshots were taken of the different modes. The four-wheel steering mode now worked correctly when turning both ways. See figure 8.1.

The two six-wheel steering modes worked as expected. Both modes look quite similar. See figures 8.2 and 8.3. They can be distinguished however, by looking at the middle axle wheels. For method 1, it can be seen that these have the same turning angle, while for method 2, they are oriented in a way that their axes meet in a common center of rotation.

This means that it is actually possible to see that the first method will introduce some skidding, as was expected. However, the amount is quite small, far less than in the difference of wheel speeds turning mode (see section 4.1.1).

8.3 General performance tests

8.3.1 Test setup

To have an assessment of the performance of the different steering modes, they were tested on a specially constructed test path. This path was input in the remote operation GUI. The corridor margin was 200mm. The test terrain was *pakua900*, a fairly bumpy terrain. Its height was scaled with a factor of 0.3.

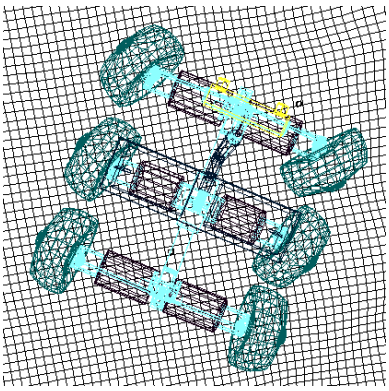

Figure 8.2: Snapshot of six-wheel steering, method 1. Notice how the steering for the middle axle wheels is the same for both wheels. This comes from the “pure” crab part.

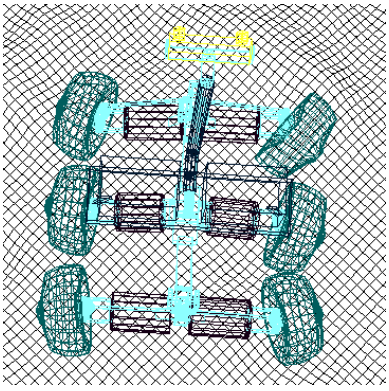

Figure 8.3: Snapshot of six-wheel steering, method 2. All wheels have different steering and their axes meet in the same point.

Steering mode	Time used	Corridor exits
Difference of speeds	3m 22s	3
4-wheel	2m 26s	1
6-wheel method 1	1m 52s	0
6-wheel method 2	1m 40s	0

Table 8.1: Test results

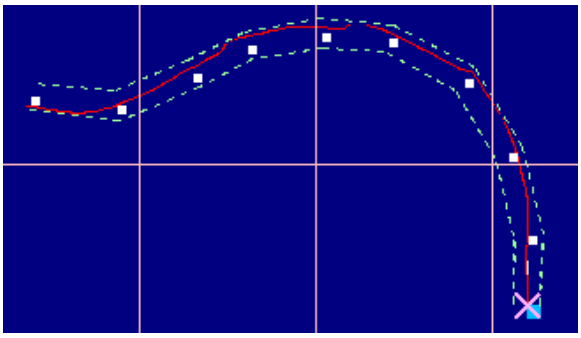

Figure 8.4: Path trace of turning by difference of wheel speeds. The 3 corridor recenter operations can be spotted where the path trace has “bumps”, close to the corridor margin.

Each of the four different steering modes for the motion control algorithm were tested: The difference of wheel speeds mode, the turn around a point mode, and the two six-wheel steering modes.

It was measured how much time each mode used for the path, as well as how many unavoidable corridor exits that were produced. In addition to this an overall assessment was made by looking at the rover’s actual path trace compared to the ideal line segment path. The automatic rotation in place at line segment transitions (see section 6.2) was disabled. This was done in order to better observe the different modes’ capability of deviation correction.

8.3.2 Results

The measurable results are compared in table 8.1. The path traces can be found in figures 8.4, 8.5, 8.6, and 8.7.

