École Centrale de Nantes

University of Genova

MASTER ERASMUS MUNDUS

EMARO "European Master in Advanced Robotics"

2012/2013

Thesis Final Report

Presented by

Nemanja Rakićević

On 12/09/2013

Title

Guidance, Navigation and Control for a Planetary Rover

Jury

President:	Wisama Khalil	Professor, IRCCyN, ECN
Evaluators:	Wisama Khalil	Professor, IRCCyN, ECN
	Philippe Martinet	Professor, IRCCyN, ECN
	Gaëtan Garcia	Assistant Professor, ECN
	Weizhong Guo	Professor, SJTU, Shanghai

Supervisors:

Masaki Takahashi
Keio University, Department of System Design Engineering

Philippe Martinet
Laboratory: Institut de Recherche en Communications et Cyberntique de Nantes

Giuseppe Casalino
Laboratory: Genoa Robotics and Automation Laboratory, DIST

Abstract

Lately, the importance of autonomous rovers in space exploration, through the Mars missions, has been proven as incredibly great, due to the quality and quantity of information they convey to the home base. The main challenge in making these rovers autonomous, is to enable them to understand their surroundings and successfully navigate through the environment. The report at hand introduces a short overview of the state-of-the art in rover planetary exploration and presents a novel approach to real-time path selection and terrain mapping. A mapping method which uses patches, made out of local views taken at specific locations, that form the global map of the explored area is explained. Adding to this, a control algorithm based on fuzzy logic is elaborated. The fuzzy controller is based on the terrain information stored in the global map, and especially concentrates on the terrain roughness information and not solely on the discrete classification to traversable and untraversable areas based on a pre-set threshold. Having formed this system, it has been tested in simulation with carefully planned terrain maps. After being validated in simulation, the program has also been tested on the actual rover system "Cuatro" at the JAXA's Institute of Space and Aeronautical Science rover laboratory, using realistic terrain set ups. Finally, the test results are displayed and analysed, in order to form a final conclusion of the systems' performances.

Résumé

L'importance des rovers autonomes se révèle d'une grande importance dans l'exploration de l'espace, notamment à travers les missions sur Mars, grâce à la qualité et la quantité d'informations qu'ils communiquent à la base. Le défi principal que représente l'augmentation de l'autonomie des rovers est de leur donner la faculté de comprendre leur environnement et d'y naviguer. Le présent rapport introduit une brève vue d'ensemble de l'état actuel de l'exploration planétaire par les rovers et présente une nouvelle approche de la sélection des trajectoires en temps réel et du mapping du terrain. Nous exposons une méthode de mapping qui utilise les patchs, faits à partir des vues locales prises aux locations spécifiques et formant le plan global de la zone explorée. En addition, nous élaborons un algorithme de contrôle à partir de la logique fuzzy. Le contrôleur fuzzy est basé sur l'information qui provient du terrain et qui est stockée dans le plan global. Il est spécialement concentré sur l'information concernant la rugosité du terrain et pas seulement sur la classification discrète des zones traversables et non-traversables basé sur un seuil préréglé. Après sa construction, ce système a été testé dans une simulation comportant des plans du terrain soigneusement planifiés. Une fois validé par la simulation, ce programme a également été testé sur le système du rover "Cuatro" dans le laboratoire rover à l'Institut de l'Espace et des Sciences Aronautiques de la JAXA, à l'aide des configurations de terrains réalistes. Finalement, les résultats des tests sont affichés et analysés, ayant comme but final une conclusion portant sur les performances du système.

要旨

近年，火星ミッションを通じて，基地局に伝える情報の質と量から宇宙探査における自律ローバの重要性が証明されている．ローバの自律化における主な課題は，周囲環境を理解し，目標地までのナビゲーションを達成することである．本稿では，最新のローバの惑星探査の概要を簡単に述べ，リアルタイムの経路選択や地形マッピングの新しいアプローチを提示する．マッピングでは，パッチを用いて特定の位置でのローカルビューから探索領域のグローバルマップを作成する方法について説明する．これに加えて，ファジィ理論に基づく制御アルゴリズムを述べる．ファジィ制御ではグローバルマップに格納された地形情報に基づいて，事前に設定した閾値に応じて単にローバが通過できるか否かといった離散的な分類ではなく地形の凹凸も考慮した設計を行う．そして，実際の地形情報を用いたシミュレーションにより本システムの検証を行った．また，シミュレーションでの検証後，宇宙航空研究開発機構(JAXA)の宇宙科学研究所において，実際にローバシステム "Cuatro" を用いて，惑星表面を模擬した環境での実機試験を行った．最後に結論として，本システムの性能試験の分析結果を提示する．

Acknowledgements

I would like to express my sincere gratitude to the people who helped me a lot during the development of my thesis. My mentor, professor Masaki Takahashi, who was abundantly helpful and offered invaluable assistance, support and guidance. Deepest gratitude are also due to professor Genya Ishigami who provided me with the possibility to test my code on the JAXA's test bed rover "Cuatro" at the Institute of Space and Aeronautical Science (ISIS) rover laboratory and his patience and suggestions. Special thanks also to all the graduate friends, especially Mr Ayanori Yorozu whose practical advice and the discussions we had were immensely useful.

Contents

List of Figures

Abbreviations

B.C.	**B**efore **C**hrist
A.D.	**A**fter **D**eath
TOF	**T**ime **O**f **F**flight
RGBD	**R**ed **G**reen **B**lue **D**epth
JAXA	**J**apanese **A**erospace e**X**ploration **A**gency
ISAS	**I**nstitute of **S**pace and **A**stronautical **S**cience
DOF	**D**egree **O**f **F**reedom
IMU	**I**nertial **M**easurement **U**nit

Chapter 1

Introduction

1.1 Human Space Exploration

Since the dawn of man, there have always been inquisitive people gazing at the sky and wondering about what is the explanation of that they see and what is there beyond. These people can be considered as the first space explorers. Space exploration, is by definition the investigation, by means of manned and unmanned spacecraft, of the reaches of the universe beyond Earth's atmosphere and the use of the information so gained to increase the knowledge of the cosmos and benefit humanity.

The earliest considered noted work on recognizing that astronomical phenomena are periodic and applying mathematics to their predictions were done during the First Babylonian Dynasty (ca. 1830 B.C.) in Mesopotamia. Additionally, in Ancient Egypt astronomy played a significant role in religious matters for fixing the dates of festivals and determining the hours of the night. Some of the first known formal definitions of the universe were formed by ancient Greek philosophers, starting from around 400 B.C. The first idea of a heliocentric model was proposed by Aristarchus of Samos around 300 B.C. Other scientists calculated the Earth's circumference and its relative distance to the Moon, some of which very accurately. A very important astronomer Hipparchus of Nicaea, considered the founder of trigonometry, introduced the concept of exact prediction into astronomy and gave a quite complete star catalogue for that time. A notable astronomer of the Roman era was Claudius Ptolemy who introduced the concept of equant in order to explain how to predict the behaviour of the planets. Moreover, very old Indian astronomical transcripts have been found, which detailed astronomical occurrences generally applied for timing social and religious events. Astronomy in China has one of the longest histories, with its beginnings dating back to the Shang Dynasty (around 13^{th} century B.C.). Highly accurate observations of space phenomena were made, giving cosmological models, sky partition into constellations, star catalogues and maps, etc. Also, the development of astronomical equipment.

While Europe was falling into the Dark Ages, the Greco-Roman astronomy was superseded by Arabic. This type of research was supported through the House of Wisdom situated in present day Baghdad, Iraq. It has used and reshaped the Ptolemaic model, and gave basis for further development. One of the contributions was also the development of astronomer's instruments such as astrolabes, equatorium, sundials and other.

The European Renaissance period brought fundamental innovations and some of the most important works in astronomy and astrophysics. In 1543, Nicolaus Copernicus in his book *De revolutionibus orbium coelestium* (On the Revolutions of the Celestial Spheres) provided a full

mathematical discussion of the proposed heliocentric system. His work was further defended by Galileo Galilei. In 1609, Galileo invented the first refracting telescope, with which he observed the sky and celestial bodies, and, amongst other, he observed the Moon's craters and for the first time the four largest moons of Jupiter. Johannes Kepler was the first to attempt to derive mathematical predictions of celestial motions from assumed physical causes and thus developing the three laws of planetary motion (1609-1619) which unite the previous observations and physics. These relations were further deepened by Sir Isaac Newton through his work *Philosophiae Naturalis Principia Mathematica* (Mathematical Principles of Natural Philosophy) in 1687. and his theory of universal gravity. His dedication laid ground for many modern approaches. During the 18^{th} and 19^{th} century, the new progress in classical mechanics done by Euler, Clairaut, D'Alembert, Lagrange and Laplace enabled a more precise estimation of the motions of celestial bodies. Further, the calculation of the masses of these object have been done also.

FIGURE 1.1: Early space explorers: Aristarchus, Ptolemaeus, Kopernikus and Galileo (images taken from Wikipedia)

Technological advances in the 20^{th} century led to an even faster expansion of the human's knowledge and understanding of space. This allowed the demonstration of the existence of the Earth's galaxy, the Milky Way, as a separate group of stars, and also the existence of "external" galaxies, the expansion of the Universe since other galaxies are moving away from ours, etc.

The subsequent step that people made was to, apart from observing the space, also try to reach it. This would not have been possible without Wernher von Braun and his work on the Agregate series of rockets leading to the development of the V-2 rocket. It was the first human made object to cross the Kármán line which defines the borderline of outer space (3 October 1942). By further modification of V-2 people were able to achieve more; perform tests, take pictures, send insects (fruit flies), etc. Another important event was when the USSR launched the first artificial satellite into orbit, the Sputnik 1, with which they could communicate. The first, complex organism, an animal, to be sent to outer space, was the famous dog Laika (1957), but unfortunately she died, probably due to overheating. In 1959, the Soviet "Luna 2" was the first spacecraft to reach the Moon surface. Four years after Laika, the first human to exit the Earth's atmosphere and the pioneer of human space flight was Yuri Gagarin (12 April 1961) through the Vostok 1 mission. This is the period when the "Space Race" started between the United States of America and the Soviet Union. The race led to an accelerated development of technologies that always improved the human space exploration.

A new breakthrough was when man finally set foot on another celestial body and successfully launched from it. Apollo 11 astronauts Neil Armstrong, Buzz Aldrin and Michael Collins landed their shuttle on the Moon on 20 July 1969. This moment represents a milestone and the beginning of planetary exploration. Afterwards, during the following decade, humans have returned to the Moon five times within the "Apollo" missions, where the last one was "Apollo

17" in 1972. The last three of these missions had a human-driven lunar rover. On the other hand, in 1971, the Soviets launched a first autonomous lunar rover "Lunokhod 1" and 2 years later the second "Lunokhod 2". Meanwhile, the first interplanetary surface mission to return surface data from another planet was the 1970 landing of "Venera 7" on Venus. After, in 1971, the "Mars 3" mission performed a soft landing on Mars and sent signals. Later, longer surface missions were achieved, including Mars surface operation by the US "Viking 1" from 1975 to 1982 and on the surface of Venus the Soviet "Venera 13" in 1982.

FIGURE 1.2: First artificial satellite Sputnik 1, Laika, the golden plate on board Voyager 1 and the first Moon walk (images taken from Wikipedia)

Later on, many other missions had been realized that composed of satellites, which travelled large distances and gathered information about space and other planets of the solar system. The most significant of these is the "Voyager 1" mission which is now the farthest man-made object from Earth. It was launched by NASA on 5 September 1977, and as of 15 June 2012, it has been reported that it is to become the first artificial satellite to leave the solar system. It was able to provide detailed images of Saturn and Jupiter and their moons. With it, the Voyager carries a gold-plated audio-visual disc in the event that this spacecraft is ever found by intelligent life-forms from other planetary systems. The Hubble Space Telescope was launched in 1990 and it enables the observation of Deep Space phenomena. The International Space Station is a modular structure whose first component was launched in 1998. It consists of pressurised modules, external trusses, solar arrays and other components and it is used as a multifunctional research laboratory for conducting experiments in various fields, with the possibility to be inhabited by a crew.

The planetary exploration is currently performed by autonomous rovers, with sophisticated equipment which is used to examine the terrain and atmosphere of the planets on which it is deployed. This is done for various reasons; in order to survey whether a planet is suitable for human life, learn about its geology, search for traces of extra-terrestrial life forms, etc. Currently, the attention has been focused on Mars and the recent rover missions have been directed there. Such as "Mars Pathfinder", "Mars Exploration Rover" and the most recent "Mars Science Laboratory". The new technologies enable the exploration of Mars in new ways, resulting in higher-resolution images, precision landings, longer-ranging surface mobility and even the return of Martian soil and rock samples for studies in laboratories on Earth.

In the future, the space agencies see the Moon missions as essential to the exploration of more distant worlds. Since the further goal would be space colonization, also called space settlement and space humanization, as the permanent autonomous (self-sufficient) human habitation of locations outside Earth, especially of natural satellites or planets such as the Moon or Mars. This can be seen since most of the announced missions are lunar-based. Such is the International Lunar Network, a proposed network of a series of landed stations of the US and other countries, to be installed by the end of the decade (first launch expected 1 March 2018). The Russian Luna-Glob is the first of four missions planned before the creation of a fully robotic lunar base scheduled for after 2015. Also the Chinese and Indian space companies have the ambition to send a manned mission to the Moon by 2020/30.

1.2 Using Rovers in Space Exploration

A planetary rover is a space exploration vehicle projected to traverse on the surface of celestial bodies and perform desired tasks. They are used instead of manned mission for many reasons. Principally, because humans are unable to withstand the conditions in some environments where the rovers can. Then, the cost of sending a rover is much lower both financially and concerning liability issues, also because the robot does not need any support systems. Additionally, most of the rover missions so far do not foresee the return of these vehicles which is unacceptable for human beings. During the Apollo lunar program, the rover used was human-driven so it was used just to facilitate the humans exploration and help cross large distances easily. The primary purpose of the "unmanned" rovers is to gather information about the terrain they are on and to transmit them back to Earth for processing. NASA distinguishes between "mission" objectives and "science" objectives. Mission objectives are related to progress in space technology and development processes and the latter are met by the instruments during their mission in space. The advantages over stationary landers are that the mobile rovers can examine more territory and can be directed to interesting sites. If they are solar powered, they can place themselves in sunny positions for optimal usage of the Sun's solar energy. Compared to orbiting spacecraft, they can make observations to a microscopic level and directly conduct physical experimentation. However, they are more prone to failure due to the intricate nature of landing.

To position and control a rover on another planet is a complex task which comprises of the launching part, guidance towards the goal, the most complex and error prone landing part and communication establishment and control. Rovers usually arrive on the planet's surface using a lander type spacecraft. This lander is a part of system involving an entry capsule, a supersonic parachute, followed by solid rockets and large airbags to cushion the impact. Recently, for the latest "MSL - Curiosity" mission which was heavier, a new high-accuracy entry, descent, and landing (EDL) with the novel sky crane powered descent landing system was used, placing Curiosity within a 20 by 7 km landing ellipse, in contrast to the 150 by 20 km accomplished by the previous "MER" landing system. During and after descent, the rovers have to withstand high levels of acceleration, high and low temperatures, pressure, dust, corrosion, cosmic rays and remaining functional without repair for a needed period of time.

In this report, attention is focused on the independent mobile rovers used, without physical human guidance. An overview of those used in previous missions and the ones still active in ongoing missions is given.

In the following, a short presentation of some of the successfully performed rover missions is given. They are divided into two groups: Lunar and Martian rovers. The rover specifications, together with mission objectives, are presented followed by the mission outcome.

1.2.1 Moon

Lunokhod 1

The Lunokhod 1 rover, shown in figure 1.3 on the left, as stated in the first chapter, was the first roving remote-controlled robot to land on any celestial body. It landed on the Moon in November 1970. It was launched by the Soviet Union, as part of its "Lunokhod" program, on board the Luna 17 spacecraft on 10 November 1970. The rover was equipped with a cone-shaped antenna, a highly directional helical antenna, four television cameras, and special extendable devices to test the lunar soil for soil density and mechanical property tests. It also had a X-ray spectrometer, X-ray telescope, cosmic

ray detectors, and a laser device. The Lunokhod was powered by batteries which were recharged during the lunar day by a solar cell array. It also possessed a heater unit that kept the internal components at operating temperature during lunar nights.

Lunokhod was intended to operate through three lunar days[1] but actually operated for 11 lunar days, when the mission was declared over due to loss of communication. For this time it travelled 10540 metres and returned more than 20000 TV images and 206 high-resolution panoramas. In addition, it performed 25 lunar soil analyses with its RIFMA X-ray fluorescence spectrometer and used its penetrometer at 500 different locations.

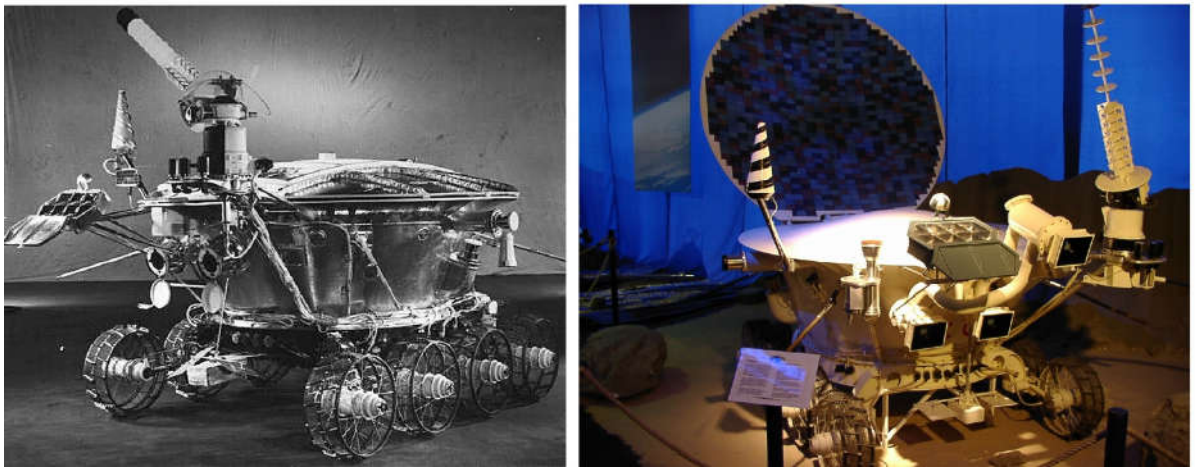


FIGURE 1.3: Lunokhod series Soviet Moon exploration robot vehicle. Lunokhod 1 (left) and Lunokhod 2 (right). (images taken from Wikipedia)

Lunokhod 2

It was a successor to Lunokhod 1 which began its lunar mission on 16 January 1973, being launched aboard the Luna 21 spacecraft. It had a more advanced construction than its predecessor. Lunokhod 2 (1.3 right) was equipped with three television cameras, one mounted high on the rover for navigation and four panoramic cameras. Scientific instruments included a soil mechanics tester, solar X-ray experiment, an astrophotometer to measure visible and ultraviolet light levels, a magnetometer, a radiometer, a photodetector (Rubin-1) for laser detection experiments, and a French-supplied laser corner reflector. The primary objectives of the mission were to collect images of the lunar surface, examine ambient light levels to determine the feasibility of astronomical observations from the Moon, perform laser ranging experiments from Earth, observe solar X-rays, measure local magnetic fields, and study the soil mechanics of the lunar surface material.

Lunokhod 2 operated for about 4 months, when the program was declared over, because of the failure due to overheating caused by the dust entering the rover heater. While it was operational, it covered more than 37 km of terrain and sent back 86 panoramic images and over 80000 TV pictures. Also many mechanical tests of the surface, laser ranging measurements, and other experiments.

[1] A lunar day is a period of time needed for the Earth's Moon to complete one full rotation on its axis with respect to the Sun, which is approximately one Earth month

1.2.2 Mars

Sojourner [Mars Pathfinder]

Sojourner rover, a part of NASA "Mars Pathfinder" mission, depicted in figure 1.4, was the first rover to successfully reach another planet. It was launched on 4 December 1996, and started its mission on 4 July 1997. The Sojourner is a 6-wheel vehicle with a rocker-bogie suspension system (this concept is implemented in all NASA's Mars exploration rovers). The rover's dimensions are length 65 cm, width 48 cm and height 30 cm and it weights 10.5 kg with a maximum speed reaching one centimeter per second. For processing it carried a CPU which was a 80C85 with a 2 MHz clock. Moreover, the rover had three cameras: front B&W stereo and 1 rear color, Laser striper hazard detection system, Alpha Proton X-ray Spectrometer, Wheel Abrasion Experiment, Materials Adherence Experiment and accelerometers. It had solar panels and a non-rechargeable battery, which could allow some night time operations, but once the batteries depleted, it could only operate during the day.

Concerning its control, Sojourner operations were supported by Rover Control Software, which ran on a Silicon Graphics Onyx2 supercomputer back on Earth, and allowed command sequences to be generated using a graphical interface. The rover driver would wear 3D goggles supplied with imagery from the base station, and move a virtual model with a specialized type of joystick. This method allowed the rover and surrounding terrain to be viewed from any angle or position, supporting the study of terrain features, placing waypoints, or doing virtual flyovers. Sojourner's on-board safety system also looked for obstacles.

The final data transmission received from Pathfinder was on 27 September 1997, and the failure is assumed to be due to the malfunctioning of the battery used to heat the essential components during night. Sojourner travelled approximately 100 m in total, never more than 12 m from the Pathfinder station. During its 83 sols[2] of operation, it sent 550 photographs to Earth and analysed the chemical properties of 16 locations near the lander.

Spirit (MER-A) and Opportunity (MER-B) [Mars Exploration Rover]

The NASA's MER mission is a still active robotic Mars exploration mission consisting of two rovers. It represents the continuation of the NASA's Mars Exploration Programme and its scientific objective was to search for and characterize a wide range of rocks and soils that hold clues to past water activity on Mars. It consists of two identical rovers Spirit and Opportunity (one model presented in figure 1.4), launched on 10 June and 7 July 2003, respectively.

The Mars Exploration Rovers are six-wheeled, solar-powered robots, of dimensions: 1.5 m high, 2.3 m wide and 1.6 m long, weighing 180 kg. Each wheel has its own motor and the two front and two rear wheels each have individual steering motors, which allows the vehicle to turn in place. Their structure can endure the inclination of terrain of up to 45° without tipping over, but its hazard avoidance system never subjects it to tilts of more than 30°. The average speed of the rovers is 10 mm/s (max 50 mm/s), because its hazard avoidance software causes it to stop every 10 seconds for 20 seconds to observe and understand the terrain into which it has driven. This will be explained in more detail in

[2] A mean Martian solar day, or "sol", lasts 24 hours, 39 minutes, and 35.244 seconds.

chapter 3. The rovers run a VxWorks embedded operating system on a radiation-hardened 20 MHz RAD6000 CPU. Their essential components have to remain within a temperature range of 40 °C to +40 °C, so during night, special heaters are used. Each rover has a total of 9 cameras, which produce 1024-pixel by 1024-pixel images at 12 bits per pixel, but most navigation camera images and image thumbnails are truncated to 8 bits per pixel to conserve memory and transmission time. The MER rovers carry on themselves much more scientific equipment compared to the previous missions. One group consists of the Panoramic Camera (Pancam), for determining the texture, color, mineralogy, and structure of the local terrain; Navigation Camera (Navcam), that has higher field of view but lower resolution and is monochromatic, for navigation and driving; A mirror for the Miniature Thermal Emission Spectrometer (Mini-TES), which identifies interesting rocks and soils for closer examination. Also, four monochromatic hazard cameras (Hazcams) are mounted on the rover's body, two in front and two behind. They are used also to help with navigation and obstacle detection. The instrument deployment device (IDD), the robot manipulator, holds additional instruments: Mössbauer spectrometer (MB) for mineralogy tests; Alpha Particle X-Ray Spectrometer (APXS); Magnets, for collecting magnetic dust particles, which are then analysed by the previous two instruments; Microscopic Imager (MI) for obtaining close-up, high-resolution images of rocks and soils; Rock Abrasion Tool (RAT), developed for removing dusty rock surfaces and exposing fresh material for examination by instruments on board.

On 29 April 2006, the Spirit rover got stuck in sand and in March the following year communication with it went on hiatus. Finally, on 25 May 2011, its mission was declared finished. Conversely, the Opportunity rover is still active and is exploring the Mars surface. A more complete timeline of the MER mission is given in Appendix A.

***Curiosity* [Mars Science Laboratory]**

The Curiosity rover is currently the most sophisticated and highly equipped NASA's rover (figure 1.4). It was successfully launched for Mars on 26 November 2011 and landed on the surface on 6 August 2012. The main objectives are to determine whether Mars could ever have supported life, and search for evidence of past or present life on Mars. Moreover, to investigate Mars' habitability, studying its climate and geology. It is possible that NASA will send another MSL type rover on a second mission by 2020.

The six-wheeled rocker-bogie system, has no solar panels, but is powered by a radioisotope thermoelectric generator (RTG), weighs 899 kg and can travel up to 90 m/h. Each wheel is 50 cm in diameter, and has cleats for better gripping and is independently actuated and geared, to easily overcome in soft sand and rocky terrain. The four corner wheels are steered independently, so the robot can turn in place and perform arc paths. Each wheel has a pattern that helps it maintain traction and leaves patterned tracks in the sandy surface of Mars, which is used by the on-board cameras to judge the distance travelled, by the means of visual odometry. The pattern itself is Morse code for "JPL" (·--- ·--· ·-··). Based on the center of mass, the vehicle can withstand a tilt of at least 50° in any direction without overturning, but automatic sensors will limit the rover from exceeding 30° tilts. For data processing it has two identical on-board rover computers, "Rover Compute Element" (RCE), which use the RAD750 CPU (a successor to the RAD6000 CPU used in the Mars Exploration Rovers) operating at 200MHz and also operates on VxWorks, a real-time operating system.

The rover carries a very large variety of scientific instruments which help it fulfil its desired goals. The most interesting to note is the ChemCam developed by the IRAP

laboratory based in Toulouse, a suite of remote sensing instruments, including the first laser-induced breakdown spectroscopy (LIBS) system to be used for planetary science, and Curiosity's fifth science camera, the remote micro-imager (RMI). Likewise, there are 12 additional cameras that support mobility: Hazard avoidance cameras (Hazcams)located on each of its four corners, providing closed-up views of potential obstacles about to go under the wheels. Navigation cameras (Navcams) mounted on the mast to support ground navigation and provide a longer-distance view of the terrain ahead.

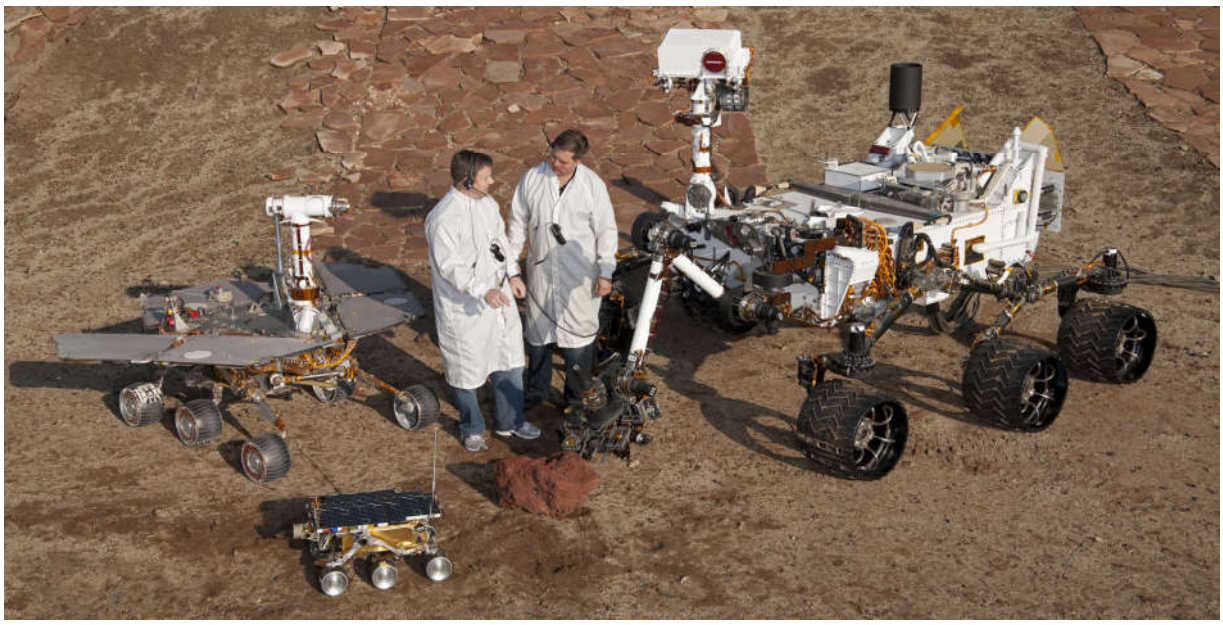


FIGURE 1.4: The three Mars Exploration Program rovers (left to right): MER, Sojourner and Curiosity. (images taken from Wikipedia)

1.2.3 Scheduled Rover Missions

Chang'e 3

Is a rover announced by the China National Space Administration (CNSA) to be launched within 2013, as a part of their Lunar Exploration Program. The purpose of this rover would be the inspection of the Moon's surface and resources, and also to provide data to determine the selection of a site for a manned Moon base. A six-wheeled lunar vehicle has been under development since 2002, around 1.5 m high, weighing 200 kg, this rover is designed to transmit video in real-time, dig into the lunar surface and analyse soil samples. Its designed average speed is 100 m/h, and it can negotiate inclines and use automatic sensors to avoid the obstacles.