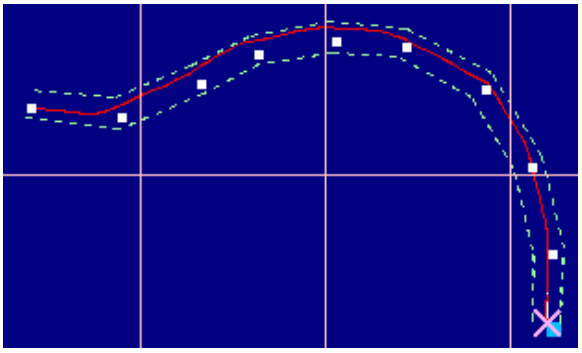

Figure 8.5: Path trace of turning by four-wheel steering. The recenter operation can be spotted close to the 7th waypoint.

Difference of wheel speeds

The difference of wheel speeds turning mode performed worst. It ran into 3 unavoidable corridor exits. It does not manage to do sharp turns and therefore quickly gets problems with a curved path. See figure 8.4. The slowness of this mode can be explained by two factors: Firstly, since there are considerable amounts of skidding, the energy is not used in an efficient way. This makes the rover slower. Secondly, since the rover several times is unable to avoid corridor exits, the control algorithm enters the recenter phase. It takes time to get back on the track with two rotations in place and a return speed that is lower than the normal speed.

Four-wheel steering

The four-wheel steering mode works better than the difference of speeds mode. Only one unavoidable corridor exit was encountered. A considerable time difference between this mode and the previous mode indicates that this mode is much more efficient. By observing the trace in figure 8.5 one can see that it can handle sharper turns without exiting from the corridor. The command sent from the line following algorithm is calculated in exactly the same way for these two modes. This means that it is clear that the wheel steering introduces new flexibility.

Six-wheel steering, method 1

Compared to the previous modes, this mode performs very well. There were no corridor exits detected, and the rover sticks very close to the path. See figure 8.6. The position deviation is corrected very quickly. However, as the crab amount commanded is controlled by the position deviation (see section 7.1), the line following

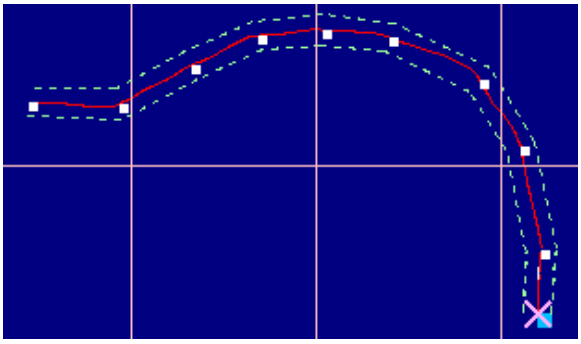

Figure 8.6: Path trace of turning by six-wheel steering, method 1. The rover manages to stick close to the line segments between the waypoints.

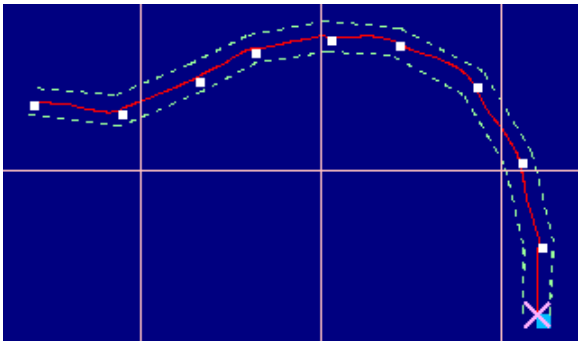

Figure 8.7: Path trace of turning by six-wheel steering, method 2. Little difference can be spotted from the first method.

algorithm has changed. This means that a direct comparison with the previous mode would be incorrect. Although this solution works better than the previous ones, it is possible that the line control algorithm could be tuned to give the four-wheel steering better performance than it currently has. However, as it is now, the six-wheel mode offers vastly increased flexibility over the previous mode. The corridor width could have been decreased. This mode also offers a speed increase, although some of it can be due to the fact that there were no corridor exits detected.

Six-wheel steering, method 2

This mode has little visible difference from the first six-wheel steering mode. See figure 8.7. Speed-wise, it is a bit quicker. Inaccuracies in time-taking and some minor waypoint position changes cannot be excluded, but it is not unlikely that this mode is slightly quicker. This could come from the fact that skidding is, to a high

degree, removed with this mode, and therefore this locomotion is more effective than the first six-wheel steering method.

8.4 Deviation and control analysis

8.4.1 Test setup

The setup for this test is the same as in section 8.3.1. In addition, the line-following algorithm was modified to dump position and heading deviation, as well as the corresponding commanded correction amounts, to a file.

8.4.2 Results

The deviation and command data were gathered and plotted in figures 8.8, 8.9, 8.10, and 8.11. A general observation on all of the plots is that there are sudden discontinuities in the position and heading deviations. With this follows discontinuities in commands too, trying to compensate. These discontinuities are caused by transitions to new line segments. As the new point can be taken into account within a certain range of the endpoint of the previous segment, such discontinuities are bound to appear.

Difference of wheel speeds

The difference of wheel speeds mode plot shows that there is much position deviation, but little heading deviation. See figure 8.8. Heading deviation is quickly corrected, while position deviation is not.

Four-wheel steering

The four-wheel steering mode plot shows that correction is more effective. See figure 8.9. Especially heading deviation is corrected more quickly than in the previous mode. Position deviation is also somewhat decreased, but still a problem. The better heading correction can be explained by the increased steering flexibility the wheel steering gives.

Six-wheel steering, method 1

The first six-wheel mode shows the same amount of heading deviation as the previous mode. See figure 8.10. This is logical, as the curvature component of the

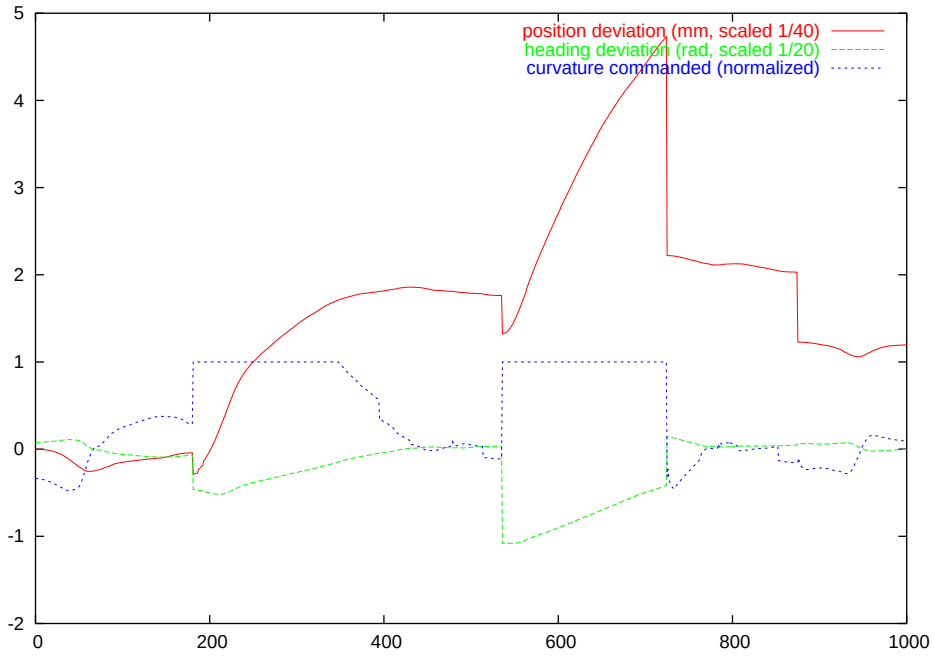

Figure 8.8: Difference of speeds control and deviation. Sudden discontinuities are due to segment transitions or recenter operations.