Chandrayaan 2

Is a lunar rover developed by the Indian Space Research Organisation (ISRO) in collaboration with Russia, to be sent to the Moon using a Geosynchronous Satellite Launch Vehicle (GSLV) in 2015, but might be postponed due to the delay in the construction of the lander. The rover will weigh 30-100 kg and will operate on solar power. The rover will move with 6 wheels on the lunar surface, pick up samples of soil or rocks, perform chemical analysis and send the data to the orbiter above, which will relay it to the Earth station. The rover will be controlled from the ground team, and its mobility is ensured

using the following systems: Stereophonic camera based 3D vision; Kinematic traction control which will permit the rover to traverse the rough lunar terrain thanks to the independent steering on four of the wheels; Control and motor dynamics system which control 6 separate motors on each of the wheels. Regarding the equipment, the rover will also have a Laser induced Breakdown Spectroscope (LIBS) and Alpha Particle Induced X-ray Spectroscope (APIXS).

ExoMars Rover

Is the European Space Agency's (ESA) rover, which is still under testing, developed within the ExoMars program in cooperation with the Russian Federal Space Agency (Roscosmos). It is scheduled to be launched in 2018, as the second part of the mission after the orbiter. The ExoMars Rover, is designed to operate on the Martian surface for 180 sols, however an extension of this lifetime is expected. Some of the objectives of this mission are to search for possible biosignatures of Martian life, past or present and also to study the surface environment and identify hazards for future manned missions. The configuration which is currently being considered is the EXOMADER (EXOMArs DEmonstration Rover) shown in figure 1.5 . It is a 1/2 scale model of the rover considered for ESA's ExoMars mission.

FIGURE 1.5: The EXOMADER, on a test bed (left) and the CAD model (right). (images taken from ESA's website)

Due to the communication delay, the remote control is intended to be substituted by autonomous software for visual terrain navigation, using compressed stereo image information, obtained from the panoramic and the infrared cameras set on the mast, and independent upkeep. In this method, digital maps are created from navigation stereo pair cameras and the suitable trajectory is computed autonomously. Close-up collision avoidance cameras are used to ensure safe evasion of obstacles. Employing this system, the rover is enabled to navigate and safely travel approximately 100 m per day. All wheels can be individually pivoted to adjust the Rover height and angle with respect to the local surface, and to create a sort of walking ability, particularly useful in soft, non-cohesive soils like dunes. In addition, inclinometers and gyroscopes are used to enhance the motion control robustness. Finally, Sun sensors are utilised to determine the Rover's absolute attitude on the Martian surface and the direction to Earth. Thus the rover provides key mission capabilities: surface mobility, subsurface drilling and automatic sample collection, processing, and distribution to instruments. It hosts a suite of analytical instruments dedicated to exobiology and geochemistry research, the Pasteur payload.

Mars Exploration with a Lander-Orbiter Synergy (MELOS)

MELOS is the Japanese Aerospace Exploration Agency's (JAXA) proposed mission to Mars, which was scheduled between 2016 and 2020. The stated goal of the mission is to explore Martian atmosphere, water and climate. The mission would consist of an orbiter which would observe the Martian atmosphere, and the landers would be used to investigate the surface, Mars' astrobiology, the planet's interior and finally perform the sample return. There are many configurations which are still being developed and tested. However, JAXA will acquire the final rover from a private contractor, and the software and navigation algorithms are being developed by JAXA's institutes.

1.3 Motivation

1.3.1 JAXA's Space Exploration Ambitions

On 1 October 2003, JAXA was formed joining the Japan's Institute of Space and Astronautical Science (or ISAS), the National Aerospace Laboratory of Japan (NAL) and National Space Development Agency of Japan (NASDA). The plans for JAXA's further space exploration move toward effectuating smaller and faster missions from 2010 onwards. This is due to the fact that planning interplanetary research missions can take long time, like up to seven years for the failed ASTRO-E satellite. Due to the delay between these interplanetary events and mission planning time, opportunities to gain new knowledge about the cosmos might be lost.

In 2009/10 a study was organised to propose a concrete plan for robotic lunar exploration for science and utilization, foreseeing manned lunar exploration afterwards. The suggested approach is to perform by 2015, the first lunar landing and short-term investigation (SELENE-2) and by 2020, assembly of the base, long-term investigation and sample return (SELENE-X). Human Lunar Exploration with landing on and ascent from Moon would be possible later. Also, a human habitation module on Moon has been foreseen. This lunar base is planned to be done in collaboration with other national space agencies within the International Space Exploration Coordination Group (ISECG), after 2020. The JAXA's rover development planned timeline and a conceptual lunar base drawing and are shown in appendix C (the images are courtesy of JAXA).

Concurrently, the Mars mission MELOS is being developed. The tasks are similar to the lunar mission, of course, adapted to the different planetary conditions. To send a lander and the rover to a planet greatly increases the amount of data that can be gathered, which is a lot more and of different nature than that obtained by the orbiter solely. The scientific mission with a small lightweight rover has been proposed, whose main objective is related to the search for methane oxidizing microbes on the Mars surface, followed by a dust monitoring for understanding of the Mars climatological characteristics, and a geological survey with a Laser Induced Breakdown Spectroscopy (LIBS) and/or spectroscopic cameras.

A study discussing the necessities and requirements of such a Mars mission employing a light-weight rover, has been done to determine the feasibility of such a concept [10].

1.3.2 JAXA Rovers

JAXA plans to introduce rovers whose purpose would be to provide a safer environment, reduction of crew work time, efficient task performance, etc. Therefore, helping the astronaut with transfer and carrying loads, execute dull and dangerous tasks on his behalf and assist him by monitoring or manual support. In figure 1.6, from left to right, JAXA's manned transport rover, construction machine vehicle and the assistant robot concepts are presented, respectively.

FIGURE 1.6: The JAXA rover concepts: the transport, construction and assistant robots (images courtesy of JAXA)

The configuration for the lunar rover is the Micro 6 rover, as shown in figure 1.7.

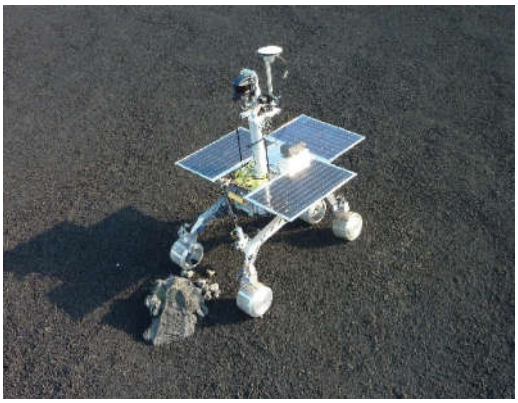


FIGURE 1.7: The JAXA's Micro 6 rover (images courtesy of JAXA)

For the Mars mission, JAXA intends to diverge from the recent trends, and deploy a light-weight rover, which would lead to a cost effective mission. Moreover, an autonomous rover needs to have the capability of long-range travel. However, currently, the Mars exploration rover within the MELOS program, is still under development and many conceptualisations exist, which are being tested.

Therefore, these kinds of missions, both SELENE-2 and MELOS, require an apt autonomous exploration vehicle, capable of running even for longer time periods without direct human control. Besides this, it needs to perform other secondary scientific tasks, at specific locations and under other constraints, in order to gather new information about the planet.

1.4 Problem statement

Considering that MELOS is a mission set for Mars, a significant amount of autonomy is expected. The optimal solution would be for all the computation to be done on board the rover, and in real-time. This consists of image acquisition and processing, to understand the environment. Also, the navigation algorithm application for the given inputs with control output generation and movement execution. In order to process the inputs as they arrive and give outputs so they can be executed in real-time, high processing power is needed and therefore a good power supply. However, the policy of JAXA and the Japanese government in general, is to never utilize nuclear materials as a power source in space applications. This, and the preference of a lightweight vehicle, yields a constraint on the power consumption and thus also on the processing capabilities. There are many algorithms which can determine a global path to the goal, but they need large area scans which take long time to process and might not be very accurate, since the information about the terrain far away is of a lower resolution. Therefore, one problem to be overcome, is to develop an efficient path re-planning algorithm, which corrects the approximate global path, based on the local terrain data gathered on the way to the goal and the previously explored terrain data. This algorithm should output motion commands so that the rover can avoid unforeseen obstacles and unnecessary path sections in order to optimize the overall path if possible.

Another issue to be addressed is the handling of the acquired terrain data. This consists of both the processing of this data as it is introduced by the imaging system and the techniques of storing it. The rover system should be able to extract local terrain information from the acquired images, and using it together with the information from other sensors, such as the robot's absolute pose within the inertial reference frame, define where is that terrain patch located and memorize it. The system should be able to form a global map of the already explored area. This would be beneficial for two reasons; a map of the explored planet which is much more detailed that the one produced by the orbiter would be obtained, and secondly, this information could be used in further exploration for finding optimal paths (if some back-tracing is better than going forward).

To summarize, these are the two main challenging tasks for an exploration rover to successfully negotiate the unknown environment. Others are processing soil, atmosphere and other samples, and communicating the results to the Earth base etc. To approach both of these tasks, a method needs to be developed, based on the requirements and constraints of the system and built up on the previous work in literature and current implementations.

1.5 Objective

Addressing the challenges of modern exploration rovers, a novel navigation method is proposed by using fuzzy logic to interpret the input data and based on that, produce the outputs as commands. The inputs to the fuzzy controller are the spatial terrain properties and the goal/ sub-goal direction. Terrain properties depict the state of the ground, the height and the distance from the untraversable obstacles and also the terrain roughness.

1.6 Thesis Outline

In the remainder of the thesis, in Chapter 2, different contemporary approaches to rover navigation and solutions to the connected issues are addressed and presented, to give a general overview of the state of the art.

Following this, in Chapter 3, a solution to the presented problem, which this thesis deals with, is presented and elaborated.

Further, in Chapter 4, the experimental test results are given and analysed.

Finally, the thesis is concluded and summarized, mentioning some of the possible future developments, in Chapter 5.

Chapter 2

Rover Navigation

2.1 Types of Control

There are several ways in which the ground crew can interact with the rover on another planet. This mainly depends on the type of task at hand, its complexity, and the environment. Considering these factors, one can distinguish three types of control: teleoperation or remote control, fully autonomous and semi-autonomous or supervisory control. The last type can be further classified as discussed in [11]. If the rover pilot has enough perception information of the robots environment, the direct link to the rover system and has to perform a demanding task in which experienced decisions need to be made, the preferable choice would be direct control of the rover. There exist various set-ups for this kind of control (monitor-mouse-keyboard, dedicated ergonomic systems, virtual reality interfaces...). Concerning the situations in which it is impossible to gain reliable control of the robot once it has been launched, or if this is not important, since achieving the goal does not require it, one can revert to creating a fully autonomous robotic system without human guidance. This robot needs to gain information about its environment and also learn or gain new capabilities like adjusting strategies for accomplishing its tasks or adapting to the changing surroundings. The gained results or information are supplied to the humans. A semi-autonomous system refers to a rover system which is guided by a ground team, but not constantly. The human supervision and intervention consists of only giving sub-tasks, which the robot then executes on its own and notifies the supervisor upon its completion and provides the results. After that, the cycle is repeated.

Concerning interplanetary rover missions, several factors affect the final design of the rover system. Firstly, the perception capabilities of the robot cannot be completely trustworthy and the environment is unpredictable and some situations simply cannot be foreseen and taken into account. However, due to the vast distances between the ground center and the rover and the limited travel speed of the signal (light speed), a delay between sending and receiving instructions does occur. With regard to the current space missions, the signal delay between Earth and the Moon is approximately 2 seconds (1 in reception and 1 in transmission). Nevertheless, these rovers can still be driven in real-time. Conversely, the rovers sent to Mars experience a much greater lag in control signal. It ranges, depending on the position of Mars relative to Earth, from 8 minutes to even 30 minutes. The biggest problem is when Mars goes behind the Sun, which then requires a halt of all communication which might last up to a couple of weeks.

On the present Mars rovers, a semi-autonomous paradigm is implemented. A set of tasks to be executed is prepared and given at the beginning of each sol. At the end of the sol, the rover reports back about its achievements and gained information, which is used to plan the activities for the following sol. The Moon rovers used in the 70s were remote controlled. However, if the robot is to travel long distances and reach the poles or the far side of the Moon, the communication will be difficult. The same fact applies to Mars also. This issue can be resolved by allowing the rover to have some autonomy.

2.2 Terrain Representation

In order to successfully navigate through unknown territories, a rover must be able to perceive the environment in which it is in, and based on that to plan its route to the goal. Besides the sole path planning process, the map representation and data structures are very important. The map representation can make a huge difference in the performance and path quality and it has to be compatible with the available processing power, memory and other constraints.
Some of the used map representations are presented in the following.

Grids

A grid map [12] (shown in figure 2.1(a)) uses a uniform subdivision of the terrain into small regular shapes. Common grids in use are square, triangular, and hexagonal. Each of these grid elements, or tiles, has its value which represents the traversability of that part of the terrain. Also, occupancy grids have been used [13], where each cell holds a probabilistic estimate of its state, which can be empty or occupied. The size of the tile depends on the application, but for example, on Mars rovers, it is taken so that the wheel could fit in this square. This representation is simple and local changes have only local effects, thus it is well suited for dynamic environments. The center of each tile in the grid, or its edges or vertices, can become a node, leading to a highly connected graph. The movement between these tiles can be perpendicular or diagonal (4- or 8- connected) [14].
The disadvantage of this type of representation is that it is awkward for objects that are not tile-sized and shaped. If an object falls into an even small portion of grid element, the whole element is marked as occupied.

Quadtrees

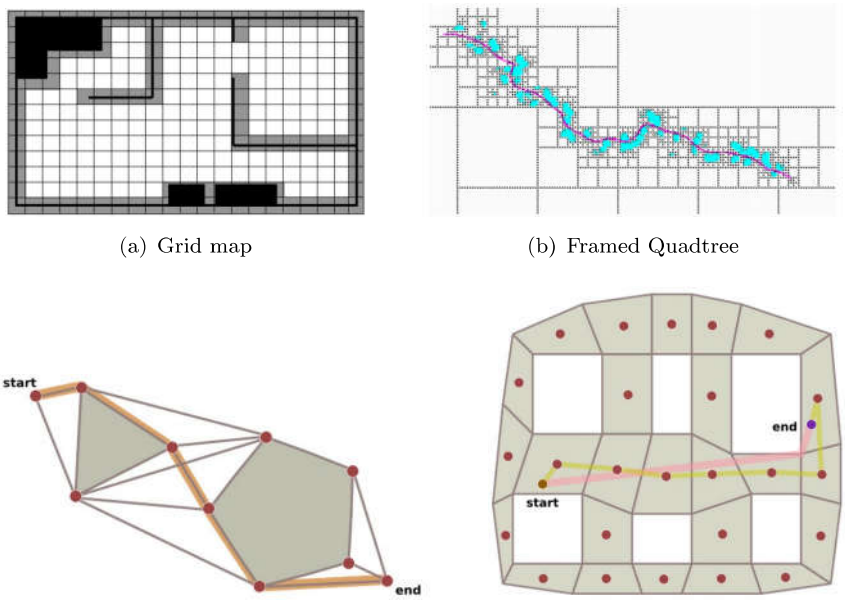
The problem with regular grids is that if the obstacle enters a grid element even a little, the whole element has to be marked as an obstacle, which leads to a lot of useful space being lost. One way to avoid this is to refine the grid size, but this will increase their number and with it the computational cost.
Thus, the quadtree approach is useful because with them, the map is first divided into large grid elements and when an object falls into a part of some grid element, that element is subdivided into four smaller grids and so on recursively (figure 2.1(b)). Therefore, the overall map will not have evenly sized cells, a large clear area with one single cell and smaller cells will be used near irregular shaped obstacles.
Another data structure, framed quadtree, has been proposed as a method to overcome some of the sub-optimality issues related to the use of quadtrees [15].
Using these map representation provides efficiency in memory and computational resources [16].

Visibility Graphs

The most common alternative to grid maps is the polygonal representation. If the movement cost across large areas is uniform, and if the rover can move in straight lines instead of following a grid, a non-grid representation is desirable. As in the example shown in figure 2.1(c), the shortest path will be between corners of the obstacles. So those corners are chosen as the key navigation points. A way to identify which of these nodes are connected, is to build a visibility graph [17]. This graph consists of lines connecting the pairs of points with a visible connection between them, not passing through any obstacle. The path found by the visibility graph will be the actual shortest path. There is no grid resolution on which precision of the path depends.
In some maps with lots of open areas or long corridors, a problem with visibility graphs becomes apparent. A major disadvantage of connecting every pair of obstacle corners is that if there are N corners, vertices, there are N^2 edges, and this greatly affects the memory usage.

Navigation Meshes

The obstacles do not have to be represented with polygons, alternatively, the traversable areas can be represented with non-overlapping polygons, also called a navigation mesh [18]. Using this representation, the obstacles need not be stored as additional information. Also, it has benefits such as a low number of edges per node, like the grid representation, and a linear number of nodes as with the visibility graph. Similarly with a grid, the polygon centers, edges, or vertices can be selected as navigation points. A navigational mesh can be made out of polygons of an arbitrary number of sides, as long as they are convex.
A comparison of the path obtained using this method (orange) and the visibility graph (pink) is presented in figure 2.1(d). The visibility graph representation would produce the pink path, which is ideal. Using a navigation mesh makes the map manageable but the path quality suffers.

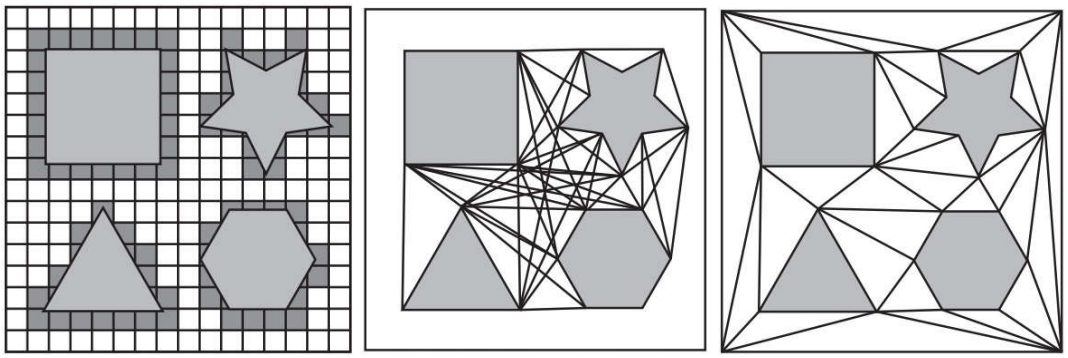


(a) Grid map (b) Framed Quadtree

(c) Visibility graph (d) Mesh grid

FIGURE 2.1: Terrain representations (images taken from [1, 2])

The comparison between some of the mentioned methods can be seen in figure 2.2. For the same map settings, the number of nodes obtained will not be the same for different approaches. Only with the grid representation the number of nodes is independent of the number of obstacles. The number of edges of the navigation mesh is a function of the number of the corners of the obstacles, and for the visibility graphs the complexity depends on the square of the number of corners.



(a) Grid map (b) Visibility graph (c) Navigation Mesh

FIGURE 2.2: Different representations of the same map (images taken from [3])

Which type of terrain representation is employed largely depends on the application. Most of the techniques use grid maps due to their simple implementation. However, if the map is to big it can be a heavy computational burden.

Most of the methods represent the robot as a single point on a map, in configuration space [19], or a circle, so that the orientation does not matter, which in the end greatly simplifies path planning. This reduced representation is possible when the obstacles are increased to compensate for the robots dimensions, and on the map they present the unreachable parts. Moreover, the robots are considered holonomic, although there is much research dealing with path planning in nonholonomic robots. The robots currently deployed on Mars are able to turn in place, thus holonomic, so these approaches can be applied to them.

2.3 Traversability Analysis

The value of each cell, or node, can be determined in different ways, depending on the approach. This defines the effort, or cost, that the robot needs to invest in order to traverse it. Based on this, considered for relevant nodes, the path planning algorithm selects the shortest path towards the goal so that the overall cost of the nodes contained in that path is minimized. The cells can be, for example, defined simply as occupied or free, i.e. there is an obstacle present and the robot cannot go through that cell or not. In this case the robot would have to avoid every cell labelled as an obstacle. However, a more detailed property can be given to the node, as seen later on, such as the goodness or traversability of the node. This approach gives a more descriptive notion of the terrain and gives the rover more manoeuvrability, since depending on the wheel size and the rovers configuration, some obstacles might be small enough so that it can pass over them.

There are several approaches for determining the traversability. It is usually determined by calculating a value directly from the sensor data or by classifying the type of

terrain the vehicle is driving over. The first can be implemented by using a line-scanning laser to scan the area in front of the rover and then by analysing this data, the roughness of the terrain is determined and the traversal speed is managed accordingly [20]. Also, an approach using stereo vision to determine local roughness, as well as the slope of the terrain, can be applied in order to plan paths which avoid rocky and sloped areas [2]. Calculating traversability based on terrain colour was discussed in [21], but this approach greatly depends on the correlation between the colour and the relief of the terrain. Moreover, terrain properties can also be extracted by analysing the images made by the rover, by e.g. calculating disparities, and concentrating on terrain texture and geometry features.

Another idea is using information from stereo vision systems as input to machine learning algorithms which perform terrain classification, which is further used for planning. Also some approaches relate the information from satellite images with local classifications to improve long distance traversability estimates. One such example was carried out for the DARPA challenge [22], where a neural network based classifier was used to classify terrain as traversable or not. This type of terrain assessment relies on training data and outputs from expert systems to provide the traversability information. The efficiency can be increased by adding some supervision which determines the accuracy of the classification process. For this purpose proprioceptive sensors can be used which give information about energy consumption, tilting, vibration, wheel slipping etc. One such implementation was developed in [23] where rover-terrain interaction was modeled based on vibration sensing, which is performed by a sensing microphone fixed to the rover's suspension. Another method, using the Inertial Measurement Unit (IMU) is elaborated in [9]. This implementation of proprioceptive sensors allows the rover to avoid areas which can cause damage to the robot and its equipment.

Therefore, the most reliable approach would be having some internal sensors to asses the classification process and thus making the traversability analysis more robust. After, the path planning algorithm will have a good starting point to determine the optimal path to the goal.

2.4 Approaches to Path Planning

The aim of the path planning methods is to generate the trajectory for a robot to execute in order to reach the goal location. Planning can also be done under constraints, such as obstacles, stability, some optimization criteria etc. so that the robot can find the best possible path under the given circumstances. Most of the algorithms used are the search-based algorithms. The potential field approach [24] is efficient, it combines attraction to the goal, and repulsion from obstacles, but its drawback is that it can get stuck in local minima. Sampling-based algorithms avoid the problem of local minima, they give a complete solution if the sampling is done infinitely long. A discussion of their performance is presented in [25]. Moreover, the algorithms can be distinguished based on the environment in which they are performing. The environment can be static, when the map does not change with time and there is enough time for the robot to plan the path. Conversely, a much more realistic case represents the dynamic environment, which is mostly unknown or its map is incomplete, thus a robot gains new knowledge about the surroundings and must efficiently re-plan the path. On planetary missions, the surroundings are unknown, but also resources are limited, thus an algorithm which

satisfies these condition must be implemented.
Some of the used algorithms are shortly described in the following:

A*

The A* algorithm is a graph search algorithm [26] , which finds a least-cost path from a given initial node to one goal node. This is done by determining the cost of each node f, to know if it will be visited, as the sum of the path-cost function, which is the cost from the starting node to the current node g, and the heuristic estimate of the distance from the current node to the goal h. This heuristics should always be lower than or equal to the real cost of moving from the current location to the goal in order not to overestimate it and to be able to find the optimal path. The final cost is then as in 2.1.

$$f = g + h \tag{2.1}$$

The open list is created and there the non visited nodes are kept. When the algorithm begins, the initial node is placed on the open list. From there other nodes from the start are expanded and put on the open list as well. Now, the already visited nodes are put on the closed list, which collects all the already expanded nodes. In this way, it does not return into a previously explored node and thus can extract itself from local minima. Using these two lists provides more selectivity about what is examined next in the search. The A* pseudocode is displayed in appendix B.1. This is one of the basic grid search algorithms which finds the shortest path. Its main drawback that it works only in static environment and does not perform replanning along the way needed in dynamic environments.

D*

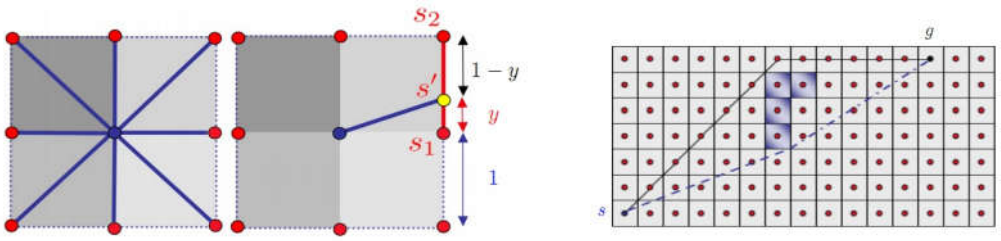
Dynamic A* or D* search algorithm is a dynamic, on-line variant of A* [**?**]. D* starts with a map containing all known, assumed, and estimated mobility cost data for the environment. Assumptions are made about unknown part of the terrain. An initial path is planned from the rover's starting location to the goal. As the robot moves along the path, its sensors discover new information and discrepancies between the initial map and environment (such as previously unknown obstacles). The map is then updated, and if necessary, the rover's path is re-planned from its current location to the goal. This process repeats until the rover reaches the goal or it determines that it cannot be reached. When traversing unknown terrain, new obstacles may be discovered frequently, so this replanning needs to be fast.
For large environments, D* has been shown to be hundreds of times faster than re-planning from scratch using A*, because it minimizes the computational cost of replanning by recalculating the navigation function only where necessary. Current systems typically implement D* Lite rather than the original D* or Focused D*, D* Lite is based on Lifelong Planning A* [27]. The pseudocode of D* Lite is given in appendix B.2. It is more efficient, simple, it uses only one tie breaking criteria when comparing priorities and it employs the same navigation strategy as D* but algorithmically different.

Field D*

Searching the grid graph typically finds suboptimal paths since the transition between nodes is constrained to the predefined directions in the multiples of 45 degrees. The Field algorithm D* [5] brings a novelty in path planning by generating paths of continuous

headings to eliminate unnecessary steering, and using the interpolation method to determine the cost of traversal of each grid element (depicted in figure 2.3(a)). And the new cost is then calculated as in 2.2.

$$cost(s') \approx y * cost(s_2) + (1 - y) * cost(s_1) \tag{2.2}$$



(a) Interpolation method (b) Optimal path

FIGURE 2.3: The representation of the interpolation performed by the Field D* and the final obtained path compared to a suboptimal one. (images taken from [4, 5])

Implementing continuous heading transitions a path like the one in 2.3(b) can be acquired. However, Field D* is constrained to movements along graph edges. The algorithm's pseudocode is given in appendix B.3.

E*

The E* algorithm [6] is a generic path planning method that combines dynamic replanning capabilities with path cost interpolation. In order to avoid the discrete path selection problem related to grid methods, the concept of E* is formulated in the continuous domain independently of any specific configuration representation. It regards a wavefront propagating from the goal throughout the environment as depicted in figure 2.4. Then, by varying the propagation speed in function of environment, the traversability of certain regions, the crossing-time map is generated that has steeper gradients around the obstacles, and the gradient descent can be done to form the trajectory towards the goal.
The wavefront represents the range of all the possible locations the robot can arrive to starting from the goal location and moving at the maximum speed, and it is always expanding. Therefore, by tracing back the propagation direction from a certain location, the region of influence at thet point can be defined. And then, if there have been some changes at that location, it is possible do determine which areas of the crossing-time map have stayed the same and which need to be re-calculated, to provide efficient re-planning. Basically, the wavefront it is a queue of cells which are yet to be expanded, and it spreads until the this list becomes empty or the cell containing the robot becomes expanded.

The environment in which the rover is evolving is represented as a grid, made up of cells c, which have the properties, $v(c)$ which represent the "height" of the navigation function of the cell and $r(c)$ which is the risk or cost of traversing the given cell. This parameter corresponds to the wavefront speed.
The queue key is calculated as the minimum between the $v(c)$ and one-step lookahead of the crossing time called rhs-value, $ths(c)$. Then, the propagation is directed to the state where all nodes have these two values equal, or they are locally consistent.
To perform the interpolation, an interpolation kernel is used, which is a function that determines the crossing-time value of a node, based on its traversal risk and the surrounding

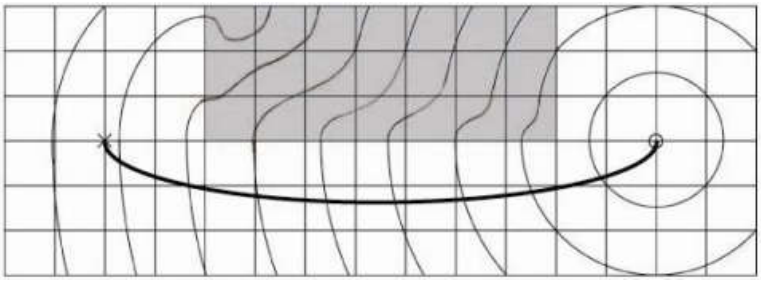


FIGURE 2.4: Wavefront propagation with E*, from the goal (o) to the robot (x). The Trace proposed by E* is defined using the gradient descent after the wavefront propagation. (image taken from [6])

node values.
The pseudocode of the E* algorithm is given in the appendix B.4. The algorithms achieves to obtain smooth navigation functions that approximate true distance much better than other grid methods but this is at a cost of an increased computational complexity.

Thus, to summarize, Like A*, D* plans an initial path from the robot's start state to the goal state using all available information, known or assumed. However, with D*, as the robot follows this path and discovers discrepancies between the map and world. The map is updated, possibly leading to the re-planning of the path, by computing, saving, and re-using partial solutions to the planning problem. The major drawback of the grid search methods, the non-smoothness and suboptimality of the path due to the discrete transitions available between the nodes is evaded with the interpolation based algorithms such as Field D* and E*. The former found its application in outdoor field robots but also on the MER mission rover Opportunity as explained in the following section. The E* algorithm was implemented also to outdoor robots, but also to autonomous vehicles [28].

2.5 Navigation Methods

Since the planet where the rovers are sent is unexplored, a map of its terrain is inherently unknown. Apart from the images of the ground obtained by the orbiting satellites, no other information can be gained. Usually, these images do not have sufficient resolution to detect all the hazards so that the path-planning algorithms could rely on them. This said, a need for developing other techniques for attaining needed information is developed.

2.5.1 MER's AutoNav System

The autonomous navigation with hazard avoidance (AutoNav) system [4] of the MER, is based on the GESTALT (Grid-based Estimation of Surface Traversability Applied to Local Terrain) path-planning algorithm discussed in [7, 29]. It takes pictures of the nearby terrain using one of the stereo camera pairs (body-mounted hazard-avoidance cameras on Spirit, mast-mounted navigation cameras on Opportunity) and gathers geometric information about the surrounding terrain. These stereo images are processed and 3D terrain maps are generated automatically by the rover software to create a model of a local terrain. Traversability and safety is then determined from the height and density of rocks or

steps, excessive tilts and roughness of the terrain. The map is divided in cells, where each cell has its goodness/cost value. High goodness values indicate easily traversable terrain, and low goodness values indicate hazardous areas and they are computed within the field of vision. Once the terrain has been evaluated dozens of possible paths are considered before the rover chooses the shortest, safest path toward the programmed geographical goal. These paths are evaluated based on the criteria which consider avoiding hazards, minimizing steering time and reaching the goal. These concepts are depicted in figure 2.5. After selecting the path, the rover then drives between 0.5 and 2 meters toward its goal, depending on how many obstacles are nearby. The whole process repeats until it either reaches its goal, or is commanded to stop. Another improvement over Sojourner is the Mars Exploration Rover Visual Odometry software system. Since the operational environment consists of sand and rock segments, unpredictable slipping of the wheels might occur. The notion behind the Visual Odometry system is to provide the rover with additional information of how far it has actually travelled so it can cancel out inaccuracies due to slipping. The system compares the images taken before and after a short drive, automatically finding features in the terrain (rocks, rover tracks and hills) and tracking their motion between images to obtain the distance.

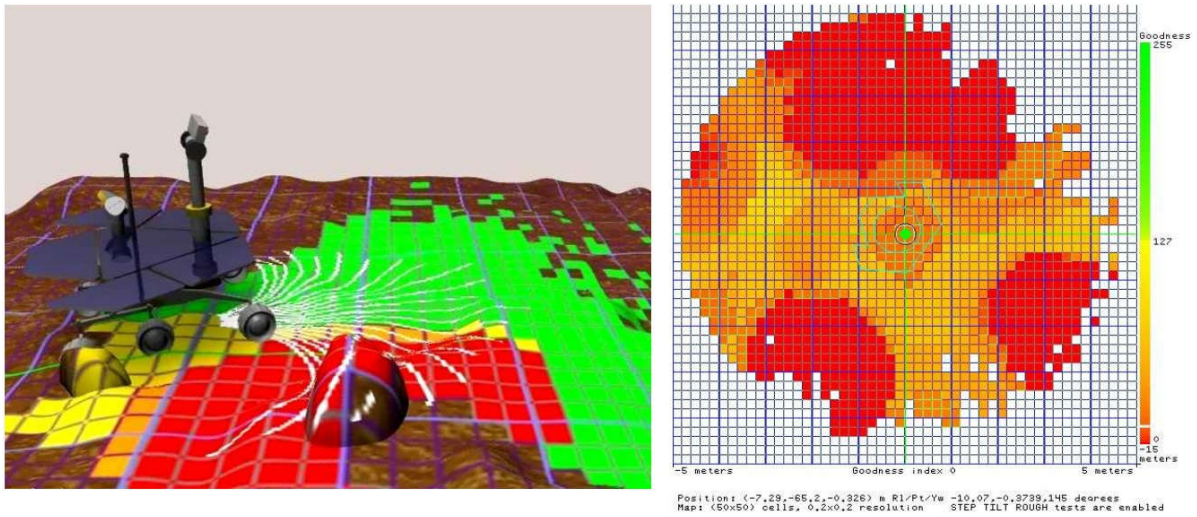


FIGURE 2.5: Terrain assessment and path selection (left). An example of a Goodness map (right) (green/yellow/orange indicate traversable and red unsafe areas).
(images taken from [7])

However, in some situations AutoNav is not able to reach the goal. This can happen when a rover reaches an extended obstacle formed by a cluster of closely spaced rocks. An example of this occurred on sol 108, when Spirit was unable to autonomously navigate to the goal and spent around 105 minutes and performed 47 drive steps trying to overcome the obstacles. This happens because when large obstacles are encountered, the hazard avoidance votes and shortest distance votes given to a path, based on which the selection is made, come into conflict. The hazard avoidance votes will not allow the rover to drive through the unsafe area, and the waypoint votes will not allow enough deviation from the straight-line path for the rover to get around the hazard. The rover then becomes stuck and is unable to reach the goal.

An update to the system adding the Field D* algorithm was updated to Opportunity in 2006. This was intended to improve the performance of selecting the waypoints, as the Euclidean distance, as used sometimes by A*. More precise paths can be obtained

by including the available information about the obstacles since this provides better estimates of the environment. The integration of this algorithm lies within making the goodness maps generated by the AutoNav compatible with Field D* CostMap. Moreover, the optimal path outputted by the Field D* in form of the traversal costs needs to be converted in arc votes which AutoNav uses.
The CostMap is similar to the goodnes map, but the goodnas map is centered around the rover's current location and processes only local terrain information, while the is fixed to the environment and does not follow the rover. Additionally, the "goodness" and "cost" of grid cells are opposite, higher goodness means lower cost and vice versa. Even though some goodness map elements can have unknown values, this cannot happen with the CostMap, thus the corresponding cells have to be initialized to some neutral value. The rest of the conversion procedure is straightforward if the goodness map and CostMap cells are the same size and are aligned, but these should designed at the beginning to be compatible. This transformation can be viewed in figure 2.6.
The second part of the integration refers to obtaining the arc votes. The traversal costs of the arcs increase directly as the starting distance from the goal is bigger, simply because of the amount of cells between the start and goal position. The important information is the relative traversal costs between the arcs, and of course the arc with the lowest cost is the best one. The numerical vote values are obtained using the weighted sum of v_{scale} and v_{close}, where the former is the scaling factor of the cost values into vote values and the latter bases vote values upon the distance of the rover to the goal. These values are calculated for each candidate arc following equations 2.3 and 2.4. Here, v_{max} is the maximal vote that can be given, c_{min} and c_{max} are the minimal and maximal traversal costs of the arc set at hand and c_i is the traversal cost of the arc for which the votes are being calculated.

$$v_{scale_i} = v_{max} * (c_{max} - c_i)/(c_{max} - c_{min}) \tag{2.3}$$

$$v_{close_i} = v_{max} * c_{min}/c_i \tag{2.4}$$

Basically, the notion behind these calculations is logical. When the rover is far from the goal, all arc votes will be close to v_{max}, while when it is close, the vote values will range from 0 to v_{max}. Together with v_{scale}, v_{close} sets the scope of the votes, depending on their distance to the goal. Therefore, when the rover is distant it is not very important which arc is selected exactly if they provide the correct direction. More useful in this case would be other votes taken into account, like hazard avoidance and steering bias which influence the final selection. While the rover approaches the goal, the precise selection of arcs becomes crucial and a broader vote span is given. When the final arc votes have been obtained, they are used instead of the GESTALT waypoint votes and combined with the steering bias and the hazard avoidance votes as mentioned earlier. This combination also resolves eventual ties which might have occurred amongst the arc votes. Additional optimizations are then performed regarding the map size, in order to accommodate the available resources such as computational power, available memory, communication bandwidth with the Earth station, etc.
During both the ground Surface System Test Bed testing and the Mars surface tests, this update performed well and was chosen for everyday use [30].

A similar system employing vison motion estimation should be implemented in the ExoMars rover [31]. The cameras main configuration should be the same (mast mounted and body mounted in front) yet their purpose might be different. Usually the mast camera is used to obtain the image of the environment and the lower camera for hazard

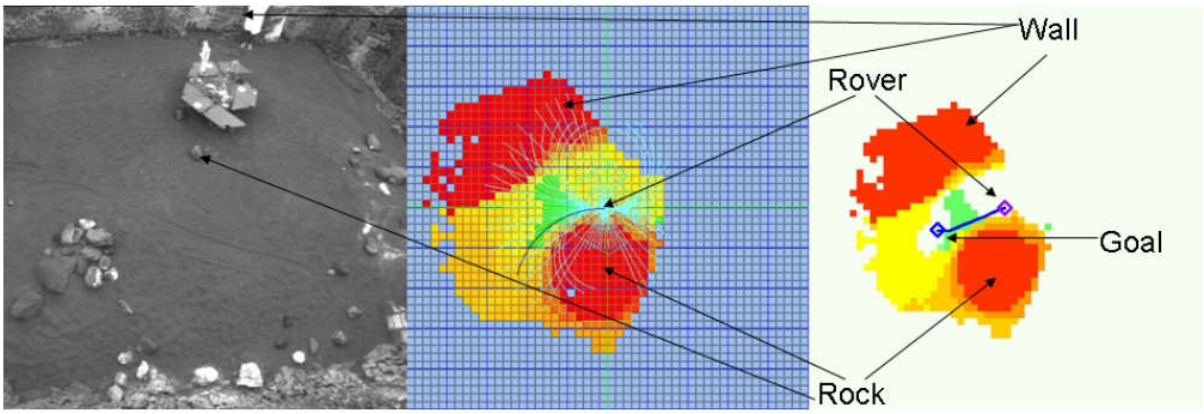


FIGURE 2.6: Surface System Test Bed site view (left), the corresponding goodness map (center) and the Field D* CostMap (right). (images taken from [4])

avoidance, but this can be inverted. Also the grid representation might be different due to the variance in the dimensions of the rover.

2.5.2 A Layered Approach

Another interesting strategy for robot navigation is a layered approach using fuzzy logic [8]. The idea is that the first layer of the planner uses the knowledge about the global goal together with the information from the long range sensors to produce the waypoints, intermediate goals, which guide the rover towards this goal. The second layer drives the robot towards these waypoints, but uses short range sensor data to avoid hazards along the way. Using only the fuzzy algorithms for navigation would produce a solely reactive approach and would cause a so-called shortsighted behaviour, which yields not optimal paths. Therefore, the first layer is needed to give global information about the goal. The representation of the robot and the notation used is presented in figure 2.7, where (x_i,y_i,θ_i) is its current position and orientation, and the indexes change dependent on the position as 0, w and g, for initial, waypoint and goal, respectively.

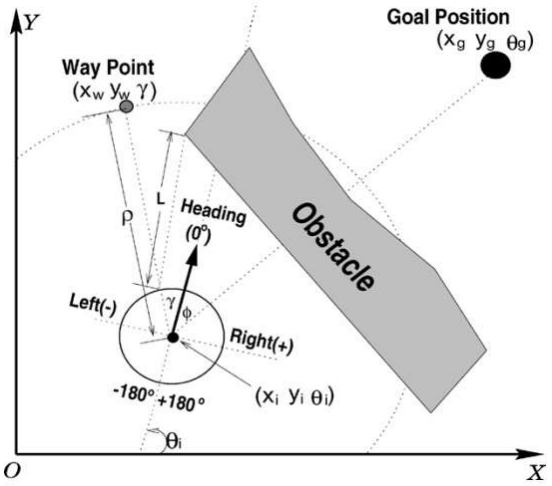


FIGURE 2.7: Representation of the robot and the desired goal with an obstacle on the way. (image taken from [8])

The first layer is comprised of three modules whose purpose is to find the traversable area, search for the global goal and to fuse the commands in order to output the orientation towards the waypoint, γ, with respect to the robots heading direction. In order for these modules to perform well, long range sensors, such as laser range finders or sonars are needed. The goal seeking module takes the target angle ϕ, between the heading and the goal, as input, and produces the desired area (in direction of the goal). Both the traversable and desired area are fuzzy sets upon which a fuzzy conjunction is performed by the command fusion module which yields γ, as mentioned before. Knowing this, also the position of the waypoint (x_w,y_w) and the orientation θ_w (of the robot at the waypoint) can be calculated as in 2.5, where ρ is the distance to the waypoint.

$$\begin{aligned} x_w &= x_i + \rho cos(\theta_i - \gamma) \\ y_w &= y_i + \rho sin(\theta_i - \gamma) \\ \theta_w &= \theta_i - \gamma \end{aligned} \tag{2.5}$$

More specifically, the rule base in the first traversability module consists of eight fuzzy sets with given labels (as shown on the left of figure 2.8). The membership functions used are of trapezoidal or triangular shape, since the piecewise linear functions are not a computational burden. When the i^{th} sensor of the rover denoted by s_i, whose position makes the angle α with the heading direction, has a reading, the untraversable area within the region monitored by this sensor, τ_i, is obtained as a composed fuzzy set. As shown in the upper right part of figure 2.8, the τ_i is obtained by firing two adjacent rules with strengths μ_1 and μ_2, based on the input angle α (sensor position angle). Formally, it can be written as in equation 2.6.

$$\tau_i = \mu_1 \oplus \mu_2 = min\{1, \mu_1 + \mu_2\} \tag{2.6}$$

The t-conorm operator $\oplus$ depicts the overlapping of the fuzzy sets. This is done the area monitored by s_i is mostly obstructed, so these directions are joined. Applying this rule to all the sensors, the total untraversable area is attained by combining all untraversability information gathered by each of the sensors, calculated as in 2.7. Here, n is the total number of sensors taking readings, and the operator max is used because the strongest reading is the one which should be considered most seriously.

$$\Gamma = not \bigcup_{i=1}^{n} \{\tau_i\} = 1 - \max_{i=1}^{n} \{\tau_i\} \tag{2.7}$$

The goal seeking module has five fuzzy sets evenly distributed in order to have the broad span of flexibility, so that each direction has the same probability of being the desired direction. Knowing the orientation of the goal position with respect to the heading, ϕ, the desired area can be obtained by combining the fired fuzzy sets, as it can be seen in the lower right of figure 2.8. This can be written as in 2.8, where μ_1 and μ_2 are again the weights associated to the fired sensors.

$$\Omega = \mu_1 \vee \mu_2 = sum\{\mu_1, \mu_2\} \tag{2.8}$$

Finally, the waypoint direction is calculated as in 2.9, but this will be a fuzzy set with multiple peaks to which a defuzzification strategy has to be applied to obtain a usable γ.

$$\tilde{\gamma} = \Gamma \wedge \Omega = min\{\Gamma, \Omega\} \tag{2.9}$$

There are several methods to accomplish defuzzification. The proposed strategy is using Mean of Maximum of Largest Area which is a combination of the Mean of Maximum and the Centroid of Largest Area defuzzification methods. After obtaining γ, as said before, the position of the waypoint, together with the orientation θ_w can be calculated as in 2.5 and used as the subgoal in the second layer of the planner.

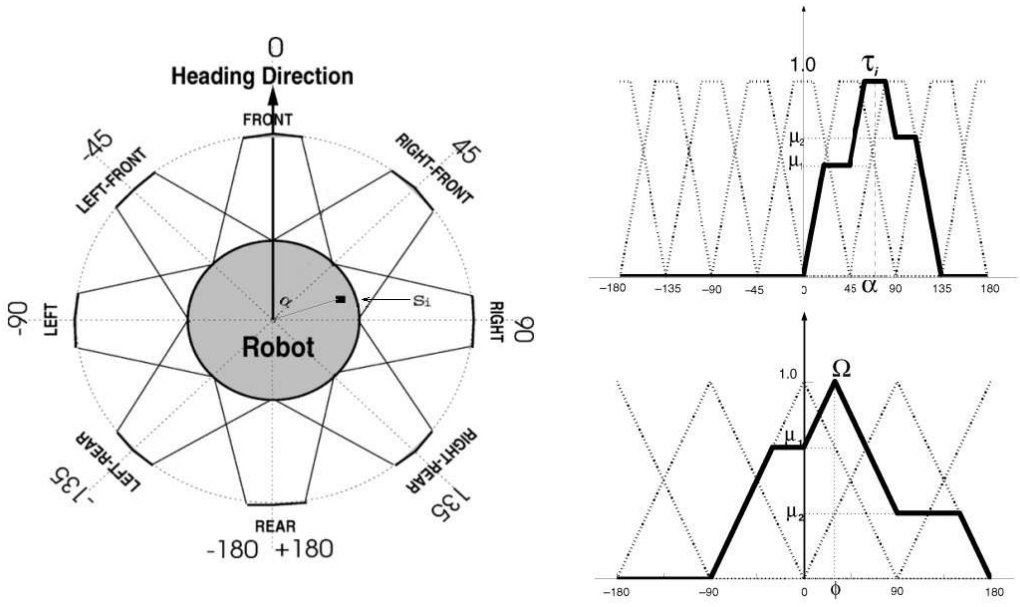


FIGURE 2.8: Rule base (directions) for traversability finding module (left). Non-traversable area identification based on sensor readings (upper right). Goal direction estimation based n readings (lower right). (images taken from [8])

The second layer, as said before, guides the robot towards the subgoal avoiding obstacles. For this it utilises short range sensors implementing a pure reactive fuzzy navigation algorithms such as direction-based (taking into account obstacle and goal positions to generate final steering direction) and speed-based (considering obstacle repulsion and goal attraction fixes the motors' speed). The three modules constituting the second layer, similar as previously, serve for avoiding the obstacles, searching for the subgoals and fusing the information to generate steering commands. The procedure is similar to the first layer, the fired sensor indicates that the corresponding direction has to be avoided, and the subgoal seeking is done similarly as before, but now the membership function calculated in 2.8 has a more concentrated set of desired directions.
Another difference is that not all subgoals need to be reached. One example is when the robot gets stuck. Then, the first layer is recalled after the previously set deadline has been reached to generate a new subgoal. This deadline is calculated in 2.10, considering the robot's base, maximal, speed v_b and the distance from the subgoal ρ. Therefore, when the deadline has expired and the new subgoal generated, the rover will continue towards it.

$$T = \frac{\rho}{v_b} \tag{2.10}$$

Alternatively, the subgoal is not reached if on the way towards it, the long range sensors, which are rechecking the perceptive region frequently enough, discover a more convenient path to the global goal or the direct path to it.

One important situation to be considered, is when the robot goes into a deadlock. This occurs, for example, when the robot comes across a wide wall perpendicular to its direction towards the (sub)goal. Here it is preferable that the rover is able to follow the wall in order to exit this situation. The anti-deadlock mechanism is based on the angles of the fired sensors which detect the wall, and utilizes one of these angles is set as the temporary target angle. In this way, the first layer will output a satisfactory waypoint which leads the robot out of the deadlock.

This method poses no previous assumptions about the environment. Also, using sensors the robot can navigate through an unknown and dynamic environment successfully avoiding obstacles. However, due to the sensor constraints and bad readings, the robot can wander off from the desired path. This method was tested in an indoor environment with an elementary sensor system, but applying appropriate sophisticated sensors would greatly improve its performance.

2.5.3 RTILE - Adaptive rover navigation based on online terrain learning

Another interesting concept is allowing the rover to gain and use the knowledge about the unknown terrain it is traversing, especially the interaction wheel-soil, based on its interaction with the terrains encountered. Moreover, it would be also able to classify the terrain. To realize this concept, a framework Rover Terrain Interaction Learned from Experiments (RTILE) was developed and tested on the CRAB rover [9] [32]. The implementation is based on using long distance sensors, such as cameras, to generate the remote terrain perception (RTP) model, and local sensors, such as the Inertial Measurement Unit for example, to gain the rover-terrain interaction model. This learning is done without supervision and automatically and creates the mentioned Bayesian models. This model is then used to make the prediction of the rover-terrain interaction based on the acquired knowledge, of the terrain not crossed yet (figure 2.9). The E* algorithm is used to perform the path planning and drive the robot to the goal. The cost functions used are updated based on the predicted terrain interaction, which in this way influences the trajectory. The pseudocode of the RTILE approach is given in appendix B.5.

The approach was tested in an experiment where the rover dynamically learned three terrain classes (grass, asphalt and tartan). Based on learned characteristics, it was able to predict the terrain and performed trajectories which caused much lower vibrations, but had a little longer length. The important thing is that the learning process was unsupervised, without previous knowledge of the terrain, thus it could be implemented in space exploration as well.

Of the three presented systems, only the first one has been implemented on a rover actively exploring another planet. However, even though the others have not been tested in such a way, the approach would be valid regardless of the environment, since they are designed for the unknown. The most important issue with them is the accuracy of the sensors. The layered approach needs accurate short range sensors for obstacle avoidance, but also the long range sensors to obtain the proper perception of the environment where the goal is located. Also, the RTILE approach also depends on the accurate readings of

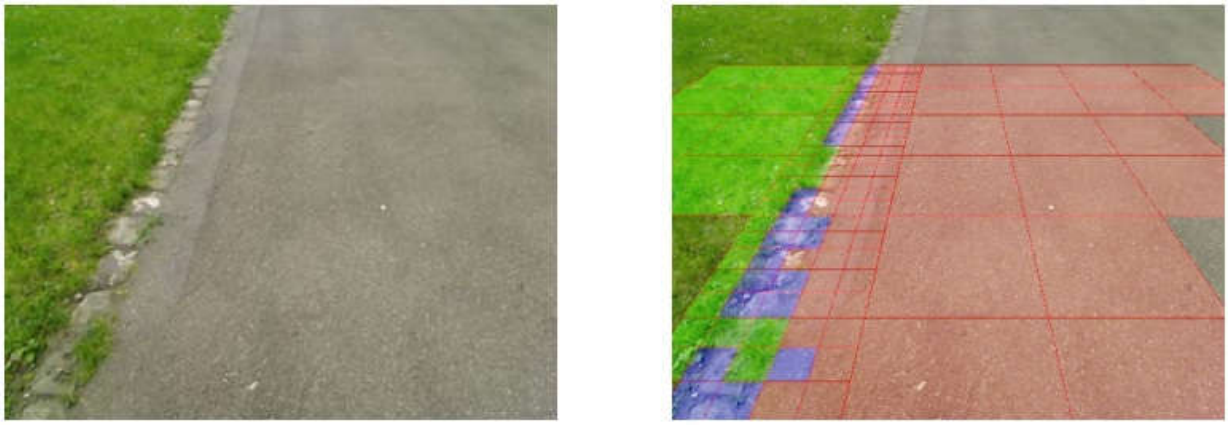


FIGURE 2.9: The original image of the terrain (left) and its cell decomposition and terrain interaction prediction (right). (images taken from [9])

the IMU. The AutoNav system determines the traversability analysing the reconstructed 3D map obtained by the stereo camera pair, but its performance would be improved, if it had some proprioceptive sensor information processing to make the terrain conception more robust and accurate.

Chapter 3

Proposed Solution and Realization

3.1 Proposed System Overview

Addressing the challenges of modern exploration rovers, as presented in the previous chapters, a novel navigation method has been proposed. The problem to be solved consists of two parts which are mutually interconnected.

Firstly, it is necessary to have a system which will be capable of keeping track of its position in the new and unknown environment and saving the already explored area as a knowledge base and material that can be reused if necessary. This global map would comprise of multiple local maps taken by the rover at certain locations it explored and then put together in the appropriate way. All of these processes are done autonomously. As the system acquires a local view image, it is used together with the readings of the Roll, Pitch and Yaw angles from the inertial measurement unit (IMU) and the odometry information for the position and orientation to determine where, within the global reference frame, has this local view been taken. Then, this view, after the necessary transformations, is assigned to its according position. This process is referred to as "map patching" and presents the use of local map information to form the global map. In the next iteration, this global map is used, and the new view is appended to it. However, to preserve it efficiently, the viewable area is divided into tiles whose size depends on the rovers wheel size, which is considered as the appropriate resolution.

Regarding the second part of the problem, the navigation, the approach is to use fuzzy logic to interpret the input data and based on that, produce the outputs as commands for the rover motion. The inputs to the fuzzy controller are the spatial terrain properties and the goal/ sub-goal direction. Terrain properties depict the state of the ground, the height and the distance from the untraversable obstacles and also the terrain roughness. The definition of untraversable is that all such obstacles, whose height exceeds the half of the wheels diameter, are considered such that the rover cannot go over them with the wheel. All other obstacles along a single direction around the rover take part in forming a roughness value along that direction. Using the roughness information to form a more accurate and meaningful representation of the terrain is introduced and is one of the main contributions of this thesis. These terrain properties are based on the data obtained by the vision system and the previously known terrain information (explored area map). Another contribution of this thesis is to achieve a real-time path re-planning functionality, since the sub-goals given to the fuzzy controller are derived from the rough

estimate of the global path to the final goal, which might be inaccurate, since it has been generated from a great distance. To optimize the path, it has to be re-planned along the way, as new local terrain information is gathered from the sensor. This information is gathered locally, because the rover's camera is considered to take images of the vicinity in front of the rover at a specified frequency. The local goals, for now, are considered to be known, and derived from a rough estimate of the global path to the final goal.

Following the extraction of the outputs for the controller - the desired speed or displacement and the heading direction of the rover, these commands can be executed. After accomplishing the desired motion, the process is repeated until the sub-goals, and afterwards, the global goal has been meet. Upon reaching the global goal, a new rough path estimate with the sub-goals is generated. This process takes more time, but that is why it is not repeated very often. The overview of this system, as a closed loop system, is presented in figure 3.1.