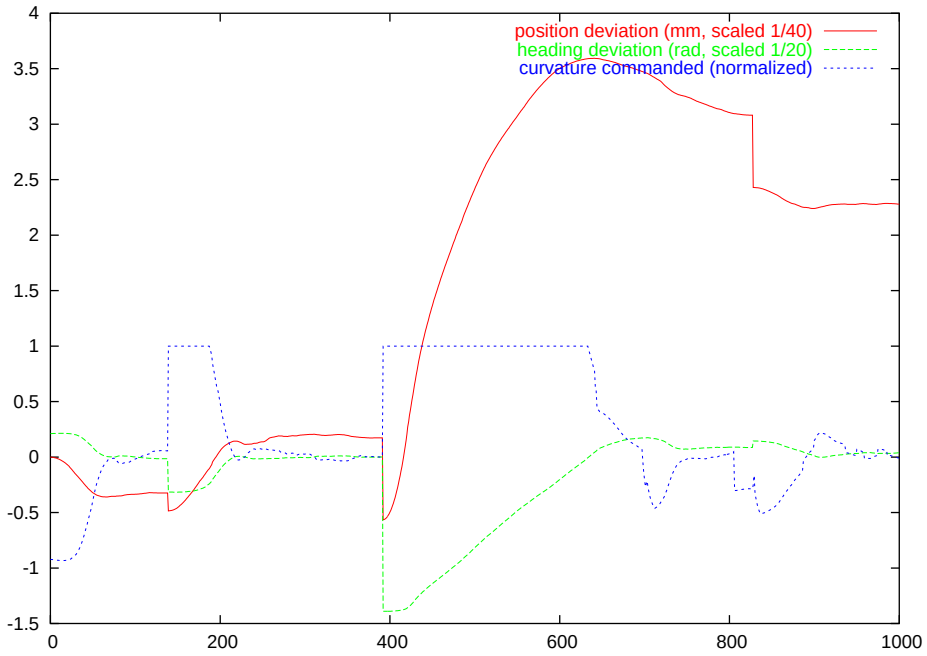

Figure 8.9: Four-wheel steering control and deviation. Notice the change in scale on the y axis.

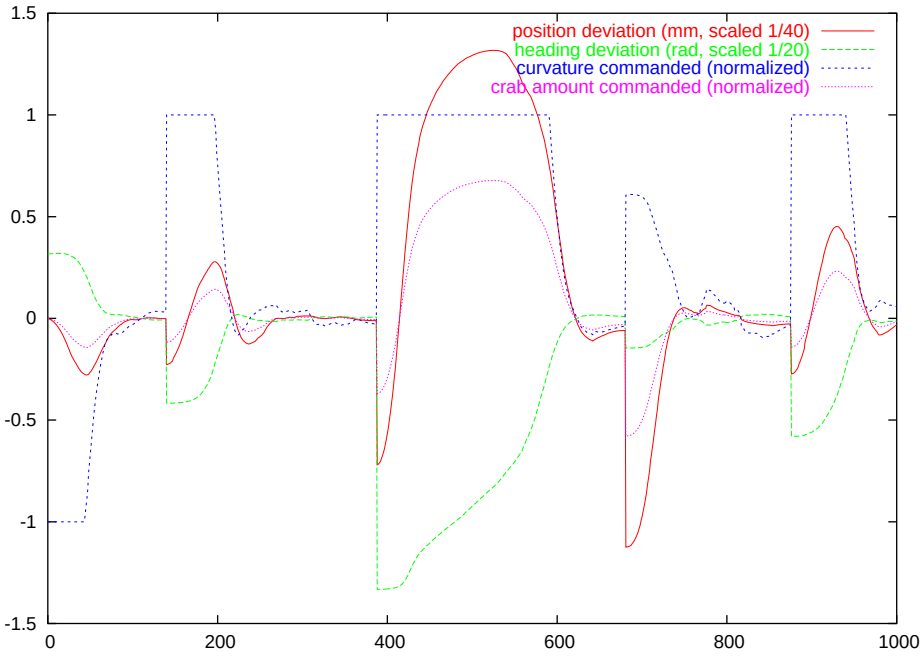

Figure 8.10: Six-wheel steering method 1 control and deviation. Notice the change in scale on the y axis.

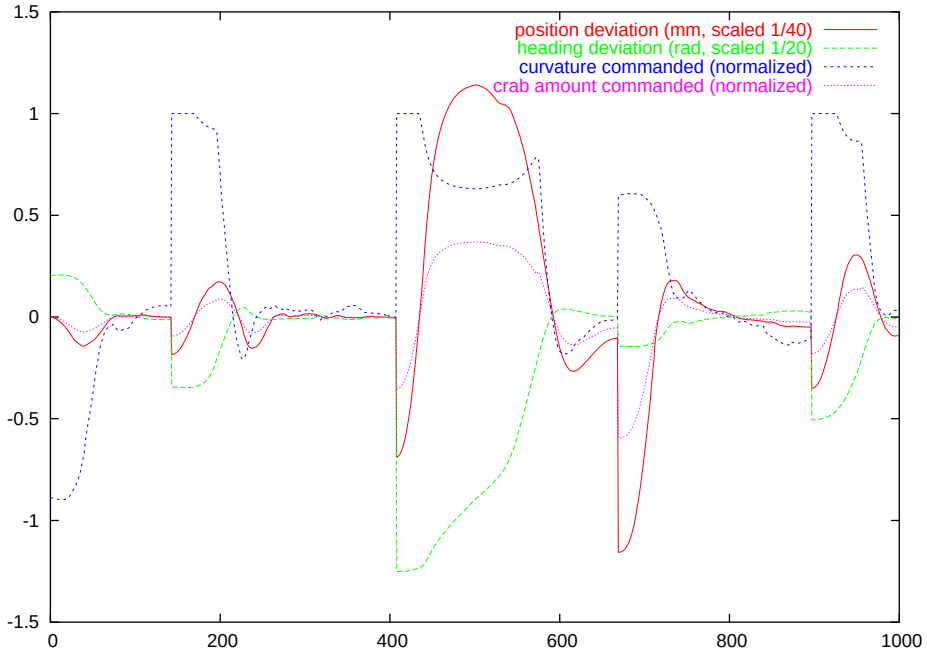

Figure 8.11: Six-wheel steering method 2 control and deviation.

control algorithm is the same, as well as the part of the wheel configuration that takes care of the turning. When it comes to position deviation, great improvements can be observed. The deviation is corrected quite quickly, and never reaches the same amounts as in the two previous modes.

Six-wheel steering, method 2

The second six-wheel mode does not differ much from the first. See figure 8.11. This indicates that the method of commanding the “crab” part separately from the turn part in this mode works as well as the first method. The position deviation is actually a little bit smaller here than in the previous mode, which indicates that this mode could be slightly more efficient.

Chapter 9

Discussion

This chapter discusses the performance and significance of the changes made to the system. Future improvements are proposed, and problems encountered during the project are described. Lastly, a final conclusion is made.

9.1 Results

This section discusses the performance of the motion control system after the general improvements and the six-wheel steering has been implemented.

9.1.1 General improvements

The correction of the four-wheel steering mode (see section 6.1) made this mode perform correctly in all cases. This mode presents a considerable improvement over the steering by difference of wheel speeds mode, in terms of flexibility, accuracy and energy efficiency. The segment transition rotation modification (see section 6.2) made transitions to new segments more efficient, especially when the envelope configuration is used for rotation in place.

9.1.2 Six-wheel steering modes

The new six-wheel steering modes increase the flexibility of the rover's motion capabilities. This is accomplished by a combination of two elements: Firstly, the possibility to more quickly correct position deviations, due to the six-wheel modes' increased use of the rover's steering capabilities. Secondly, the separation of the position correction in the line following control algorithm. This makes it possible to independently correct heading and position deviations. Earlier, a correction in

position deviation led to a new heading deviation. This is now avoided by letting the crab mode use a translational movement to take care of the position deviation.