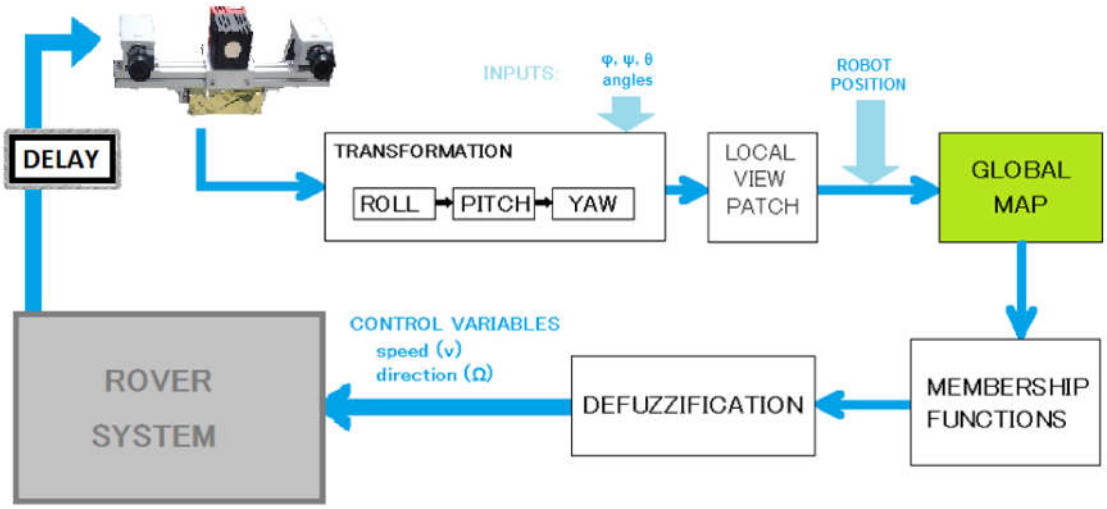


FIGURE 3.1: The proposed system scheme

To sum up, as the chart suggests, the information needed to add the current view to the global map information, are the IMU's position and orientation readings and the local 3D data itself. The local patch is then assigned to the global map, which can be used to extract the membership functions. The following defuzzification process, does the inverse of the fuzzification and gives exact values for the control variables used to give movement commands to the rover.

More information about each segment of the system will be given in more detail in the following sections.

3.2 Reading and Interpreting Image Information

The algorithm itself is made independent of the specific type of camera used to capture the environment information. However, the terrain data is transformed into a discrete 2D elevation grid map based on the 3D environment information. The size of each of the grid tiles is a specifiable variable, which is set to be the size of the wheel diameter. The reason for this dimension is that it can be assumed that this is the area for which one value

describing the terrain properties can be given, without losing much on the reliability and precision. Besides, that is the area that portrays approximately the footprint and defines whether the wheel can be placed and moved at that location, or not.

During the testing phase, the Microsoft Kinect camera was used. Kinect is a motion sensing input device by Microsoft for the Xbox 360 video game console and Windows PCs. The Kinect sensor is a horizontal bar connected to a small base with a motorized pivot and is designed to be positioned lengthwise above or below the video display. The device features an RGB camera, depth sensor and multi-array microphone. The depth sensor consists of an infra-red laser projector combined with a monochrome CMOS sensor, which captures video data in 3D under any ambient light conditions. To successfully utilize the Kinect sensor on a desktop PC running Windows7 OS, additional drivers had to be installed. Supplementary libraries also needed to be installed, such as OpenCV (Open Computer Vision) OpenNI (Open Natural Interaction) and OpenGL (Open Graphics Library), whose functionalities have been used for image recognition, 3D information extraction and graphical representations, respectively.

3.2.1 Obtaining the Local View

The information which is acquired about the terrain is the most important. With some minor modifications, it can be extracted from any RGBD or TOF device if their output is a 3D point cloud. To successfully form the 2D grid map, information about the height at each of the observed point is needed. Thus, besides the visual information, also depth perception is necessary. This work builds up on previous work, which dealt with the extraction of the terrain data from the camera feed, thus, this data is already assumed to be known, and that a point height cloud is received as the input.

Based on the 3D information provided by the sensor, the local digital elevation map can be formed. This 2D grid map consists of tiles of specified dimension, where each of them contains the height information for that particular area it covers. The height information for a particular tile can be defined in several ways depending on the desired precision needed. In this case, the physical area that one tile represents is the area which is occupied by one wheel of the rover (25x25cm). Using appropriate program variables this size can be modified to a rectangular size of arbitrary dimensions as well. Depending on the chosen tile dimension, as well as the current local view range (in length and width), the dimensions of the grid map, i.e. the number of tiles, are calculated by dividing both of the maximum absolute view dimensions by the appropriate tile dimension and then rounding the result to get an integer number. The grid map tile values are initialized to "NON" which is equal to -100, as the value which assumably cannot occur, to differentiate unpopulated tiles from the populated ones when assigning values.

From here, there are several possibilities of how to populate the 2D grid map based on the input 3D data. By dividing the area into wheel-sized tiles, naturally, much of the initial precision is being discarded, thus an intelligent way of keeping track of the important traits has to be developed. The simplest approach is to simply make the average of all the points from the original terrain information which would fall into the according tile. However, this approach would not give a realistic representation, because it might happen that within one tile, which is e.g. 25x25cm, there is a cylindrical obstacle, which is high but narrow. What would occur then, is that these couple of high points measured by the camera, would be evened out by the numerous low height points which

also fall within the same tile. This might lead to false negatives, and the rover might run into an untraversable obstacle while thinking there is only a slight bump in the ground.

One way to avoid this problem and represent the real terrain properties, has been proposed through equation 3.1.

$$value = \sqrt{\frac{1}{N}\sum_{i=0}^{N}(0 - x_i)^2} = \sqrt{\frac{1}{N}\sum_{i=0}^{N}x_i^2} \tag{3.1}$$

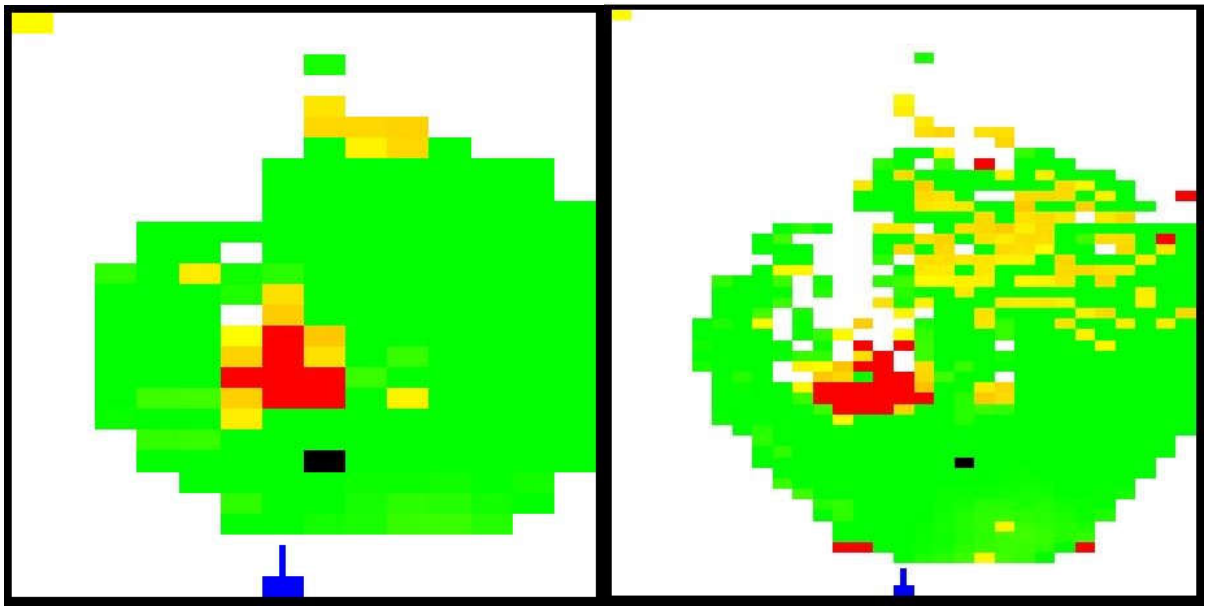
This equation represents a modified standard deviation centred around the 0. It is a function which tends to extract the extreme height values from the batch of height information points x_i. The value which is obtained in this way will be the maximum or the minimum value (depending on the sign) but it will disregard possible noise, or false readings, which could have unrealistic extreme values. To resolve the issue of the standard deviation value always being positive, the final value is given the sign of the average, of the maximum and minimum value of the height of the points within the tile, but within the obtained height value range of *+/- value*. In this way, if the absolute value of the minimum height, is greater than the positive extreme value inside this range, the average will shift to the negative side's favour, and consequently the negative sign will be designated to the final standard deviation value. In the case that both of these extreme values are equal, the average would be 0. Then, it is equivalent which value will be assigned (positive or negative) since the obstacle will have the same intensity, the sole difference being whether it is a hole or an extrusion. By default, it is assigned a positive sign.

Having repeated this calculation for each tile of the grid, the process of defining the local view map has been concluded. From here the patch can be created for further global map patching and also the obstacle membership functions can be calculated.

3.2.2 Displaying with OpenGL

In order to provide visual feedback for the 3D to 2D information conversion and the local grid map generation process, the OpenGL library functions are used to display the final obtained grid map (figure 3.2). A color gradient, ranging from green, through yellow to red, indicates graphically the height of the terrain. The green color represents totally flat surfaces, and the color turns towards yellow and red with the increasing roughness, until it reaches a predefined threshold, at which the obstacle is considered untraversable and the color from then on is solid red. The threshold is defined as half of the wheel's height, but it is a parameter which can be changed depending on the situation at hand. The black part of the window is the area not acquired by the sensor. The white area represents the tiles for which there is no height information, meaning, their values are NON. Also, since the input view is conic-like and the output view is a square, there will be much more white tiles, which are just added as additional to widen the view. Nevertheless, these values will not interfere when actual height for these locations is assigned. To additionally facilitate the post analysis, the robot and the goal are shown on the map. The goal is assigned a black color and the robot is blue, with a thin blue line representing its heading direction.

In this way, the user can visually validate whether the displayed image is an appropriate discretization of the test field's local view.



(a) Tile size 0.22 m (b) Tile size 0.11 m

FIGURE 3.2: Displaying the local map with the robot position and orientation (blue) and goal position (black)

3.3 Mapping

The mapping process is very useful because it allows the generation of the explored area map as the robot is exploring it in real-time. Beside the fact that it gives a more detailed insight of the planets surface, it also provides valuable information when a wider area re-planning needs to be done. However, for the rover to use this map for navigation and for it to be suitable for storing, its resolution should be decreased as with the local map. The obvious approach is to make the global map tiles and those of the local map, identical size. In this way, as the robot traverses, the local views can be patched along the appropriate empty tiles. Still, the global map grid has a unique alignment, that is, all the grid lines are parallel to the two principal axes. This creates a problem when the robot rotates around its vertical axis, i.e. when its Yaw angle is not a multiple of $\pi/2$. This has to be taken into account when merging the local map within the global map at a particular position.

Moreover, some of the fields within the global map will have to be assigned more than once, e.g. if the rover is reacquiring them in the local view. In this situation, a certain importance should be given to the old data, so that it is not overwritten by new data totally. Likewise, it should not be completely opposite that the new information is disregarded. This is because when the rover approaches a point which was viewed from a distance before, it will have more precise information about it and that is more credible.

Having stated all of this, the global map making procedure can be divided into acquiring the local map, creating the appropriate patch for that location which is aligned to the global grid map, placing it at a proper location and finally, assigning the patch values to the according tiles.

The perpetual problem within the field of planetary exploration is to balance out the navigation algorithm and map precision with the available processing power. With the development of more capable processors and larger rovers, the lever separating these two sides becomes smaller and the solutions produced are becoming better. One of the difficulties was to determine the optimal tile size, which is a relatively ill-defined problem, and it had to be done through testing.

3.3.1 Roll, Pitch and Yaw Transformations

The robot is assumed to have 5 DOF, two position coordinates and the 3 rotational angles - Roll, Pitch and Yaw. The axis going from the intersection of the rover's camera mast and the front wheel axle, out in front is the y axis, the lateral one, intersecting the y axis and going along the wheel axle, is the x axis, with the positive direction going to the robot's right side. Finally, the z axis is pointing upward. The Roll axis is defined as the rotation around the y, Pitch about x axis and Yaw about the z axis.

While the robot negotiates rough terrain, the position of the camera can oscillate due to the attitude change. Using the IMU, these inclinations can be observed and accounted for during the mapping process and attaching the local views at a particular offset from the robot's position. Also, the camera's height and position shifts that a particular attitude change yields are considered. The effects of these attitude changes and the appropriate transformations to compensate them, are elaborated further. When the 2D elevation map of the local view has been obtained, these transformations can be applied. The first are the Roll and Pitch transformation, and then the local view, after these two transformations, is applied the Yaw transformation, which also defines the new patch size.

All of these transformations, besides modifying the final local view and thus patch size, also modify the positions of some of the tiles containing the height information. Each of these transformations gives its contribution to the final output arrangement.

Roll

When the rover encounters a rise or a declination of the ground, whose gradient is perpendicular to the heading direction, its attitude will change and this will affect the way the image system perceives the environment. Furthermore, this change, if not taken into account, would result in an error in local view positioning within the global map (Figure 3.3).

Concerning roll motion, the offset occurs in height, because due to the rotation the height of the camera changes, becomes lower, and it might see some terrain higher than it really is. This inaccuracy can be calculated knowing the distance between the wheel axle and the camera (CAM_TO_WHEEL) through the equation 3.2. Besides this type of height offset, the one due to the image rotation is taken into account. This means that when the rover inclines, for example, to the left, the local view will show the terrain on the left as much higher than it really is, and the terrain on the right as lower or even negative. The following equation 3.3 models these height offsets. The transformation of the points from the local view to the patch needed for mapping is presented in equation 3.4. The $view\_width$ and $view\_height$ variables represent the local view dimensions.

$$RoffHeight = CAM\_TO\_WHEEL * (1 - cos(ROLL)) \tag{3.2}$$

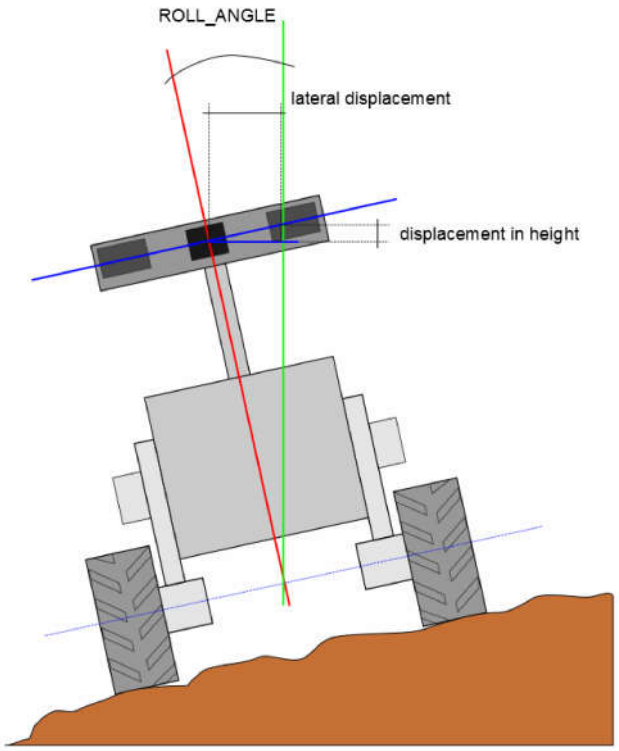


FIGURE 3.3: Rover on a lateral slope

$$height\_offset[u][v] = \left(v - \frac{view\_width}{2}\right) * sin(ROLL) \tag{3.3}$$

$$\begin{bmatrix} ph \\ pw \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(ROLL) & \frac{view\_width}{2}(1 - cos(ROLL)) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} vh \\ vw \\ 1 \end{bmatrix} + \begin{bmatrix} pv0h \\ pv0w \\ 0 \end{bmatrix} \tag{3.4}$$

Moreover, the effective useful local view will become more narrow (Figure 3.4), since the view will be transformed.

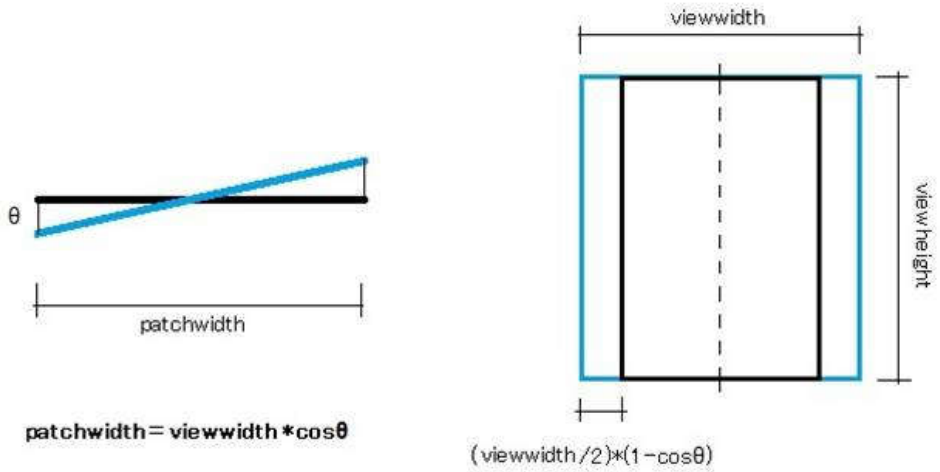


FIGURE 3.4: The change in the local view size due to the Roll transformation

Another important error would be the change of the rover's position along the axis perpendicular to the robot's heading. This occurs due to the camera's inclination to the left or right side. These offsets can be estimated using equation 3.5 and knowing the distance between the camera's pole axis and the wheel plane (WH_REL_POS) and the distance from the camera's center to the wheel axle. Then, these values are added to the robot's position to compensate for the shifts, and transformed to the global coordinates.

$$offH = WH\_REL\_POS + CAM\_TO\_WHEEL * sin(ROLL) \tag{3.5}$$

Pitch

Similar to the Roll motion, offsets due to the Pitch angles have to be taken into account and the appropriate values need to be estimated to compensate for the offsets in height and position. Contrary to the previous case, here, the position offset due to the pitch would be along the rover's heading axis, since the Pitch motion represents the inclination of the rover and thus the camera, towards its front and the rear (Figure 3.5).

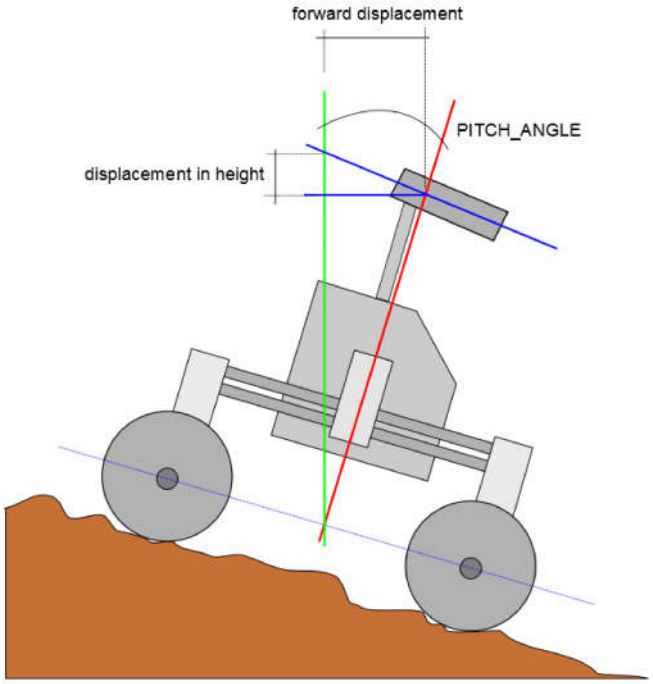


FIGURE 3.5: Rover on a frontal slope

The transformation of the position of each point from the original view to the final patch can be calculated based on the Pitch angle, using the equation 3.6. After finding the offset, of course, it needs to be converted back to the global reference frame coordinates. This type of inclination also alters the cameras height so that is also taken into account in equation 3.7. As well as the change in the lateral coordinate 3.8, knowing the height of the camera's center from the wheel axle.

$$\begin{bmatrix} ph \\ pw \\ 1 \end{bmatrix} = \begin{bmatrix} cos(PITCH) & 0 & \frac{view\_height}{2}(1 - cos(PITCH)) \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} vh \\ vw \\ 1 \end{bmatrix} + \begin{bmatrix} pv0h \\ pv0w \\ 0 \end{bmatrix} \tag{3.6}$$

$$RoffHeight = CAM\_TO\_WHEEL * (1 - cos(PITCH)) \tag{3.7}$$

$$offH = CAM\_TO\_WHEEL * sin(angle) \tag{3.8}$$

$$height\_offset[u][v] = (view\_height - u) * sin(PITCH) \tag{3.9}$$

However, the image data can also become distorted depending on the inclination in this direction. Therefore, the proper transformation compensating for the change in terrain height information is executed, as in equation 3.9. As shown in the figure 3.5, if the rover is inclined towards the front, the terrain appears closer than it really is, and conversely, towards the rear it will appear shorter. this also causes the final view to shrink, because not all information will be useful (Figure 3.6)

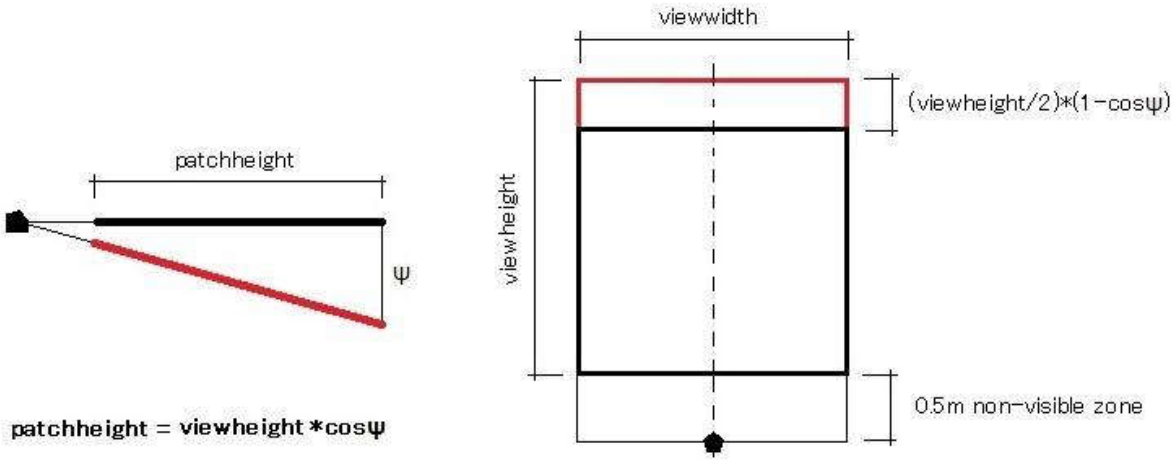


FIGURE 3.6: The change in the local view size due to the Pitch transformation

Camera Pitch

In order to have a better perception of the local surrounding area in front of the rover, the camera should be inclined downwards. Of course, it is very important to make a proper compensation for the camera orientation, regarding the offset of the terrain height values. This is done simply by applying a rotation in the opposite direction by multiplying the 3D coordinate with the appropriate homogeneous transformation matrix, and then taking the z coordinate of the point. It is similar to the normal Pitch compensation, save for the lack of the translation, since the axis of rotation goes through the camera itself.

Yaw

Lastly, the information about the Yaw angle is useful to determine the final size of the patch to be added to the global map. The mapping process is conceived in such a way that the new local view being added has to be aligned with the global grid. Every time when the Yaw angle is such that this view is not aligned, there is a problem with assigning proper values in the appropriate locations, as it will be discussed in the following subsection. In order to discretize the conic-shape local view to a square-shaped map patch, the original view will have to be widened so that all tiles carrying height information are accounted for. This original view is basically the terrain information before being put in the patch, and after the Roll and Pitch transformations have been done. Meaning, the width will be the absolute range of information along the x axis, and the length will be the absolute

range of information along the y axis. These original views are shown in green in figure 3.7, and their dimensions are vH and vW, meaning view height and width, respectively. Similarly, the final patch dimensions are pH and pW, which stands for the patch height and patch width, respectively.

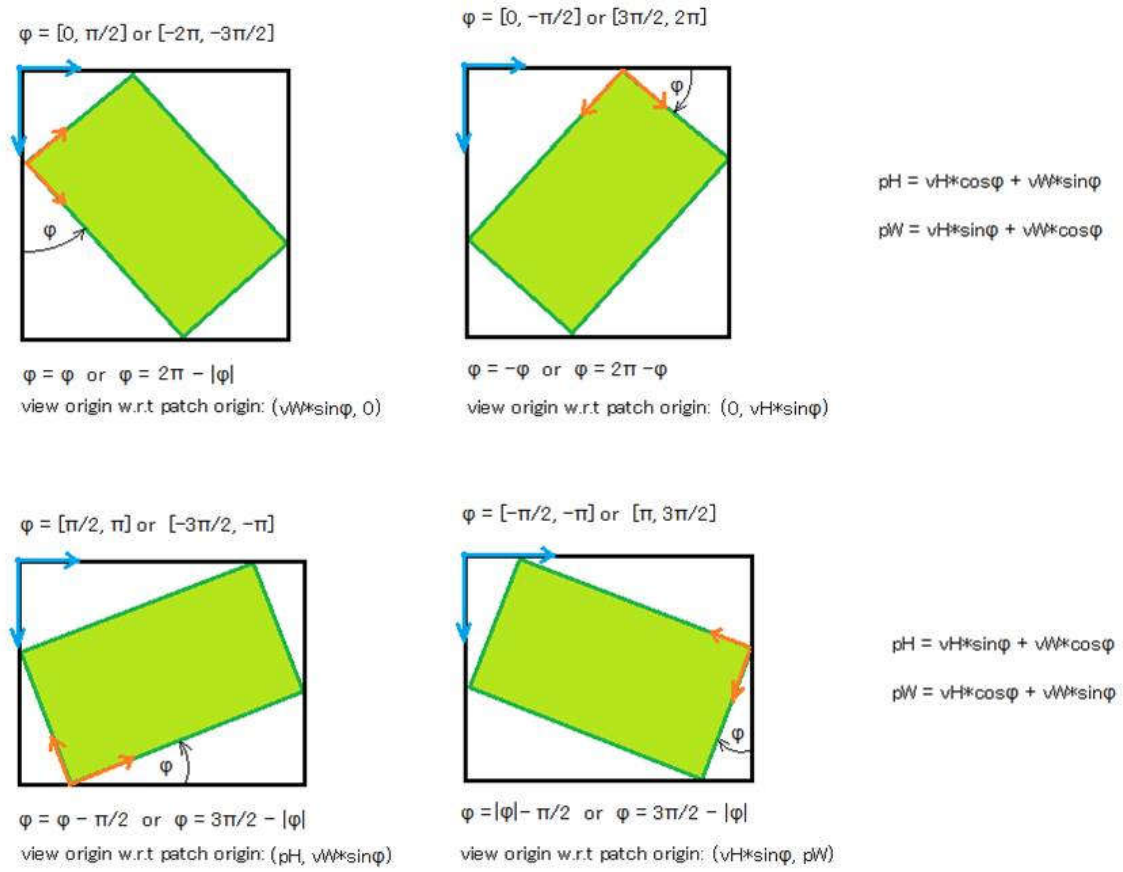


FIGURE 3.7: The local view formation based on the Yaw angle

Depending on the Yaw angle, the position of the rover with respect to the patch origin, can be determined, which is necessary when defining the location within the global map, where the new local patch will be added. The dimensions of the patch depend on the dimensions of the original view and the Yaw angle, as it can be seen in figure 3.7.

$$\begin{bmatrix} ph \\ pw \\ 1 \end{bmatrix} = \begin{bmatrix} cos(YAW) & -sin(YAW) & pv0h \\ sin(YAW) & cos(YAW) & pv0w \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} vh \\ vw \\ 1 \end{bmatrix} \tag{3.10}$$

The Yaw transformation also needs to transfer the values from their original positions in the local view, to new patch coordinates obtained after rotation. The transformation is given in equation 3.10. It determines the new local coordinates to which the values will be assigned.