Both modes perform well, although the second mode has some advantages over the first since it avoids much skidding and is slightly faster. Although the clipping of this second mode is not completely straightforward, it shows that, in practice, the approximate method works well. For the second mode, skidding is not completely removed. As the wheel angles are constantly changing due to different commands in steering, intermediate non-perfect configurations may occur. This is however applicable for the other modes too.

In terms of computation time, the new modes produce no significant penalty compared to the other modes. The resources used on the onboard computer would be minimal compared to perception and path planning algorithms.

The integration of these new modes into the motion control system opens for new possibilities. The increased flexibility can be used to either reduce the corridor width, which allows for more flexible path planning, or to allow the rover to drive on more challenging terrain.

The performance could not be tested with the real rover on the test field. It is however possible that the new six-wheel steering modes could prove to be useful for driving on sideways slopes. A sideways sliding could be compensated by the crab part of the movement. Real-life testing could also make new issues appear, and this could be a base for further fine-tuning of the modes.

9.2 Implementation assessment

The implementation of the general improvements as well as the new steering modes is straightforward. Changes are only made in already existing functions, so the program architecture remains the same. The reconstruction of the four-wheel steering mode led to less and simpler code.

9.3 Future improvements

There are still some aspects of the motion control system that can be improved.

9.3.1 Clipping for second six-wheel steering method

The clipping for the second six-wheel steering method is not yet perfect. The multipliers for the approximate method could be adjusted in order to offer a more optimized division of the steering budget between the crab and the turn parts. Then,

for example, to ensure a maximal use of the budget, a clipping approach like the one used in the first steering method could be introduced as a second pass. It could clip at angle level while locking one of the components to the amount found in the approximate pass.

9.3.2 Smoother segment transitions

The motion control algorithm causes the rover to brake up between each segment transition. If a rotation in place is not needed in the transition, this braking seems unnecessary, as it slows down the rover quite much. This is especially valid when the line segments are short, as is the case for the paths planned by the autonomous navigation algorithm. A system for trying to keep up the rover's speed during segment transitions, as long as it is possible, could be implemented. The discontinuities in deviations when passing to a new segment, as seen in section 8.4, could also be attempted removed.

9.3.3 Line following control algorithm

The line following control algorithm has not been thoroughly analyzed. Its performance could probably be improved, as the tests indicate. For the first two steering modes, only curvature is commanded. It seems like this command is efficient for correcting heading deviation but inefficient for correcting position deviation. For the second two steering modes, the position correction has been isolated with the crab command. This command is now directly proportional to the position deviation, and could possibly be regulated in a more sophisticated way. A small amount of oscillating behaviour could be observed from the traces and the plots in the tests.

It could also be useful to extend the control algorithm to take into account the time used by the motors for steering the wheels to their position. This is not instantaneous, and too quick changes in commands from the control algorithm will result in the motors not being able to follow the pace.

9.4 Problems

Some problems were encountered during the project, that slowed down the progress. In the beginning, much time had to be spent on getting to know the huge software system. The architecture of the system made it difficult and time-consuming to get an overview. The lack of documentation meant that most behaviour had to be analyzed and understood from the source code, which was also sparsely commented.

Setting up a local development version of the system, for only compiling specific modules, also proved to produce some problems. The makefiles were complex and

several problems arose with dependencies and linking.

Originally it was thought that the current motion control system was less developed. It was not known that the four-wheel steering mode was already under construction, and that the line following control system worked with this mode. The focus was changed over to six-wheel steering. Although this made it take some time to find the focus for the project, it was a necessary process for getting an overview of the system.

9.5 Conclusion

The performance of the current motion control system has been improved at some points. New functionality has been developed and implemented, in the form of six-wheel steering. This new mode of locomotion proved to be flexible and well-performing. A fine-tuning of the line following control system is needed to maximize performance, especially in the aspects of deviation regulation and segment transitions.