3.3.2 Populating the Local Map

In the previously mentioned transformations, the tile coordinates of the local 2D elevation maps have been transformed so their values are assigned to new locations within the patch. The purpose of the map population step is to transfer the values from the original positions to the new, transformed ones. The main challenge here, is the problem of grid alignment. As seen in figure 3.8, some of the tiles are, after the rotation transformation, naturally, not aligned to their destination tiles. Some of them are more covered and some are less. In order to define a method which recognizes these differences and implements a meaningful assignment, the following solution is proposed. For each point in the final patch, a value is defined based on the values from the tiles of the transformed view (in red in figure 3.8), but only from those in a predefined range (e.g. 0.707) as shown in equation 3.11. In this way, the tiles which logically have the influence on the value of the particular tile, are limited spatially. Depending on their distance from the destination tile, their values are accordingly given more or less importance using a weight factor. These weights are simply inversely proportional to their distance, which gives more importance to the values closest to the destination tile.

$$values(i)(j) = \frac{1}{\sum_{k=0}^{N}(0.707 - dist(k))} \sum_{k=0}^{N} [height(k) \cdot (0.707 - dist(k))] \tag{3.11}$$

This process is depicted following figure, where on the right side is the close-up of the example grid map. Here, a comparison of two ranges for value consideration, defined by the diameters of 0.707 and 1 tile size, is shown.

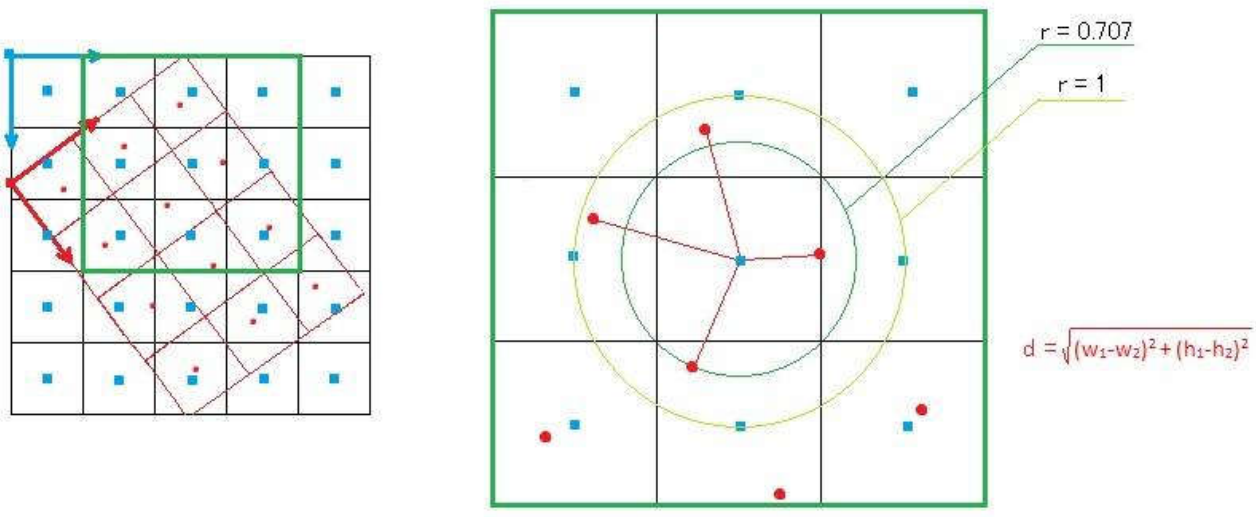


FIGURE 3.8: Method of grid tile value assignment

3.3.3 Global Map Patching

The global map is nothing more than a matrix, whose coordinates, modified, describe the rovers absolute position with respect to the global reference frame, which in fact is the rover's touch down point from where it started its mission. The field values of these matrix depict the height information of that tile sized terrain segment. Knowing this, it is not straightforward to determine the position of the goal, rover and where to place the new local view patch. To facilitate this, two new variables are added - $OrigH$ and $OrigW$, which take into account, respectively, the vertical and horizontal position of the (0,0) global map coordinate origin with respect to the matrix' (0,0) origin (H stands for height and W for width). By adding the robot's absolute global position to these variables (equation 3.12), the result will be the robot's position with respect to the matrix origin, which is necessary for map matrix manipulation tasks. The minus at the H coordinate is because the matrix coordinate axes are pointing downwards and to the right. These directions are also taken as the global reference frame, but with the coordinate origin being shifted to ($OrigH$, $OrigW$). This means that the global h axis is pointing downwards and w axis, to the right. The yaw angle is measured from the h axis, and the positive direction is the positive mathematical direction (counter-clockwise).

$$\begin{aligned} mapRpos(H) &= OrigH - globRpos(H) \\ mapRpos(W) &= OrigW + globRpos(W) \end{aligned} \tag{3.12}$$

During the creation of the local view patch, within the Yaw transformation, the position of the patch origin as well as the rover's relative position are saved. Based on these values, and knowing the rover's absolute position within the global map matrix, the simple insertion of the patch can be achieved.

One issue that needs to be considered, is the situation when there are conflicting tiles. Meaning, new information has to be written, but there is old information at that particular tile. As discussed previously, in section 3.2.1, when this occurs, it is best to keep both of the values but at different intensities. To provide this, weighted averaging is implemented, with a variable ratio coefficient α as used in equation 3.13. The variables $posH$ and $posW$ are the location at which the patch is inserted within the global map.

$$glob\_map(posH+i)(posW+j) = \alpha * glob\_map(posH+i)(posW+j) + (1-\alpha) * patch(i)(j) \tag{3.13}$$

In this way, both the new and the old values are kept, although more importance is given to the new values due to their higher reliability. This is enabled by setting $\alpha = 0.25$.

However, one problem that occurs with the mapping process, is the mismatches and the misreadings due to the discretization error. The whole map population process is based on statistical value distribution, so it is natural that sometimes it does not classify the right value at a proper place. Also, during map forming, it might happen that a tile receives a value which is the standard deviation of the pertaining 3D points, but this values does not model the real terrain properly. All of the above mentioned processes might lead to the wrong value being put at a certain tile, or when the map is being patched, due to the rounding of the robot's position to a tile sized square, a certain tile can be incorrectly assigned.

3.4 Fuzzy Control

The purpose of the fuzzy controller is to produce exact number control variables from a given range of possible values, based on the continuous terrain properties and the goal and rover relative positioning. It employs the fuzzy sets, membership functions, called the goal and the obstacle membership function. The fuzzy sets consist of 360 elements, where each represents the value for a specific direction around the rover. The 0 angle is along the h axis of the global map. The values within a fuzzy set range from 0 to 1.

The goal membership function gives the direction that the robot should follow to go directly to the goal. The obstacle membership function gives an idea of the distribution of the obstacles around the rover in all the directions. The obstacle membership functions consist of the untraversable obstacle membership function and the roughness membership function. Their purpose is to give more detail about the terrain, and not just the two discrete descriptions of the path - traversable and untraversable. Combining these membership functions and processing them in a proper way, the optimal path is extracted that weighs out the shortest way to the goal while avoiding the problematic terrain areas.

3.4.1 Goal Membership Function

As discussed before, this algorithm needs to be fed some goals towards which it re-plans the path. The goals are assumed to pertain to a rough path to the final goal. The amount and the distance between the goals is arbitrary, but it depends on the type of terrain. Generally, it is considered that the rover needs to make a couple of steps to reach one sub-goal, where one step stands for one iteration of obtaining the local view, processing that information and outputting the control variables. It is possible that some of these way-points could be placed on a part of the terrain which is inaccessible to the rover.

The rover takes sub-goals from the list of sub-goals in the order as they have been generated. Still, in order to optimize the process, a mechanism to skip and take the following goal is implemented. This is beneficial if the sub-goal is placed in an unreachable zone, or simply if it is not necessary to visit it. The rover will scan all the sub-goals in a certain range, which has been elected to be dependant on the rovers' dimensions (approximately 1 - 1.2 m). At the beginning of the navigation, the rover will go towards the first generated sub-goal. As it proceeds, before reaching the first sub-goal, the second one that has been generated might come inside the scan range, and then, a higher priority will be assigned to it, and the rover will concentrate towards it. Another security mechanism is regarding the final sub-goal i.e. the global goal. When the rover is bound for the final goal, the maximum velocity it can take will be dynamically changed to adjust to the distance between them, so that the rover cannot overpass the goal.

The sole goal membership function is formed based on the relative angle between the sub-goal position and the rover position (equation 3.14). It is converted to the absolute orientation with respect to the h axis, by adding the Yaw angle value.

$$direction = mod(atan2((RposW - GposW), (GposH - RposH) + 180 - YAW, 360) \quad (3.14)$$

Based on this value, the goal membership function is formed as the a triangular function, with its peak equal to 1 and coinciding to the goal direction angle. The bottom,

whose value is 0, is on the robot's rear. Between these two values there is a gradient descent and rise, and based on that the values for each of the directions are assigned. An example of the goal membership function, with the goal direction set to 0, is shown in the upper image of figure 3.9.

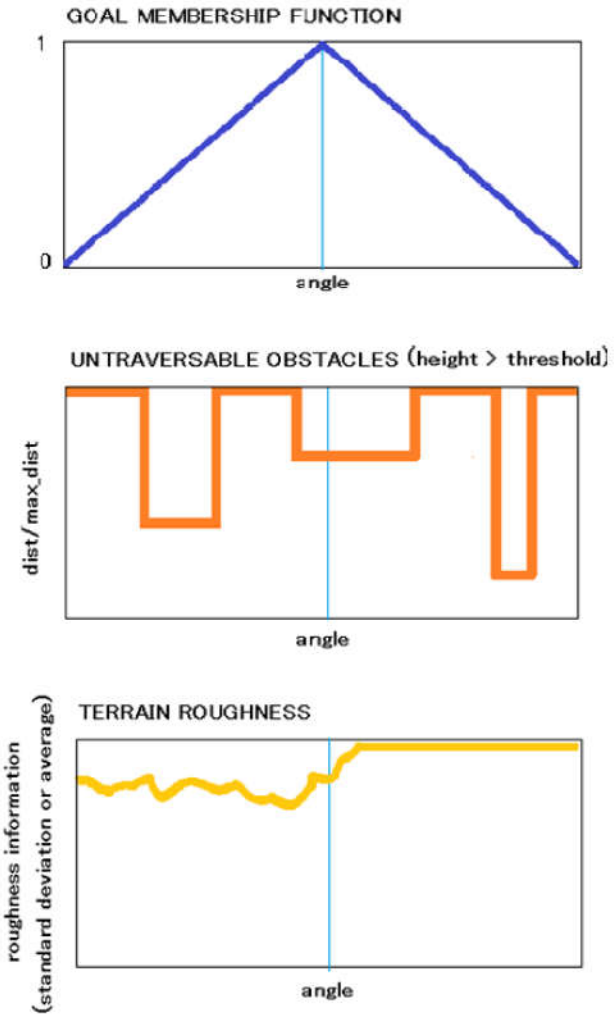


FIGURE 3.9: An example of the membership functions: goal (up), untraversable (middle) and roughness (bottom)

3.4.2 Obstacle Membership Functions

The obstacle membership consists of two parts: the roughness and the untraversable terrain membership function, which are further processed to obtain the final obstacle membership function. Together, they give a more realistic picture of the surrounding area. The main challenge of this approach is how to translate the terrain quality, which is given as a discretized 2D elevation grid map, to a simple number which is associated to a particular direction around the rover. Meaning, in order to form the obstacle membership function and the robot to select the direction to take, each of them has to represent the area behind it with a certain number. This is made easier by splitting it into two

separate membership function. However, to form each of them, an iterative approach is implemented.

Both the untraversable and roughness membership functions give the values for each direction around the rover in the range of 360 degrees. The direction which is processed is obtained by iteratively going through the desired region of the map, determined by *ScanRange* for both directions. For each point, the relative direction with respect to the robot's heading is determined as well as the Euclidean distance. Having determined to which direction it belongs, the characteristics of that point in the terrain (tile values) are processed. Depending on its height, that particular field is classified as either traversable or untraversable. If traversable, is is appended to the temporary 360 element array holding the values of all terrain points, tiles, which fall on the same, specific direction. It is possible that different directions hold various numbers of points. The process of estimating these values is further explained in the following sections.

Regarding the region from which the points are processed, as mentioned, its dimensions are 2x*ScanRange* by 2x*ScanRange*. The *ScanRange* variable is set as 2 m, although as the rover approaches the goal, if the goal comes within the 2 m, it is reasonable to shrink the range, making it consider only the terrain between the rover and the goal. Therefore, the *ScanRange* is defined so that its value is basically the minimum value of the current distance to goal and 2 m. Of course, if the scan area is to exceed the map, it is limited to the map edge and the existing values.

Another subject to consider when processing the obstacle information, is the rover's dimensions. The approach used is to make two obstacle membership functions, which consist of the untraversable and roughness membership functions, for the left and the right frontal wheel separately, taking into account their positions. These two are merged using the logical product. In this way, the final output obstacle membership function, will represent the spatial obstacle data in a way so that the rover could traverse only if there is enough space for both wheels, i.e. the whole rover. This gives the effect of increasing the obstacles, which is usually done using other methods (like, for example, in [33]). When this is done, the point tile information where the robot center and the opposite wheel is, is not taken into account.

3.4.2.1 Untraversable Obstacles

Untraversable obstacles are all obstacles whose height exceeds a predefined threshold. This is due to the fact that it is not desirable for the rover to even try to go over them, because of the high probability of failure, but also actuator and wheel damage. Moreover, it might cause the rover to get stuck and force an unnecessary high current drain (unless it is limited) which could also negatively affect the power system. Since the terrain used to form the membership functions is discretized to wheel-size tiles, the minimum size one of these obstacles can have, is exactly that tile size. Selecting the tiles and their pertaining direction is done as explained previously. Assigning the value for each of the directions is done similarly as in [8], since here the attention is only directed on the untraversable obstacles. However, processing the information of each of the tile is done using the map information, and not the distance information directly. The contribution of each untraversable tile is simply with its distance from the rover, because the degree of untraversability is irrelevant. To be more specific, its distance from the rover, divided by the maximum distance ($ScanRange\sqrt{2}$), to normalize it to 1. Each tile pertaining to a

particular direction is simply added to the membership function array at the designated position. If another closer tile falls into the same direction, its value will overwrite the previous one, since the untraversable obstacles closer to the rover are more important.

An example of the untraversable obstacle membership function is depicted in figure 3.9 (middle).

One practical issue when implementing this method, is overcoming the low terrain data resolution. This demonstrates itself when the rover is located in a tile which is very close, or right next to a a tile being processed. If this tile is, for example, on the robots left side, the span of almost 180° on its left side (which is the orientations of 180° - 360°) of its obstacle membership function array will be unknown. Only the 90° direction will be known based on the value of the tile next to it. This is particularly problematic when dealing with untraversable obstacles, since it can induce an enormous information loss. One way to solve this is to implement a linear filter, and after an additional low-pass filter whose size is inversely proportional to the distance from the observed untraversable obstacle.

3.4.2.2 Terrain Roughness

Representing the terrain roughness along one direction, based on the discretized 2D elevation map, is an interesting task. This is one of the main novelties proposed by this thesis, i.e. to evaluate the potential directions based not only on a certain threshold, but also on the roughness gradient. Firstly, in order to utilize this notion, the definition of roughness should be set, in order to extract the quality needed. There are, of course, several methods to approach this issue. The implemented one is based on the discussion done in [34], where the most effective way to represent the terrain roughness is using the standard deviation. In particular, along one direction, the standard deviation of all the values of all the tiles that fall into that direction is considered. These tile values are not simply height information, but also weighed by the distance and normalized. The value that one tile ($value(i)(j)$) contributes to its pertaining direction array ($rough(angle)$) at the postion (k), is shown in equation 3.15. Where $THRSH$ is the threshold which differentiates traversable from untraversable obstacles, and it is used to normalize the values.

$$rough(angle)(k) = \frac{value(i)(j)}{THRSH}\left[1 - \frac{dist(value(i)(j))}{max\_distance}\right] \tag{3.15}$$

The final standard deviation value for one direction is based on all the values within that direction. Along a direction, using the array containing all the values of the comprising tiles, the standard deviation is calculated as follows in equation 3.16.

$$std(angle) = \sqrt{\frac{\sum_{k=0}^{N}\left[rough(angle)(k) - \bar{rough}(angle)\right]^2}{N}} \tag{3.16}$$

Where $std(angle)$ is the final value which will represent the roughness along one direction. The $\bar{rough}(angle)$ is the average value of all the roughness values of the tiles along that particular direction, held in the array. Each of the $rough(angle)(k)$ is calculated as in 3.15.

Naturally, the algorithm does not consider the roughness along a direction behind an untraversable obstacle if such a case occurs.

Putting together these two membership functions by using the algebraic product, gives the final obstacle membership function, which takes both of these types of obstacles into account. As it can be seen in the figure 3.10, for a sample environment situation, the untraversable obstacle and roughness membership functions are created. The roughness is principally used as the addition, to give more detail about the traversable part of the terrain. Also, the roughness between the robot and the untraversable obstacle is measured to give a notion of the terrain quality. Merging these two together, the output membership function which depicts the state of the terrain with as much detail as possible is obtained, and based on it, the control outputs can be extracted.

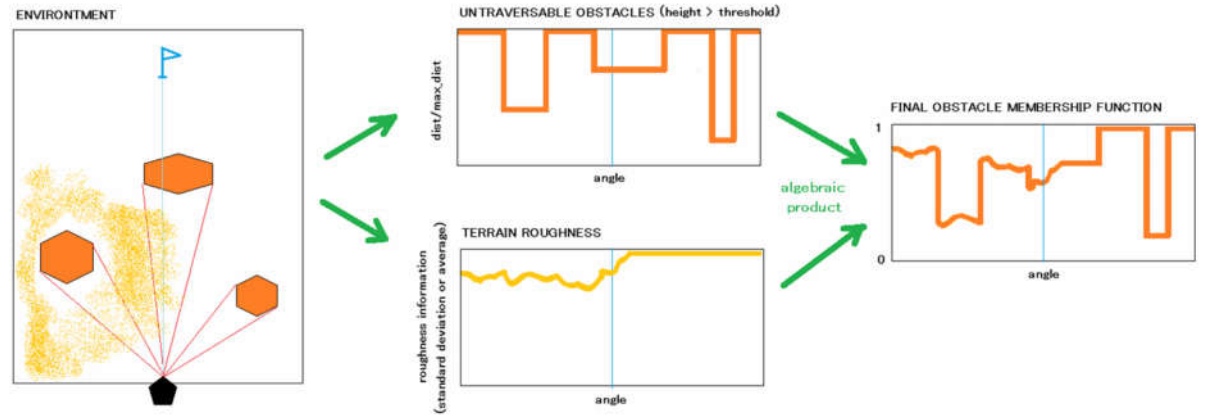


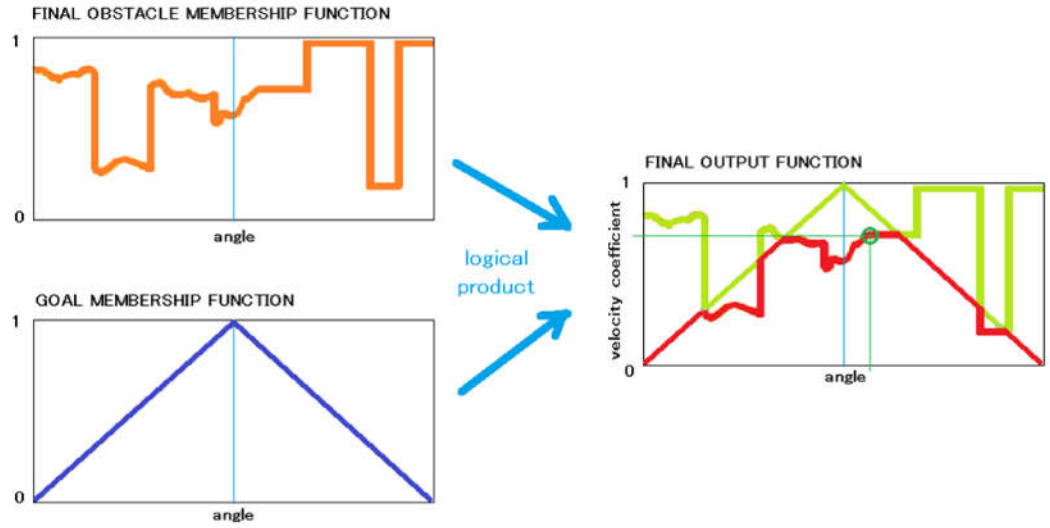
FIGURE 3.10: The process of creating the final obstacle membership function

An example of the roughness obstacle membership function is depicted in figure 3.9 (bottom).

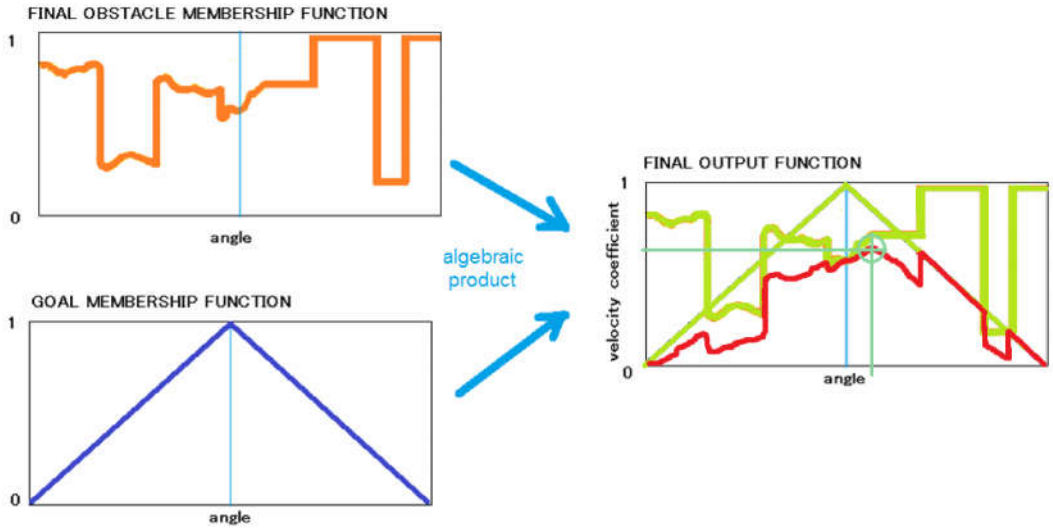
3.4.3 Defuzzification and Control Output Generation

The defuzzification process produces a quantifiable result based on the fuzzy membership functions. The defuzzification is done on the final output membership function. There are two methods to obtain the final output function, from the goal and obstacle membership functions. One is to apply a simple logical product, which returns the minimum of the two functions for each direction, as depicted in figure 3.11(a). The alternative is to apply an algebraic product of the two functions, as in figure 3.11(b). The latter one has proven to be much more effective, and meaningful, so in the end it has been implemented.

There are different ways to perform the defuzzification and extract the desired values. The outputs needed in this application are the best direction and speed coefficient that the robot should execute. The speed coefficient is a general value between 0 and 1, which can be multiplied by the maximum velocity of the rover. This means, that if there is a clear path to the goal, the robot should go there with full speed, but if it some rough ground it should go slower. The speed coefficient is taken as the maximum value of the final output membership function and this coincides with the direction containing the least problematic obstacles, or none, and which will get the robot closer to the desired goal most efficiently. The corresponding heading direction, which contains the maximum function values, is the direction in which the rover should go.



(a) using logical product



(b) using algebraic product

FIGURE 3.11: The two possible methods for obtaining the output membership function

Having extracted this information, the defuzzification procedure is finished, and the controls are forwarded to the motor controller for executing the movement.

Chapter 4

Test Bed Description and Experimental Results

4.1 Rover Description and Environment

The proposed algorithm has been tested in simulation using specifically created test maps and maps generated previously by a real system. It demonstrated good performance in global map generation and obstacle avoidance. After this validation, it has been proceeded to practical experiments with the actual rover.

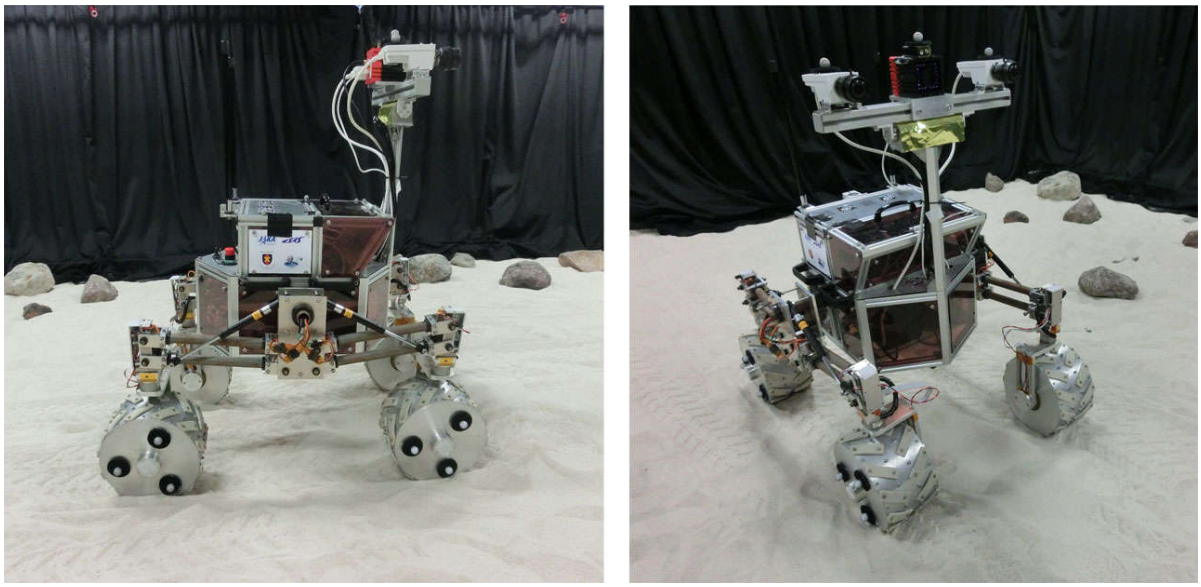


FIGURE 4.1: The "Cuatro" rover used for tests

The rover used for testing is the "Cuatro" (Figure 4.1). It was developed by professor Genya Ishigami of ISAS [35]. Currently it is used to test the navigation algorithms at the JAXA's Institute for Space and Aeronautical Science (ISAS). It was conceived based on the study conducted by the MELOS group [10]. Some of the specifications of the explorer are that it weighs approximately 35kg, it has a 4 wheel drive and a 4 wheel steering system. It possesses a rocker suspension mechanism with independent shock absorbers on each wheel. To acquire 3D images of it surroundings, it uses a stereo camera pair and a laser

range finder (LRF). An alternative to this system, a "Swiss Ranger SR4000" TOF camera has been implemented. The tests have been held at ISAS' rover laboratory (Figure 4.2).

FIGURE 4.2: The ISAS' rover laboratory sand box

There is a large sand box with rocks and black curtains around it, whose purpose is to mimic the Martian terrain conditions. The test site is equipped with a "OptiTrack V100R2" motion capture system (Figure 4.3), which can track the absolute position of the rover, which can be compared with the odometry information to analyse the wheel slippage and other deviances.

FIGURE 4.3: The ISAS' rover laboratory motion capture system

4.2 Implementation

Due to the time limitations of the research stay and the ISAS laboratory access, there have been only two test sessions. The tests have been organized as the proof of concept, meaning, the fuzzy controller and the mapping program were not directly implemented on the rover. To test the concept, it was sufficient to obtain the position and orientation data together with the local area 3D map. This data was adequate to generate the control outputs.

The connection to the rover is made via WiFi network. There is a special router, to which the user connects its computer, an through it, command are sent and data acquired

for analysis. The layout of the rover devices and the communication system is given in figure 4.4. The code that was running on the rover was on a different computer, so in

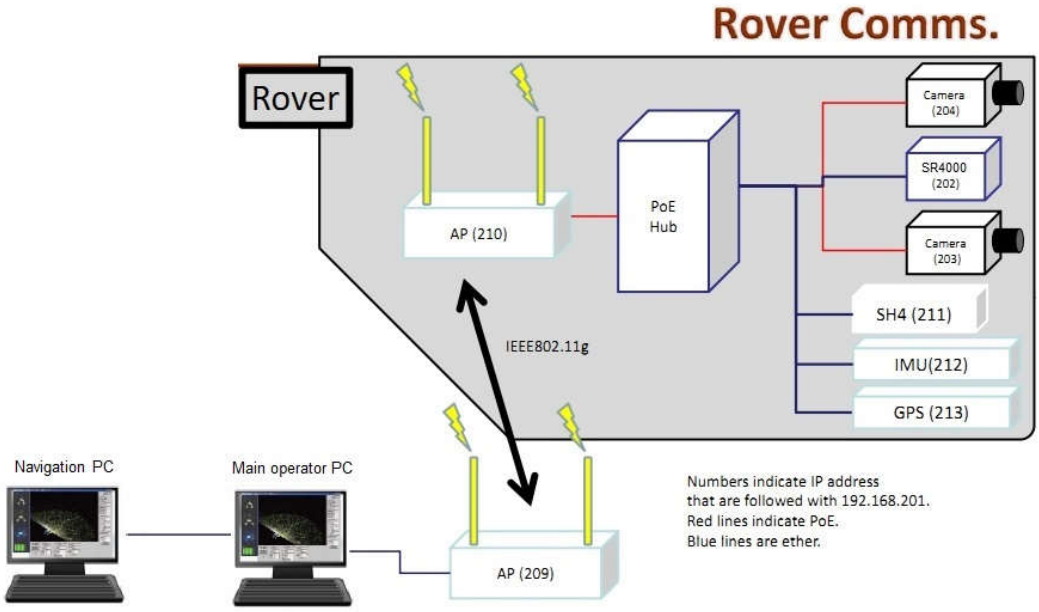


FIGURE 4.4: The "Cuatro" rover's system

order to save time and facilitate the testing process, the data was communicated from the "main" computer running the rover software, to the computer running the navigation and mapping algorithms. Therefore, the data exchange was done via FTP server or just using the flash memory. Even though it was a slower process it was sufficient to successfully test the algorithms. Similarly, in the other direction, the command outputs have been communicated to the "main" computer. First, the appropriate commands for turning in place are executed, to set the desired heading direction, followed by the required displacement forward. The data obtained from the main operator PC, is the 3D point cloud. It is a set of x,y,z coordinates for all the points that the TOF camera captures. The main software, automatically performs the Roll, Pitch, Yaw and camera Pitch transformations, so just the final transformed image is used.

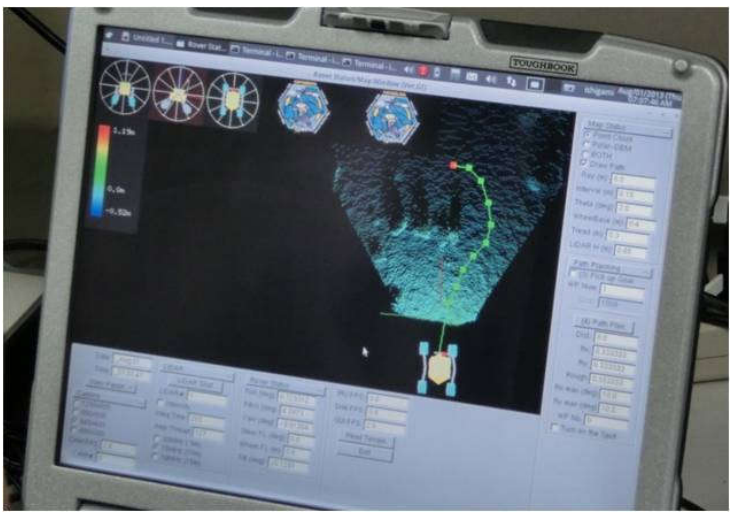
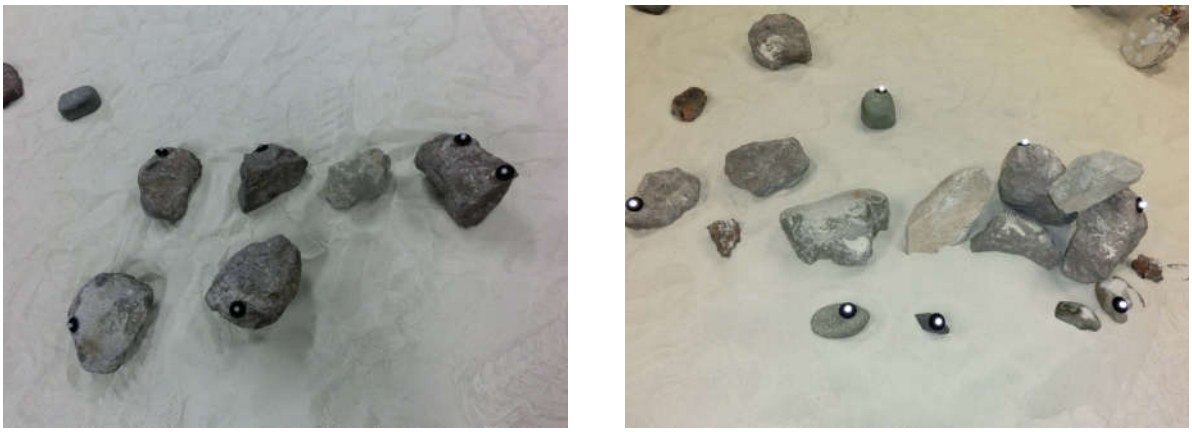


FIGURE 4.5: The generated global path with sub-goals using Dijkstra's algorithm

The software of the rover uses a post-processing technique to obtain a rough estimate of the global path to the goal, and generate the set of sub-goals, which the fuzzy controller takes (Figure 4.5). To accomplish this, the Dijkstra's algorithm was used. The positions of the sub-goals are directly forwarded to the fuzzy controller to estimate the heading direction.

The outputs that are generated are the desired heading direction, which is given relative to the current rover's heading and the desired displacement in that direction.

To further analyse the performance of the algorithm, the motion capture system was used to obtain the position of the obstacles and display them (figure 4.6). The motion capture system was also used to observe the rovers' actual displacements, so that the commands are not given simply relying on the odometry, because the positioning error would be too large and the mapping process would be inaccurate.



(a) Test 1 obstacle setup

(b) Test 2 obstacle setup

FIGURE 4.6: Optical markers on the untraversable obstacles

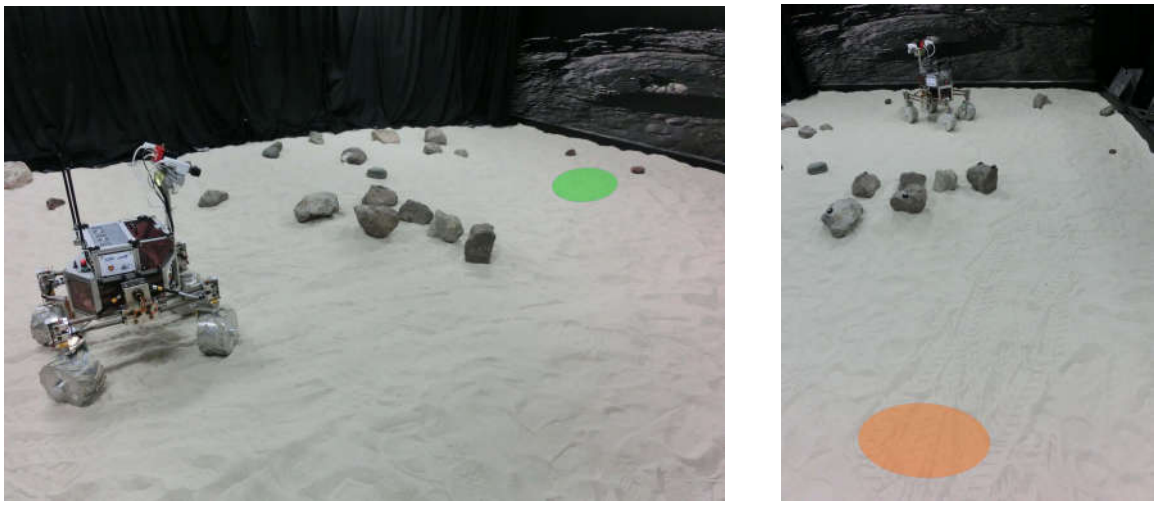
The local 3D maps, which are fed to the system from the camera at each iteration, are made with the rover's TOF camera tilted to an angle of approximately 50° downwards, with respect to the horizontal plane going through the camera center, which represents 0° the angle. This produces small and precise local maps of the immediate surrounding terrain, which are useful for path correction and accurate mapping. Conversely, the map used by the Dijkstra algorithm to extract the sub-goals, uses a significantly smaller tilt angle of about 20°, and provides a much larger terrain view, so it includes the global goal's location. This sub-goal generation, as stated before is performed only once, for one global goal.

4.3 Testing and Results

The robot has been presented with two separate test cases. The main objective was to test its capability to perform real-time mapping and path re-planning based on new information obtained from newly explored areas. In this section, the experiments are presented by first explaining the terrain set-up created to test a specific functionality, with displaying the robot's start and the goal position. After that, the actual performance is presented and analysed, and the proposed systems' benefits are emphasized.

Test 1

Firstly, a simple case is given, with an untraversable obstacle in front of the rover and a couple of rocks scattered to the left, with some random roughness of the terrain (Figure 4.7).



(a) starting position

(b) goal position reached

FIGURE 4.7: Test case 1 terrain set-up (goal positions showed as a green elipse and the starting position as an orange one)

The goal was set behind the untraversable obstacle and an array of sub-goals to reach there was calculated based on the long distance area scan and using the Dijkstra algorithm. This generated path is shown in figure 4.8.

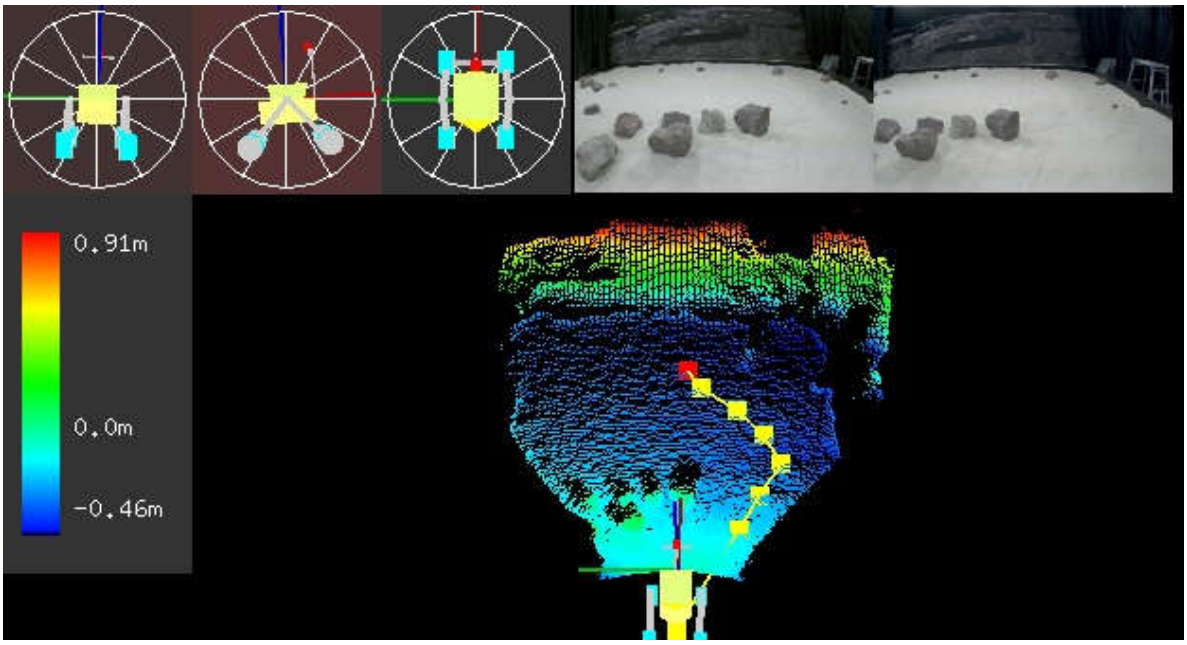


FIGURE 4.8: The print screen of the main computer while generating a global path for the first test case.

The set-up at hand was used to report the rover's capability of following the sub-goals, while taking into account the fresh terrain information obtained from the local maps. The resulting graph showing the position of the untraversable obstacle, the generated sub-goals and the rover's actual path is shown in figure 4.9.

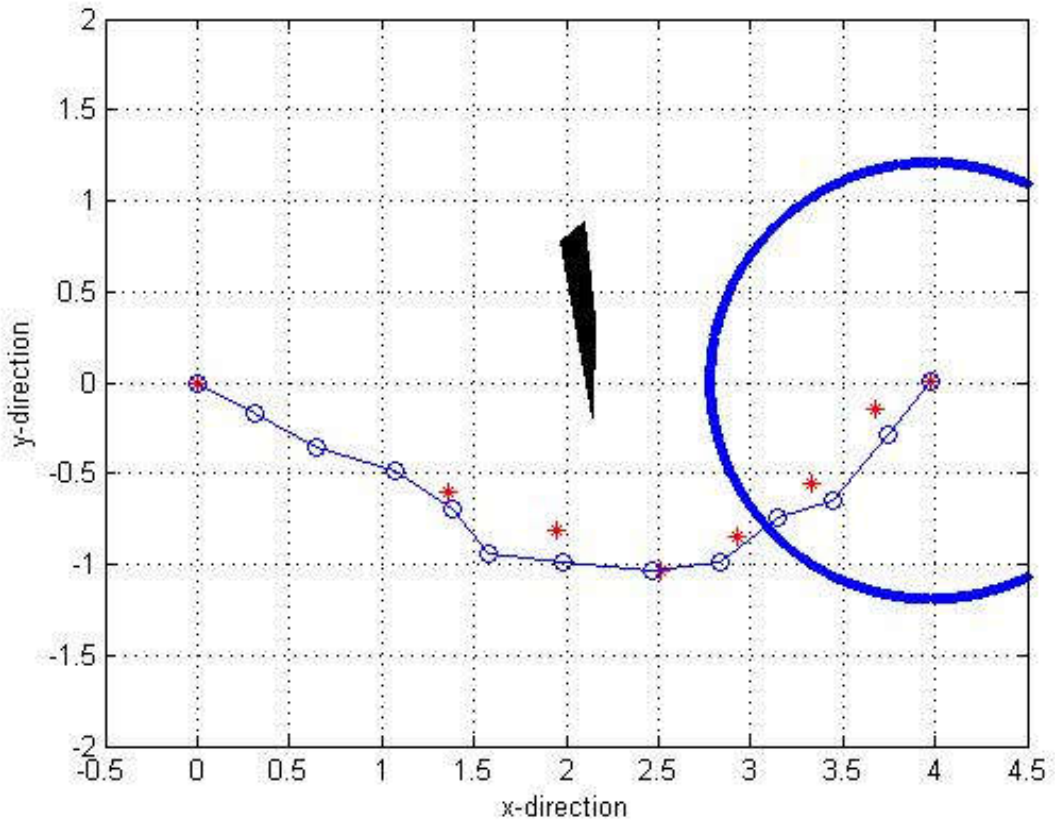


FIGURE 4.9: Test case 1 graph. The obstacle is shown in black, subgoals in red and the actual path in blue

The large blue circle is used as the sub-goal scanner, meaning, it allows the rover to skip to the next sub-goal, assigning it a higher priority, if it is close enough, so it does not have to reach all the sub-goals if not necessary. This process is performed at each step and combined with the fuzzy controller, it allowed the rover to skip some of the sub-goals as shown. The the first goals which have been skipped in figure 4.9 were avoided because the alternate path was safer from the collision standpoint. The last sub-goal were skipped in order to reach the final goal faster since there were no obstacles on the way towards it.

The final global map obtained is depicted in figure 4.10. In appendix D, the generation of the global map can be followed through the shown steps. Of course, there were some inaccuracies due to the nature of this discreet mapping process. For example, the red segment in the center of the image, which is not completely intuitive, is due to the slight change in the level of the field. However, as the rover approaches this area, it finds out that there are no obstacles here and overwrites the map data. In this case, the rover traversed more than it should have, so it "missed" to record and patch this information, so this false obstacle remained. This situation could have been avoided if a localization system more accurate than odometry was used, such as visual odometry for example,

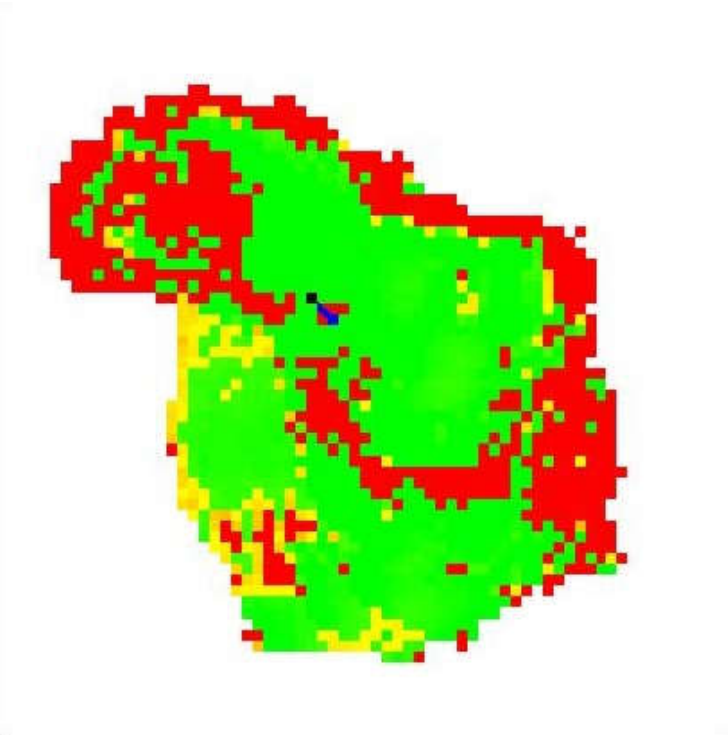


FIGURE 4.10: The Global Map of the first test case.

which is not affected by wheel slippage. The final global map resembles the actual map, though not perfectly. It can be considered as depicting the explored area and could be reused if necessary to navigate through it.

The fuzzy membership function graph shows the final output, the goal membership function and also the two constituent membership functions of the obstacle membership - the roughness and the untraversable membership function. The output commands given in the console are in accordance to the fuzzy membership functions.

The algorithm performed very good, and the difference between the given path and the effectuated one is obvious and leads the rover to a safer distance from the obstacle, and directly to the final goal when possible.

Test 2

The second test was conceived to prove the rovers ability to re-plan the path as it is being executed, which leads to a more efficient final path. This particular case deals with the occlusion of the terrain data caused by a high obstacle, which does not allow the initial global path-planner to generate the optimal path. The generated path considers the occlusion as an obstacle (Figure 4.11), so the sub-goals generated take the rover around it, with a longer trail. In this case, the local map is smaller than in the first one, because the camera is tilted more. This provides a more accurate mapping, but the path selection is based on a smaller view and could thus be less reliable.

From the starting position, a wide-area scan has been performed, based on which the Dijkstra algorithm is executed to extract the preliminary global path. The generated path of with sub-goals is shown in figure 4.12

(a) starting position

(b) goal position reached

FIGURE 4.11: Test case 2 terrain set-up (goal positions showed as a green elipse and the starting position as an orange one)

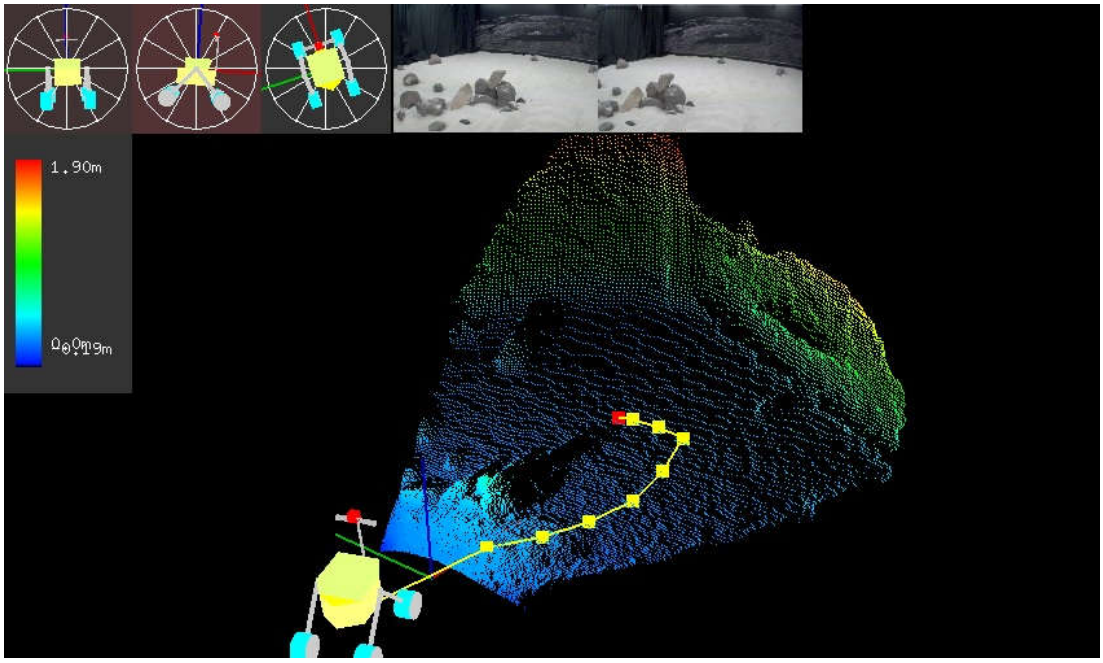


FIGURE 4.12: The print screen of the main computer while generating a global path for the second test case. The bar on the left indicates the terrain elevation.

The graph showing this situation, with the position of the obstacle, generated sub-goals and the actual path executed, is shown in figure 4.13. The final global map obtained is depicted in the figure 4.14. The steps in obtaining this map and the local patches and fuzzy membership functions for each steps are presented in appendix D. In this second case, the rover managed to "cut through" the occluded area. When it received up-to-date information about the previously occluded terrain and realized that there are no obstacles and the path was clear, it assigned a direct heading to the following goal.

The difference between the paths is not very large, but it is significant. In the end, the final step length can be set to depend on the Euclidean distance to the global goal. If the output move command is greater than the distance to the goal, then the command is changed to be this exact distance and the final goal will not be overshot. The conclusion of the practical tests would be that the algorithm has proven to be competent in the

presented situations and thus apt to improve the initial path and transform it into a more efficient one while concurrently mapping the explored area.

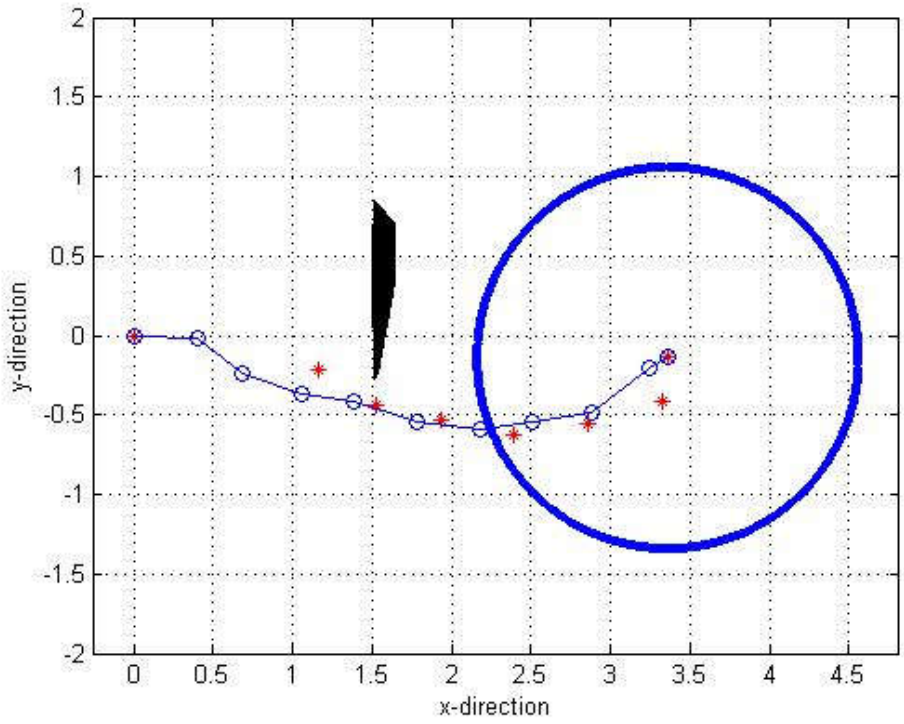


FIGURE 4.13: Test case 2 graph. The obstacle is shown in black, subgoals in red and the actual path in blue

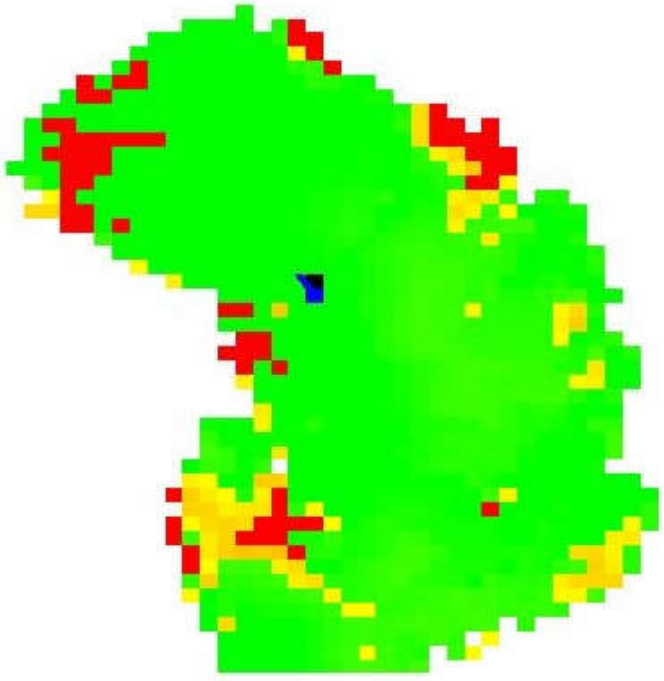


FIGURE 4.14: The Global Map of the second test case.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this work, a novel approach has been introduced, which implements a path selection approach using the terrain roughness information and a fuzzy controller enabling the rover to run in real-time. The fuzzy controller consists of two main membership functions, one that represents the goal direction function and the other that depicts the terrain properties. The goal membership function is formed based on the relative positions of the rover and the sub-goal. The obstacle membership function, which is obtained from the untraversable obstacle and roughness membership functions, is formed by scanning the rovers surrounding area and processing its height information. Merging the goal and obstacle membership functions, a final output function is obtained from which the control variables such as the heading direction and the desired speed can be extracted.

A second important task this work deals with, is concurrently building a global map of the explored area, using the local terrain information the rover gathers at specific locations and with a specific attitude. The way in which the map is stored has been conceived in order for the algorithm to be reliable and memory efficient at the same time. The first and the second part are mutually dependent, so that the fuzzy controller requires the global map information to have the absolute knowledge of the surrounding area and produce the desired output controls.

Putting this two parts together, a complete closed loop system is generated. It uses basic inputs, and employing simple path selection and mapping methods, drives the rover to the goal, while increasing the knowledge of the surroundings as it approaches them. One of the advantages of this system is that in this way, having been given an approximate global path, it can locally re-plan it in real-time and thus producing an improved path.

The system has been tested both in simulation and on a real rover test bed. The results have been presented for different realistic terrain set-ups, and the rover's performance analysed. This kind of system can be executed in real-time and produce meaningful and useful data which a planetary exploration vehicle needs.

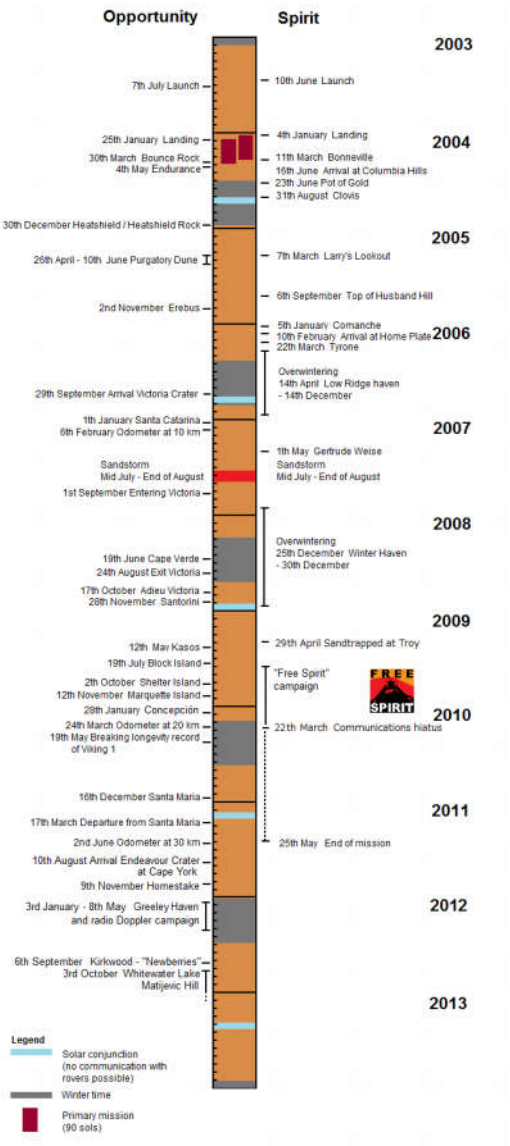
5.2 Future Improvements

Regarding the explored terrain mapping there are some practical problems which should be addressed. Naturally, as more area is being revealed, more data is being stored until it reaches some physical storage limits. This problem is not covered in this work, but there are some possibilities of dynamically changing the tile size, or making sub-maps of different tile sizes depending on the distance from the current exploration location.