The new six-wheel locomotion modes will be used in demonstrating CNES' autonomous navigation system. With these new locomotion modes available and the four-wheel steering mode fixed, it will be possible to demonstrate the performance gains by using four or six steerable wheels. This could be useful in designing specifications for rover missions. Hopefully it could be useful in the design and development processes for ESA's planned ExoMars mission.

This report will hopefully be useful for future work on the motion control system, by both giving a documentation of the system and by pointing out things that could be improved in the future.

Bibliography

[1] M. Alexander. Mars transportation environment definition document. Technical report, NASA/Marshall Space Flight Center, March 2001. Available from: ftp://ftp.estec.esa.nl/pub/aurora/Rover/mars_transp_env_def_nasa_tm210935.pdf [cited 15 July 2004].

[2] D.B. Bickler. The new family of jpl planetary surface vehicles. In *CNES, Missions, Technologies, and Design of Planetary Mobile Vehicles*, pages 301–306. January 1993. Abstract at http://adsabs.harvard.edu/cgi-bin/nph-bib_query?bibcode=1993dpmv.book..301B&db_key=INST.

[3] J. Biesiadecki, M. Maimone, and J. Morrison. The athena sdm rover: A testbed for mars rover mobility. In *ISAIRAS 2001: 6th International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, Montreal, Canada, June 2001. Available from: http://robotics.jpl.nasa.gov/people/mwm/sdm-mobility/ [cited 26 July 2004].

[4] R-M. Bonnet and J.P. Swings. The aurora programme. Technical report, ESA, February 2004. Available from: http://esamultimedia.esa.int/docs/Aurora/Aurora625_2.pdf [cited 2004-07-05].

[5] M. Delpech. *Asservissement sur trajectoire*. CNES, Toulouse, January 1999. Internal technical document.

[6] A. Ellery. Elastic loop mobility/traction system study for mars micro-rovers, final report. Technical report, ESA, March 2003. Available from: ftp://ftp.estec.esa.nl/pub/aurora/Rover/ELMS%20STUDY%20FINAL%20REPORT.pdf [cited 15 July 2004].

[7] K. Fletcher. Exomars09 cdf study report. Technical report, ESA, August 2002. Available from: ftp://ftp.estec.esa.nl/pub/aurora/Rover/CDF-14%28A%29%20final.pdf [cited 15 July 2004].

[8] JPL. Mars exploration rover mission [online]. July 2004 [cited 5 July 2004]. Available from: http://marsrovers.jpl.nasa.gov/home/index.html.

[9] M. Maurette and L. Rastel. Planetary exploration. Technical report, CNES, 2004.

[10] V. Michkiniouk, S. Medvedev, and G. Kozlov. Chassis of iares-l planet rover demonstrator with a broad range of functional opportunities. In *I-SAIRAS '97, International Symposium on Artificial Intelligence, Robotics and Automation in Space*, pages 95–103, Tokyo, Japan, July 1997.

[11] A.V. Mitskevich, V.K. Michkiniouk, G.V. Kozlov, and M.I Malenkov. *IARES-L demonstrator chassis description and operating instruction.* VNIITRANSMASH, Sankt-Petersburg, 1996.

[12] NASA. Mars exploration rover landings press kit, January 2004. Available from: `http://marsrovers.jpl.nasa.gov/newsroom/merlandings.pdf` [cited 1 March 2004].

[13] S. Nortman, A. Arroyo, M. Nechyba, and E. Schwartz. Pneuman: A humanoid robot implementation. Florida Conference on Recent Advances in Robotics (FCRAR), May 2002. Available from: `http://www.mil.ufl.edu/publications/fcrar02/sdn_fcrar_02.pdf` [cited 8 July 2004].

[14] RCL. Science and technology rover company limited (rcl) [online]. July 2004 [cited 5 July 2004]. Available from: `http://private.peterlink.ru/rcl/`.