Moreover, a general trend in navigation is to, by all means, try to avoid the obstacles, and drive through the flattest surface possible. Thus, most of the algorithms give more weight to the obstacle evasion, which is sensible because the time on a planet is not limited. This is done even though most of the rover structures are made with good suspension systems so if it were necessary, they could go over even tough terrain and some obstacles. The problem which then arises is to make a suitable AI and control system, which would take into account the terrain properties as to define the loose rocks and dangerous areas where a lot of slippage might occur, but also to properly accommodate the rover configuration to the terrain. Some of the rovers might possess articulated suspension systems which would allow them to traverse even very steep terrain without falling over.

Appendix A

Mars Exploration Rover - Mission Overview



Appendix B

Algorithm Pseudocodes

```
1  Create a node containing the goal state node_goal
2  Create a node containing the start state node_start
3  Put node_start on the open list
4  while the OPEN list is not empty
5  {
6  Get the node off the open list with the lowest f and call it node_current
7  if node_current is the same state as node_goal we have found the solution;
  break from the while loop
8      Generate each state node_successor that can come after node_current
9      for each node_successor of node_current
10      {
11          Set the cost of node_successor to be the cost of node_current plus
the cost to get to node_successor from node_current
12          find node_successor on the OPEN list
13          if node_successor is on the OPEN list but the existing one is as good or
better then discard this successor and continue
14          if node_successor is on the CLOSED list but the existing one is as good
or better then discard this successor and continue
15          Remove occurences of node_successor from OPEN and CLOSED
16          Set the parent of node_successor to node_current
17          Set h to be the estimated distance to node_goal
(Using the heuristic function)
18           Add node_successor to the OPEN list
19      }
20      Add node_current to the CLOSED list
21  }
```

FIGURE B.1: A* Algorithm Pseudocode

procedure CalculateKey(s)
{01'} return $[\min(g(s), rhs(s)) + h(s_{start}, s) + k_m; \min(g(s), rhs(s))]$;
procedure Initialize()
{02'} $U = \emptyset$;
{03'} $k_m = 0$;
{04'} for all $s \in S$ $rhs(s) = g(s) = \infty$;
{05'} $rhs(s_{goal}) = 0$;
{06'} U.Insert(s_{goal}, CalculateKey(s_{goal}));
procedure UpdateVertex(u)
{07'} if $(u \neq s_{goal})$ $rhs(u) = \min_{s' \in Succ(u)}(c(u, s') + g(s'))$;
{08'} if $(u \in U)$ U.Remove(u);
{09'} if $(g(u) \neq rhs(u))$ U.Insert$(u$, CalculateKey$(u))$;
procedure ComputeShortestPath()
{10'} while (U.TopKey() $\dot{<}$ CalculateKey(s_{start}) OR $rhs(s_{start}) \neq g(s_{start}))$
{11'} $k_{old} =$ U.TopKey();
{12'} $u =$ U.Pop();
{13'} if $(k_{old} \dot{<}$ CalculateKey$(u))$
{14'} U.Insert$(u$, CalculateKey$(u))$;
{15'} else if $(g(u) > rhs(u))$
{16'} $g(u) = rhs(u)$;
{17'} for all $s \in Pred(u)$ UpdateVertex(s);
{18'} else
{19'} $g(u) = \infty$;
{20'} for all $s \in Pred(u) \cup \{u\}$ UpdateVertex(s);
procedure Main()
{21'} $s_{last} = s_{start}$;
{22'} Initialize();
{23'} ComputeShortestPath();
{24'} while $(s_{start} \neq s_{goal})$
{25'} /* if $(g(s_{start}) = \infty)$ then there is no known path */
{26'} $s_{start} = \arg\min_{s' \in Succ(s_{start})}(c(s_{start}, s') + g(s'))$;
{27'} Move to s_{start};
{28'} Scan graph for changed edge costs;
{29'} if any edge costs changed
{30'} $k_m = k_m + h(s_{last}, s_{start})$;
{31'} $s_{last} = s_{start}$;
{32'} for all directed edges (u, v) with changed edge costs
{33'} Update the edge cost $c(u, v)$;
{34'} UpdateVertex(u);
{35'} ComputeShortestPath();

FIGURE B.2: D* Lite Algorithm Pseudocode

ComputeShortestPath()
04. while ($\min_{s \in OPEN}$(key(s)) $<$ key(s_{start}) OR $rhs(s_{start}) \neq g(s_{start})$)
05. peek at state s with the minimum key on *OPEN*;
06. if ($g(s) \geq rhs(s)$)
07. $g(s) = rhs(s)$;
08. remove s from *OPEN*;
09. for all $s' \in nbrs(s)$
10. if s' was not visited before
11. $g(s') = rhs(s') = \infty$;
12. $rhs_{old} = rhs(s')$;
13. if ($rhs(s') >$ ComputeCost($s', s, ccknbr(s', s)$))
14. $rhs(s')$ = ComputeCost($s', s, ccknbr(s', s)$);
15. $bptr(s') = s$;
16. if ($rhs(s') >$ ComputeCost($s', s, cknbr(s', s)$))
17. $rhs(s')$ = ComputeCost($s', cknbr(s', s), s$);
18. $bptr(s') = cknbr(s', s)$;
19. if ($rhs(s) \neq rhs_{old}$)
20. UpdateState(s');
21. else
22. $rhs(s) = \min_{s' \in nbrs(s)}$ ComputeCost($s, s', ccknbr(s, s')$);
23. $bptr(s) = \text{argmin}_{s' \in nbrs(s)}$ComputeCost($s, s', ccknbr(s, s')$);
24. if ($g(s) < rhs(s)$)
25. $g(s) = \infty$;
26. for all $s' \in nbrs(s)$
27. if ($bptr(s') = s$ OR $bptr(s') = cknbr(s', s)$)
28. if ($rhs(s') \neq$ ComputeCost($s', bptr(s'), ccknbr(s', bptr(s'))$))
29. if ($g(s') < rhs(s')$ OR $s' \notin OPEN$)
30. $rhs(s') = \infty$;
31. UpdateState(s');
32. else
33. $rhs(s') = \min_{s'' \in nbrs(s')}$ ComputeCost($s', s'', ccknbr(s', s'')$);
34. $bptr(s') = \text{argmin}_{s'' \in nbrs(s')}$ComputeCost($s', s'', ccknbr(s', s'')$);
35. UpdateState(s');
36. UpdateState(s);

UpdateCellCost(x, c)
37. if (c is greater than current traversal cost of x)
38. for each state s on a corner of x
39. if either $bptr(s)$ or $ccknbr(s, bptr(s))$ is a corner of x
40. if ($rhs(s) \neq$ ComputeCost($s, bptr(s), ccknbr(s, bptr(s))$))
41. if ($g(s) < rhs(s)$ OR $s \notin OPEN$)
42. $rhs(s) = \infty$;
43. UpdateState(s);
44. else
45. $rhs(s) = \min_{s' \in nbrs(s)}$ ComputeCost($s, s', ccknbr(s, s')$);
46. $bptr(s) = \text{argmin}_{s' \in nbrs(s)}$ComputeCost($s, s', ccknbr(s, s')$);
47. UpdateState(s);
48. else
49. $rhs_{min} = \infty$;
50. for each state s on a corner of x
51. if s was not visited before, $g(s) = rhs(s) = \infty$;
52. else if ($rhs(s) < rhs_{min}$)
53. $rhs_{min} = rhs(s)$; $s^\star = s$;
54. if ($rhs_{min} \neq \infty$)
55. insert $s^\star$ into *OPEN* with key($s^\star$);

Main()
56. $g(s_{start}) = rhs(s_{start}) = \infty$; $g(s_{goal}) = \infty$;
57. $rhs(s_{goal}) = 0$; $OPEN = \emptyset$;
58. insert s_{goal} into *OPEN* with key(s_{goal});
59. forever
60. ComputeShortestPath();
61. Wait for changes in cell traversal costs;
62. for all cells x with new traversal costs c
63. UpdateCellCost(x, c);

FIGURE B.3: An Optimized Version of the Field D* Algorithm

```
procedure Requeue(c)
01  if v(c) = rhs(c)
02    if c ∈ W then remove c from W
03  else
04    if c ∉ W
05      insert c with key = min(v(c), rhs(c)) into W
06    else if key(c) ≠ min(v(c), rhs(c))
07      remove c from W
08      insert c with key = min(v(c), rhs(c)) into W
procedure UpdateVertex(c)
09  if c ∉ {g_i}
10    Q ← ComputePropagator(c)
11    (rhs(c), B) ← k(c, Q)
12    for all u ∈ U(c) remove (u, c) from U
13    for all b ∈ B
14      if (c, b) ∈ U then remove (c, b) from U
15      add (b, c) to U
16    Requeue(c)
procedure Propagate()
17  c ← Pop(W)
18  if v(c) > rhs(c)
19    v(c) ← rhs(c)
20    for all n ∈ N(c) UpdateVertex(n)
21  else
22    v(c) ← ∞
23    for all d ∈ D(c) UpdateVertex(d)
24    UpdateVertex(c)
procedure main()
25  initialize rhs(c) = v(c) = ∞ ∀c ∈ G
26  initialize goal rhs(g) =true distance ∀g ∈ {g_i}
27  initialize W with {g_i}
28  while (rhs(c_robot) ≠ v(c_robot)) or TopKey(W) < v(c_robot)
29    if W = {} then the goal is unreachable
30    Propagate()
```

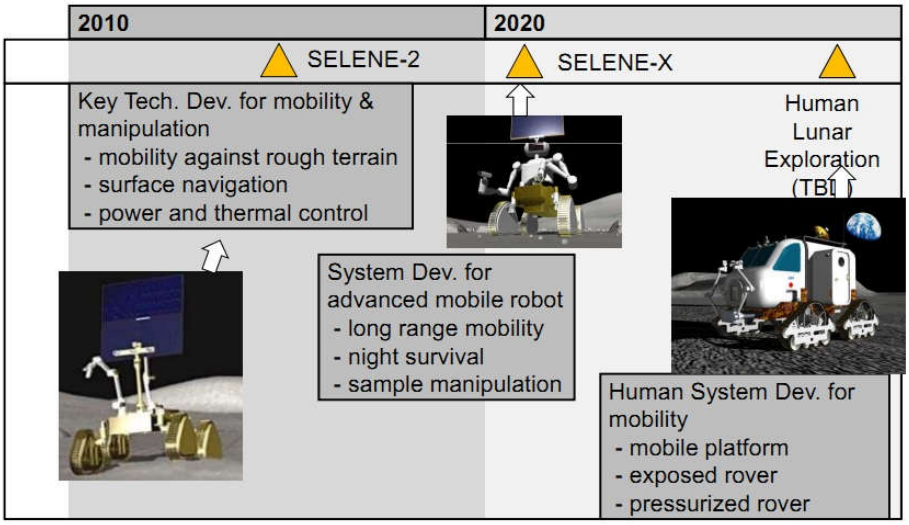
FIGURE B.4: The Pseudocode of the Core Procedures in the E* Algorithm

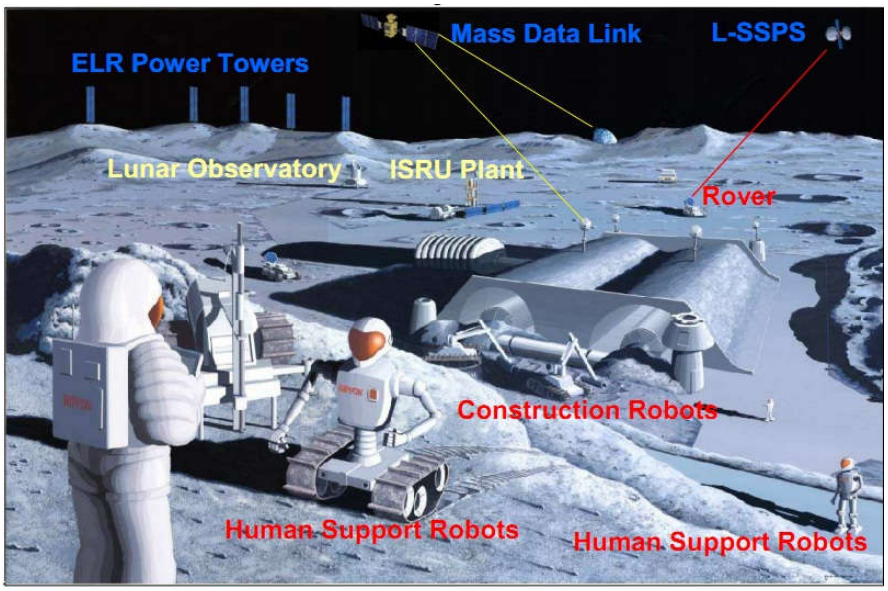
Input: $Pose_{rover}$; $Data \leftarrow$ Pose estimate and Sensors data
Output: $Ctrl_{rover} \leftarrow$ Rover commands
while *Waypoint not reached* **do**
  Update the rover's pose within G_l and G_e;
  Compute d, the distance to the last position where an image was taken;
  if *cell containing wheel pose changed from* $c_{l,i}$ *to* $c_{l,j}$ **then**
    Compute RTI, of $c_{l,i} \rightarrow F_{l,i}$;
  end
  if $d > S_{G_l}$ **then**
    Take an image;
    Extract patches from image ($n_{patches}$);
    Compute $F_{r,n}$ with $1 \leq n \leq n_{patches}$;
    Predict RTI $\rightarrow F^p_{l,n} = f(F_{r,n})$;
    Compute $r^p(c_e)$, predicted additional cost;
    Process $r^p(c_e)$, dilate and smooth;
    Send $r^p(c_e)$ to E* $\rightarrow r(c_e)^+ = r(c_e) + r^p(c_e)$;
    Update E* *trace*;
  end
  if c_{ln} *moved behind the rover* **then**
    Create s_{in} corresponding;
    Save s_{in} for learning;
  end
  Compute $Ctrl_{rover}$ based on *trace*;
  Send $Ctrl_{rover}$ to the rover control module;
end

FIGURE B.5: The Pseudocode of the RTILE

Appendix C

JAXA - Future Mission Poster





Appendix D

Test results

EXPLANATION

On the upper left is the current Global Map, on the upper right is the current local patch (after the necessary transformations, immediately before being added to the global map), on the bottom right is the console with the chosen command outputs and on the bottom left is the fuzzy membership functions' graph.

The colors of the fuzzy membership functions are:
blue - goal membership function
light blue - untraversable obstacles
black - roughness
green - final obstacle membership function
red - final output membership function

D.1 TEST 1

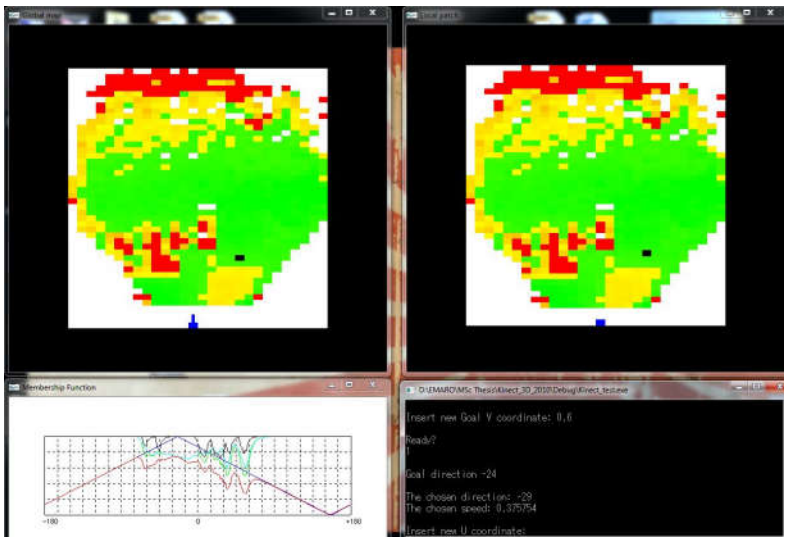


FIGURE D.1: step 0

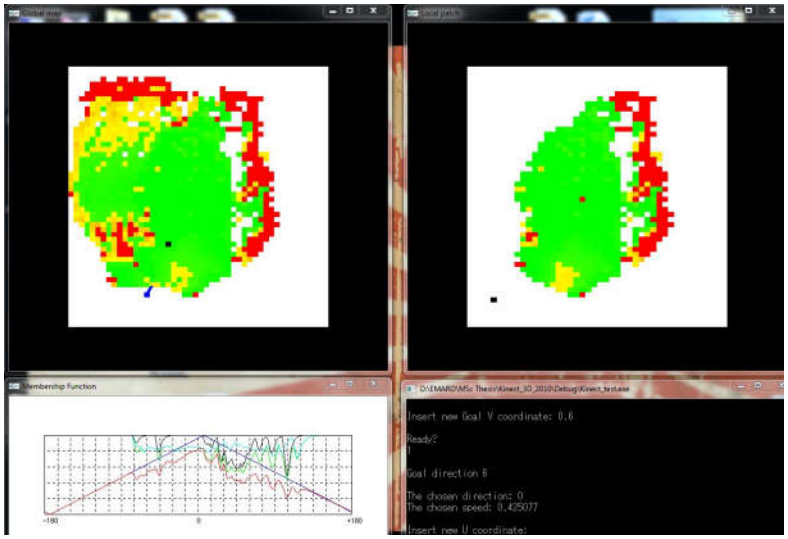


FIGURE D.2: step 1

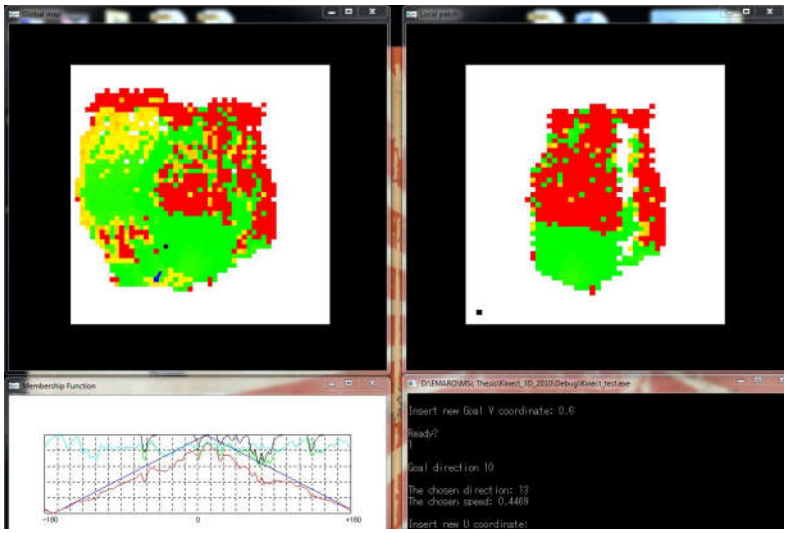


FIGURE D.3: step 2

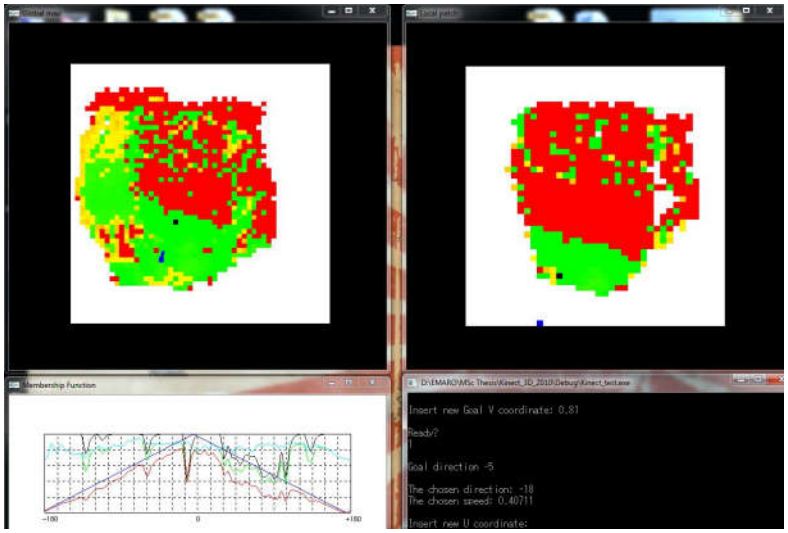


FIGURE D.4: step 3

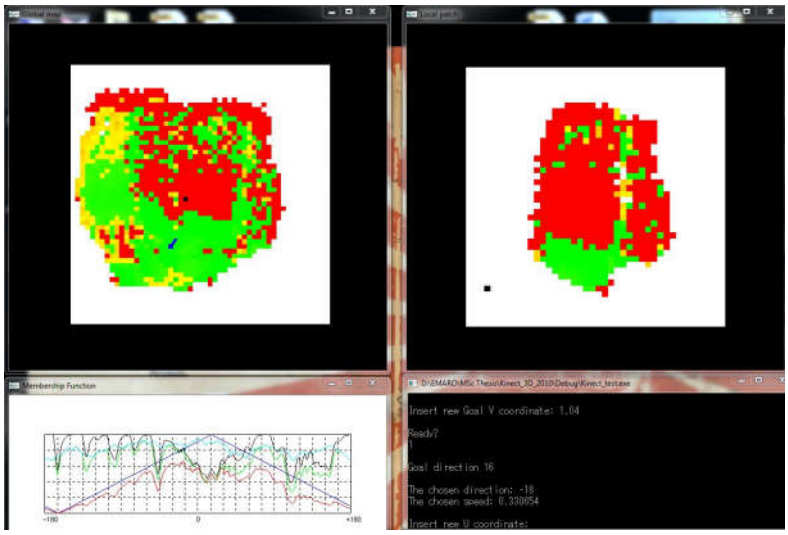


FIGURE D.5: step 4

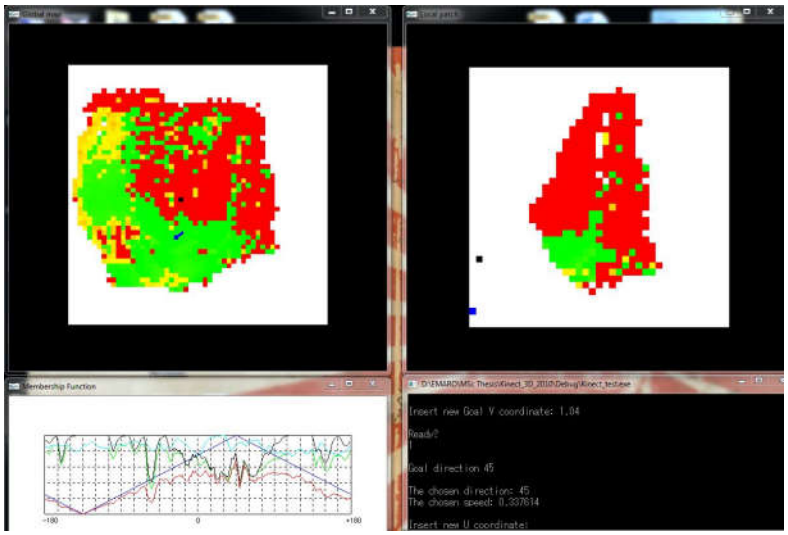


FIGURE D.6: step 5

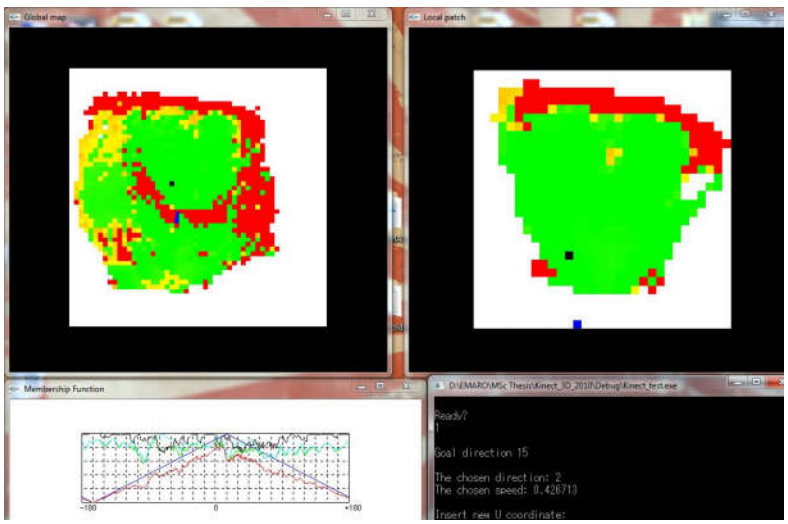


FIGURE D.7: step 6

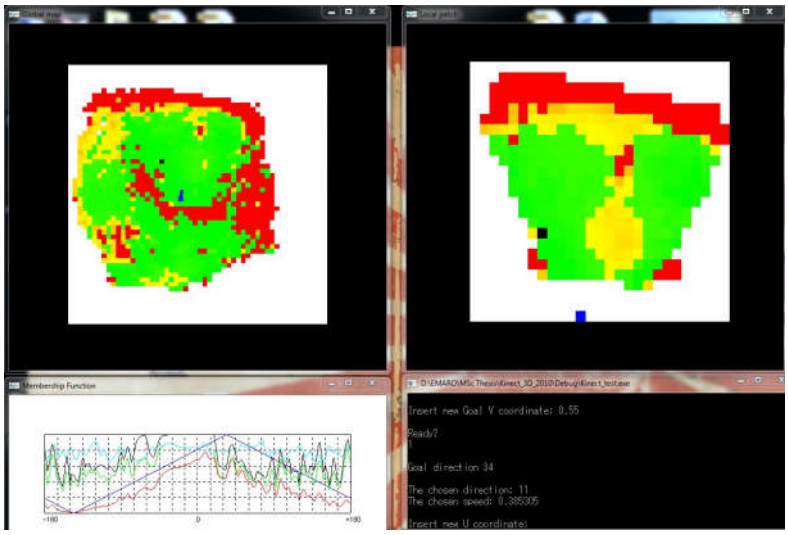


FIGURE D.8: step 7

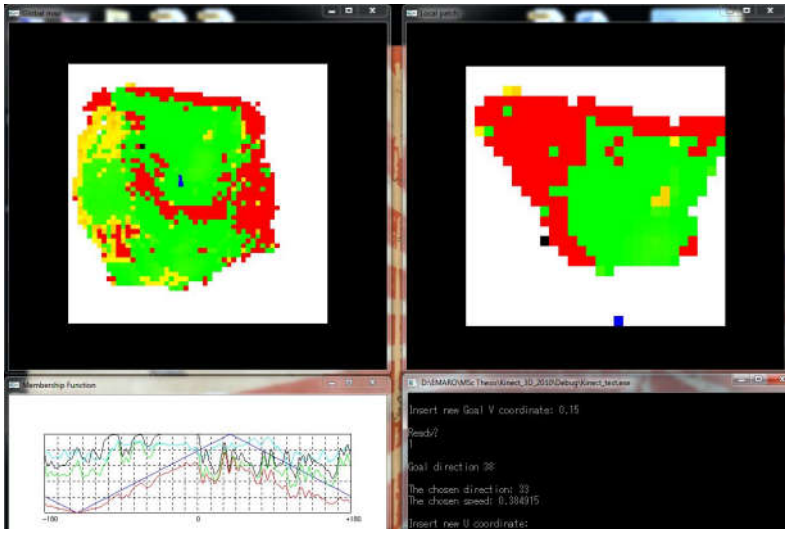


FIGURE D.9: step 8

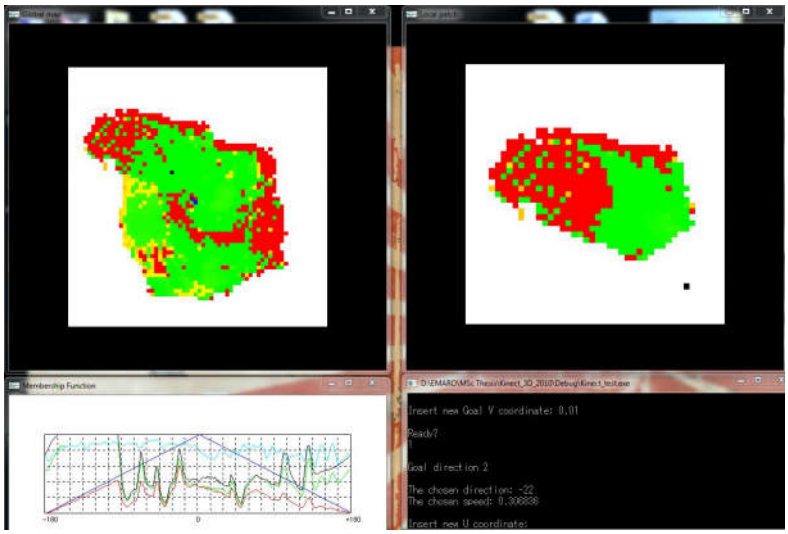


FIGURE D.10: step 9

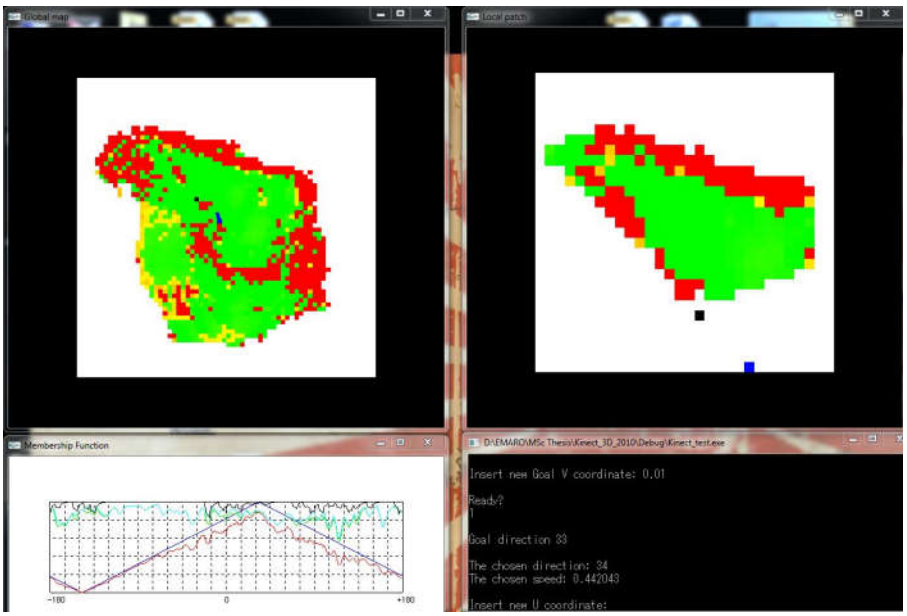


FIGURE D.11: step 10

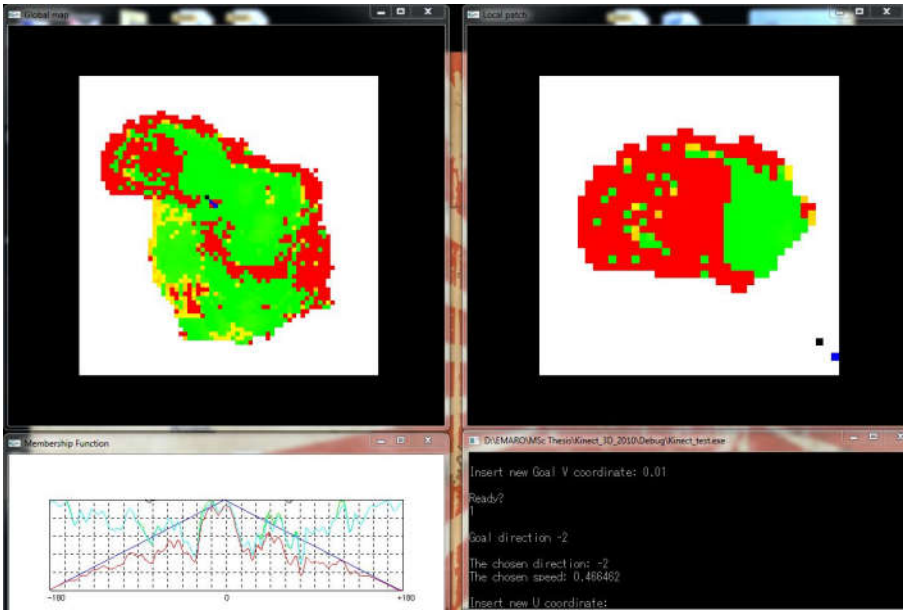


FIGURE D.12: step 11

D.2 TEST 2

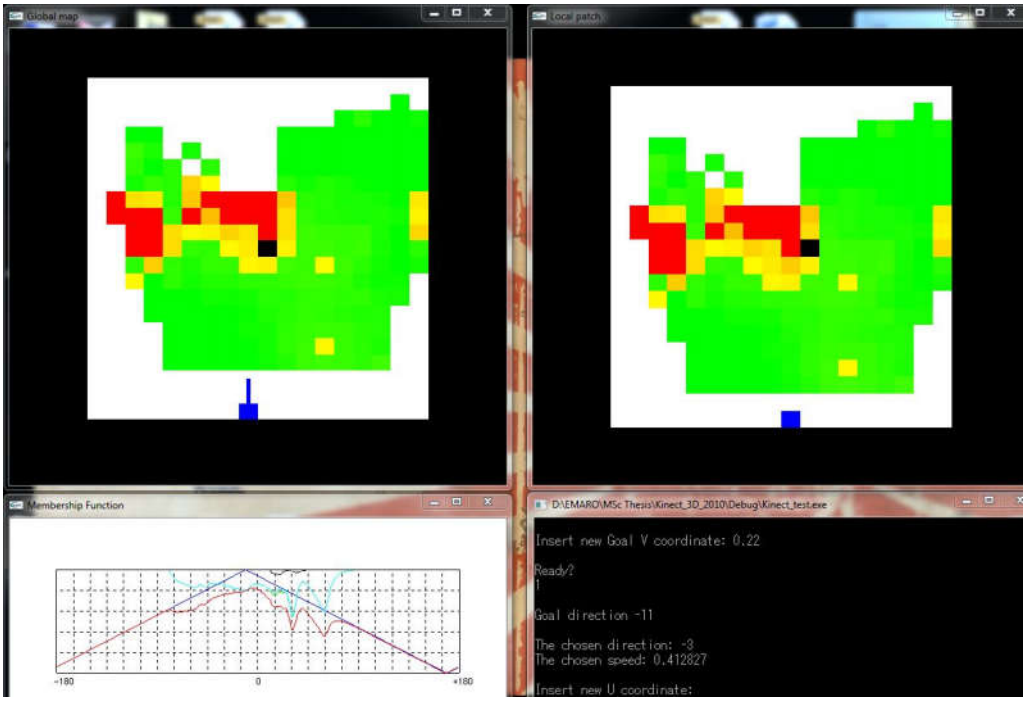


FIGURE D.13: step 0

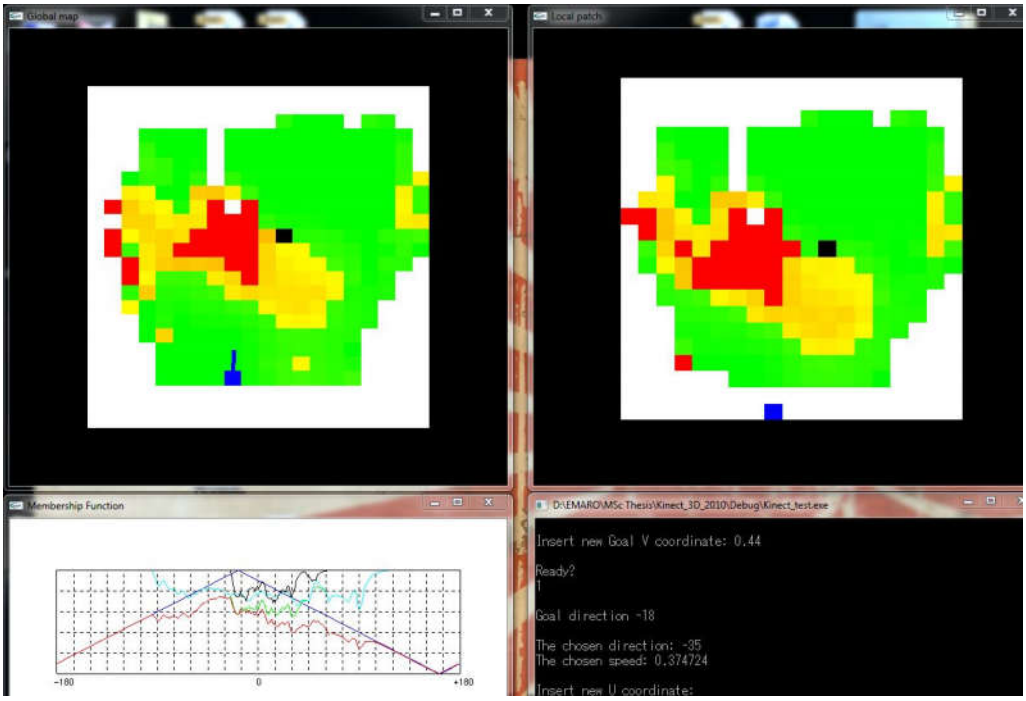


FIGURE D.14: step 1

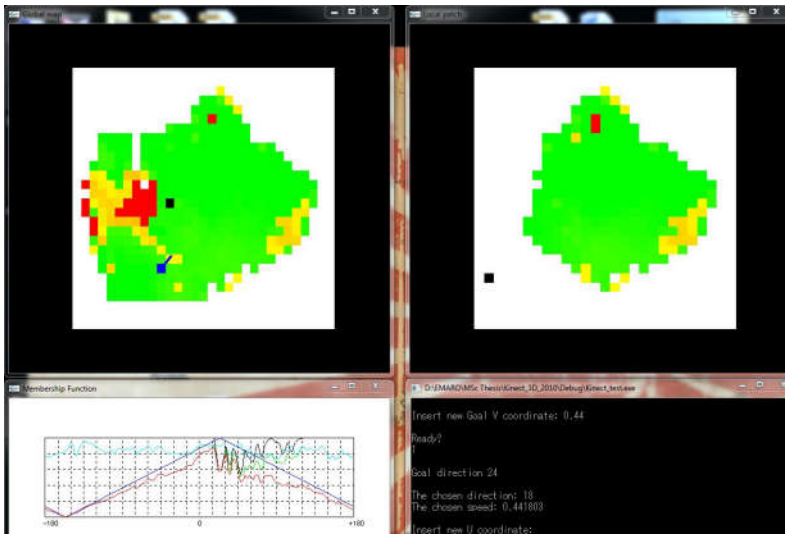


FIGURE D.15: step 2

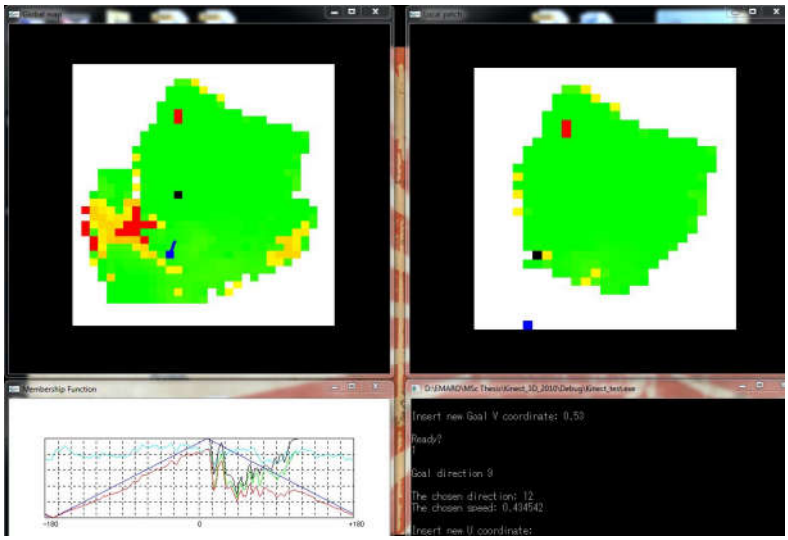


FIGURE D.16: step 3

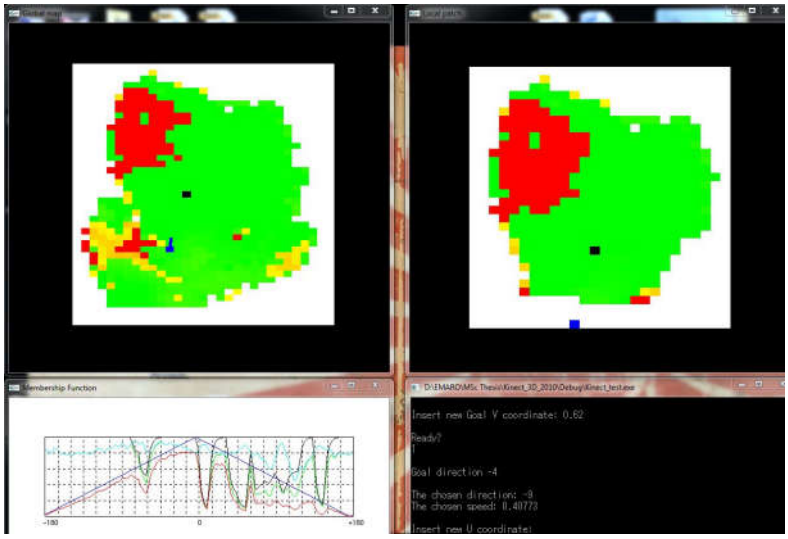


FIGURE D.17: step 4

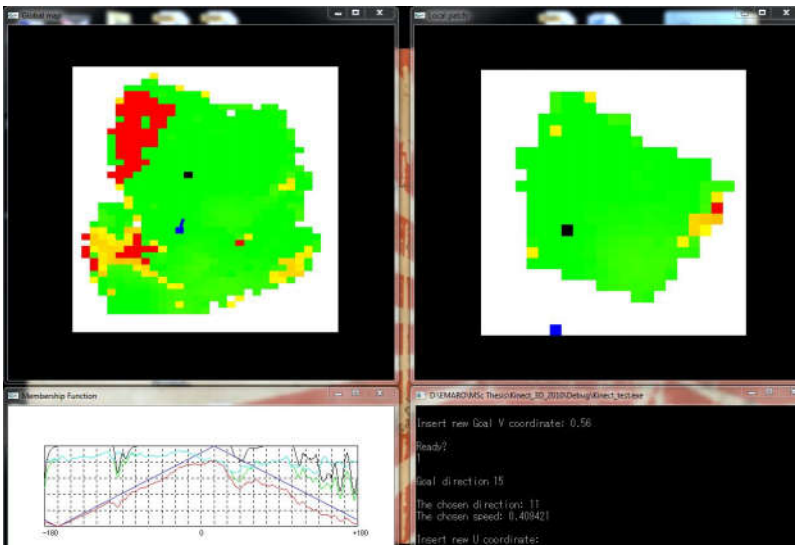


FIGURE D.18: step 5

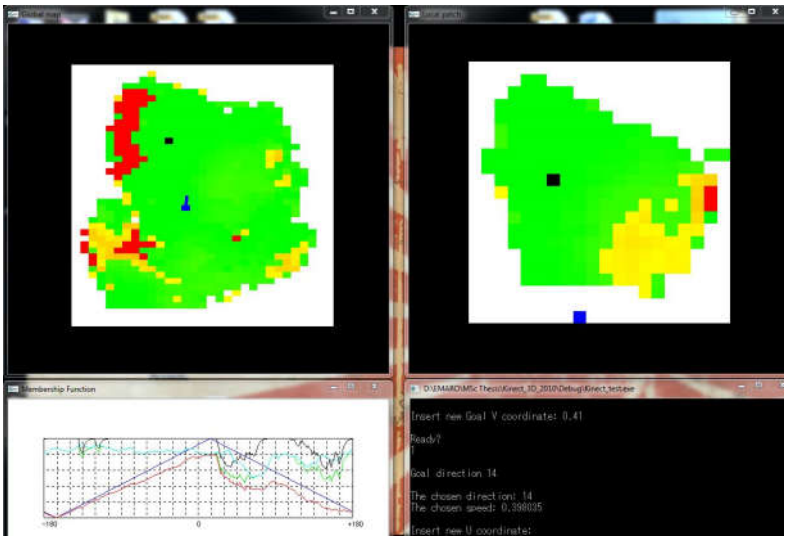


FIGURE D.19: step 6

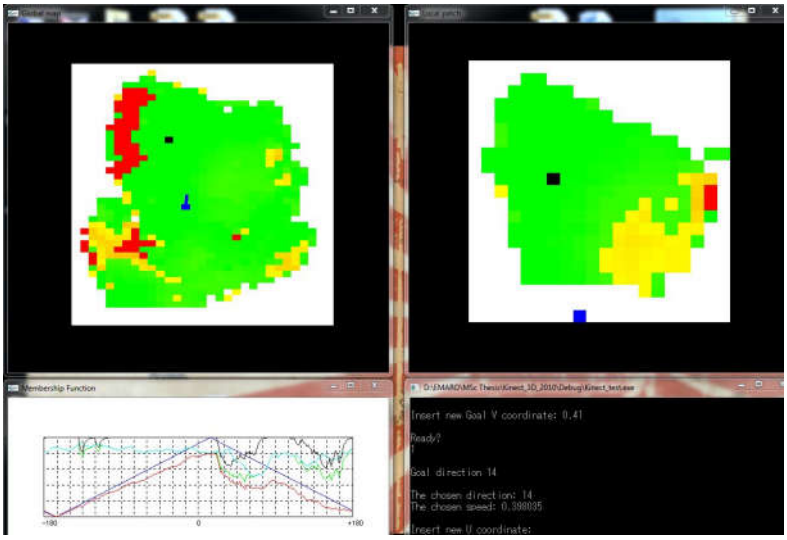


FIGURE D.20: step 7

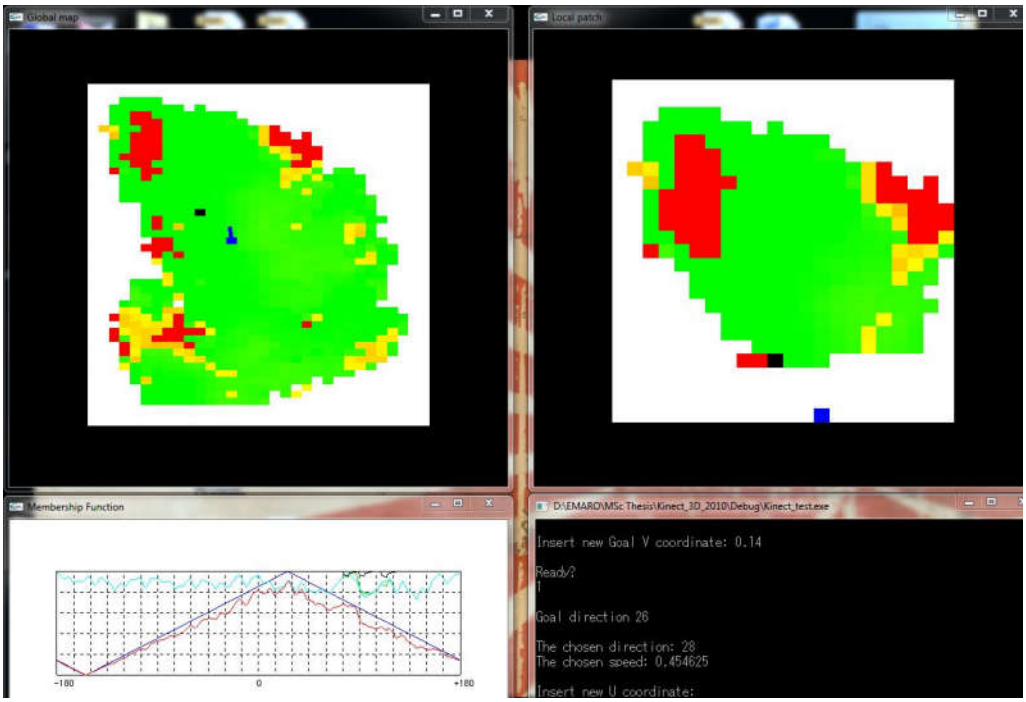


FIGURE D.21: step 8

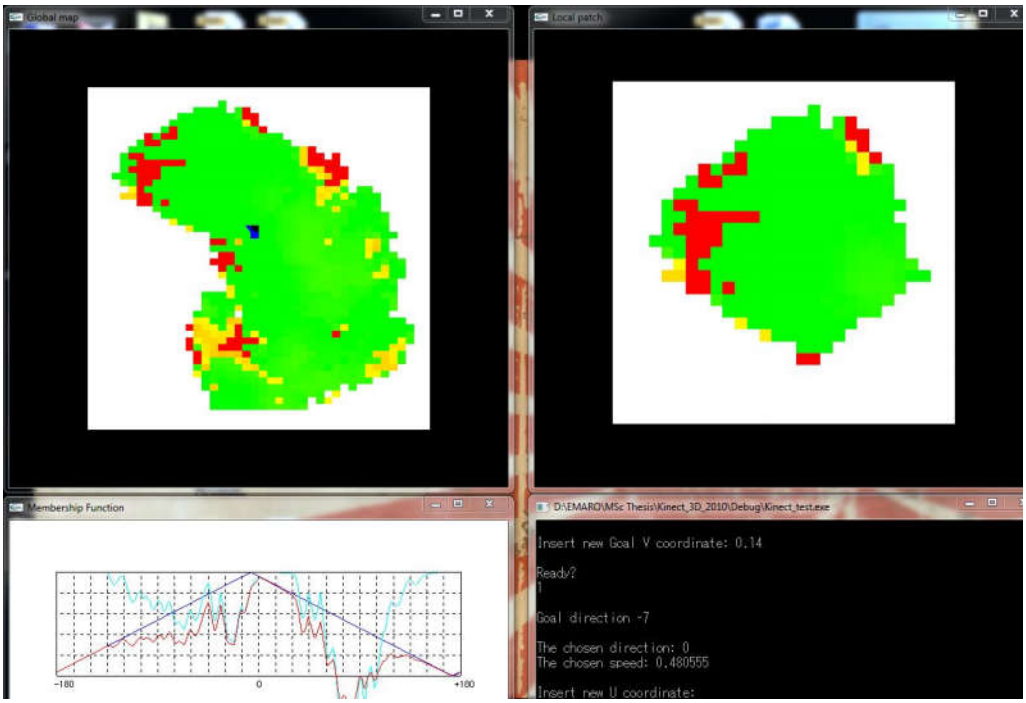


FIGURE D.22: step 9

Bibliography

[1] "Map representations," February 2013. [Online]. Available: http://theory.stanford.edu/~amitp/GameProgramming/MapRepresentations.html

[2] S. Singh, R. Simmons, T. Smith, A. Stentz, V. Verma, A. Yahja, and K. Schwehr, "Recent progress in local and global traversability for planetary rovers," in *International Conference Robotics and Automation (ICRA)*, vol. 2. IEEE, 2000, pp. 1194–1200.

[3] A. Koefoed-Hansen, "Representations for path finding in planar environments," Master Thesis, Department of Computer Science, Aarhus University, Denmark, February 2012.

[4] J. Carsten, A. Rankin, D. Ferguson, and A. Stentz, "Global path planning on board the mars exploration rovers," in *Aerospace Conference*. IEEE, 2007, pp. 1–11.

[5] D. Ferguson and A. T. Stentz, "The field d* algorithm for improved path planning and replanning in uniform and non-uniform cost environments," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-19, June 2005.

[6] R. Philippsen, "A light formulation of the e* interpolated path replanner," Ecole Polytechnique Federale de Lausanne (EPFL), Tech. Rep., 2006.

[7] J. J. Biesiadecki and M. W. Maimone, "The mars exploration rover surface mobility flight software: Driving ambition," in *Aerospace Conference Proceedings*. IEEE, 2006.

[8] X. Yang, M. Moallem, and R. Patel, "A layered goal-oriented fuzzy motion planning strategy for mobile robot navigation," *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 6, pp. 1214–1224, Dec 2005. [Online]. Available: http://dx.doi.org/10.1109/TSMCB.2005.850177

[9] A. Krebs, C. Pradalier, and R. Siegwart, "Rtile - adaptive rover navigation based on online terrain learning," in *Proc. of The 10th International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS)*, September 2010.

[10] G. Ishigami, F. Kazuhisa, T. Kubota, N. Ogawa, and T. Satoh, "Feasibility study of a small, lightweight rover for mars surface exploration," in *Proc. of the 29th Int. Symp. on Space Technology and Science (ISTS)*, 2013.

[11] K. Cavallin and P. Svensson, "Semi-autonomous, teleoperated search and rescue robot," MSc Thesis, Department of Computing Science, Umea University, Sweden, February 2009.

[12] H. P. Moravec, "Sensor fusion in certainty grids for mobile robots," *AI magazine*, vol. 9, no. 2, p. 61, 1988.

[13] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[14] P. Yap, "Grid-based path-finding," *Advances in Artificial Intelligence*, pp. 44–55, 2002.

[15] D. Z. Chen, R. J. Szczerba, and J. J. Uhran Jr, "Planning conditional shortest paths through an unknown environment: A framed-quadtree approach," in *Intelligent Robots and Systems 'Human Robot Interaction and Cooperative Robots'*, vol. 3. IEEE/RSJ, 1995, pp. 33–38.

[16] J. Hirt, D. Gauggel, J. Hensler, M. Blaich, and O. Bittel, "Using quadtrees for realtime pathfinding in indoor environments," *Research and Education in Robotics-EUROBOT 2010*, pp. 72–78, 2011.

[17] S. K. Ghosh, *Visibility algorithms in the plane*. Cambridge University Press, 2007.

[18] D. Hamm, "Navigation mesh generation: an empirical approach," *AI Game Programming Wisdom*, vol. 4, 2008.

[19] T. Lozano-Perez, "Spatial planning: A configuration space approach," *Computers, IEEE Transactions on*, vol. 100, no. 2, pp. 108–120, 1983.

[20] M. Castelnovi, R. Arkin, and T. R. Collins, "Reactive speed control system based on terrain roughness detection," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2005, pp. 891–896.

[21] I. Ulrich and I. Nourbakhsh, "Appearance-based obstacle detection with monocular color vision," in *Proceedings of the National Conference on Artificial Intelligence*. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press, 2000, pp. 866–871.

[22] K. Konolige, M. Agrawal, M. R. Blas, R. C. Bolles, B. Gerkey, J. Solà, and A. Sundaresan, "Mapping, navigation, and learning for off-road traversal," *Journal of Field Robotics*, vol. 26, no. 1, pp. 88–113, 2009.

[23] C. A. Brooks and K. Iagnemma, "Self-supervised terrain classification for planetary surface exploration rovers," *Journal of Field Robotics*, 2012.

[24] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[25] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[26] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[27] S. Koenig and M. Likhachev, "D* lite," in *American Association for Artificial Intelligence (www.aaai.org)*, 2002.

[28] R. Philippsen, S. Kolski, K. Macek, and R. Siegwart, "Path planning, replanning, and execution for autonomous driving in urban and offroad environments," in *Proc. of The Workshop on Planning, Perception and Navigation for Intelligent Vehicles (ICRA)*, 2007.

[29] S. B. Goldberg, M. W. Maimone, and L. Matthies, "Stereo vision and rover navigation software for planetary exploration," in *Aerospace Conference Proceedings*. IEEE, 2002.

[30] J. Carsten, A. Rankin, D. Ferguson, and A. Stentz, "Global planning on the mars exploration rovers: Software integration and surface testing," *Journal of Field Robotics*, vol. 26, no. 4, pp. 337–357, Apr 2009. [Online]. Available: http://dx.doi.org/10.1002/rob.20287

[31] F. Souvannavong, C. Lemaréchal, L. Rastel, M. Maurette, and F. Magellium, "Vision-based motion estimation for the exomars rover," in *International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS), Sapporo, Japan*, 2010.

[32] A. Krebs, C. Pradalier, and R. Siegwart, "Adaptive rover behavior based on online empirical evaluation: Rover-terrain interaction and near-to-far learning," *Journal of Field Robotics*, vol. 27, no. 2, pp. 158–180, 2010.

[33] M. Takahashi, T. Suzuki, T. Matsumura, and A. Yorozu, "Dynamic obstacle avoidance with simultaneous translational and rotational motion control for autonomous mobile robot," in *Informatics in Control, Automation and Robotics*. Springer, 2013, pp. 51–64.

[34] S. D. Karl Iagnemma, *Mobile Robots in Rough Terrain: Estimation, Motion Planning, and Control with Application to Planetary Rovers*. Springer, 2004.

[35] G. Ishigami, M. Otsuki, and T. Kubota, "Range-dependent terrain mapping and multipath planning using cylindrical coordinates for a planetary exploration rover," *Journal of Field Robotics*, 2013.